NICKLAUS GOH

CRC Press
Taylor & Francis Group

DATA DRIVEN GAME DEVELOPMENT

Transforming Player Data into a Better Player Experience

NICKLAUS GOH

CRC Press
Taylor & Francis Group

DATA DRIVEN GAME DEVELOPMENT

Transforming Player Data into a Better Player Experience

Data Driven Game Development

This book covers how data can be used to develop and improve video games from multiple aspects, based on the author's extensive experience working in the video game industry for over 16 years.

This book is a comprehensive tool for learning about data of various types, including analytics, research, and financial. It details a journey of building up from zero data maturity to full data maturity, covering both analytics and UX research departments, using case studies from real-life scenarios. By the end of this book, readers will have a better understanding about gathering and analysing data, then presenting actionable insights that lead to a better game and a better player experience.

This book will be ideal for any aspiring or existing games industry professional looking to understand how to use data to build better games.

Nicklaus Goh has been working with data in the video game industry for over 16 years at Creative Assembly, Sega, and King (Microsoft). He has contributed to dozens of video games, spanning genres like strategy, Match3, simulation, first-person shooters, horror, CCGs (collectible card games), MOBAs (multiplayer online battle arena), racing, and puzzle games. He helped build up an analytics department and then subsequently started and built up his own UX research department.

Data Driven Game Development

Transforming Player Data into a Better Player Experience

Nicklaus Goh

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

Designed cover image: Shutterstock

First edition published 2027
by CRC Press
2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press
4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

ISBN: 9781041230946 (hbk)
ISBN: 9781041230939 (pbk)
ISBN: 9781003735335 (ebk)

DOI: 10.1201/9781003735335

Typeset in Times
by codeMantra

Contents

1 The importance of using data in game development

DOI: 10.1201/9781003735335-1

Congratulations, you've just released your new video game! Perhaps you're a designer, ideating and developing the core experience of the game. Or you're a programmer, transforming that design into a playable concept through code. Or you're a quality assurance (QA) tester, meticulously discovering and documenting bugs to be fixed. Or you're one of the many other professions that are needed to fully actualise a video game into reality.

After years of work, people are playing your game, and hopefully they're loving it. You check out social media and… they don't. Complaints upon complaints are popping up, one after the other, detailing a multitude of problems. Not technical issues or bugs, because programmers and QA made sure the game was as bug-free as possible. No, players are complaining about not knowing what to do, dying a lot, and generally not having fun with the game.

The overall review scores on Steam/App Store/Google Play are plummeting, and so are sales, player numbers, and team morale. You and your team have spent the past few days going through all the criticism, all the player feedback, and you've identified the problems that need to be fixed. You could fix this, but there are too many problems and not enough time. If only you knew about these problems earlier. If only you could go back and do it all again, with everything you know now.

Well, here's your chance. You just need to get the data first.

WHAT IS DATA?

To put it as simply as possible, data is a collection of facts or observations. On its own, data has minimal value with regards to decision-making, much like say, flour has minimal value as something to be eaten. Flour must be mixed with water to create a dough, which has to be combined with leavening agents to cause it to rise, kneaded, and then baked to finally become bread, which has far more consumption value.

So too is it with data; a series of facts and observations on their own rarely translate to effective decisions and results. Data has to be cleaned, processed, and organised in order to transform them into insights. These insights then inform or drive decisions that lead to results. All of which will be covered in the following chapters.

Within the video games industry, data can be sourced in multiple ways, in multiple forms. It can be publicly available in the form of player-written reviews on store platforms like Steam or review aggregators like Metacritic. The games themselves can collect the data, tracking the actions players take, such as killing enemies, using specific weapons, wins & losses, or simply how long they play the game. Companies can recruit players to playtest their games in development, observe how they interact with the games, and interview them directly or provide surveys to fill in. All this, and more! There are a multitude of ways to gain all kinds of data to help drive decisions.

For now, we'll look at some examples of using data to solve problems and make impactful decisions, both inside and outside the video game industry.

2 Historical examples of using data to solve problems

DOI: 10.1201/9781003735335-2

JOHN SNOW AND PLOTTING CHOLERA DEATHS

In the 1800s, the population in London grew, along with cases of cholera and cholera-related deaths. In the early part of the century, cholera spreading via bacteria-contaminated water had yet to be discovered, and people instead believed it was due to bad air or miasma.

With this miasma belief, people would take preventive measures such as burning tar to remove the bad air, burning the clothes of victims, and praying. Since none of these "preventive" measures would prevent people from drinking water contaminated with the bacteria, this also serves as an example of bad data leading to poor and ineffective decisions.

In August 1854, there was an outbreak of cholera in Soho, London, resulting in hundreds of deaths. A physician by the name of John Snow gathered raw data in the form of the addresses of people who had died from cholera. He then processed and organised that raw data by visualising the cholera cases as stacked bars on a map, as shown in Figure 2.1.

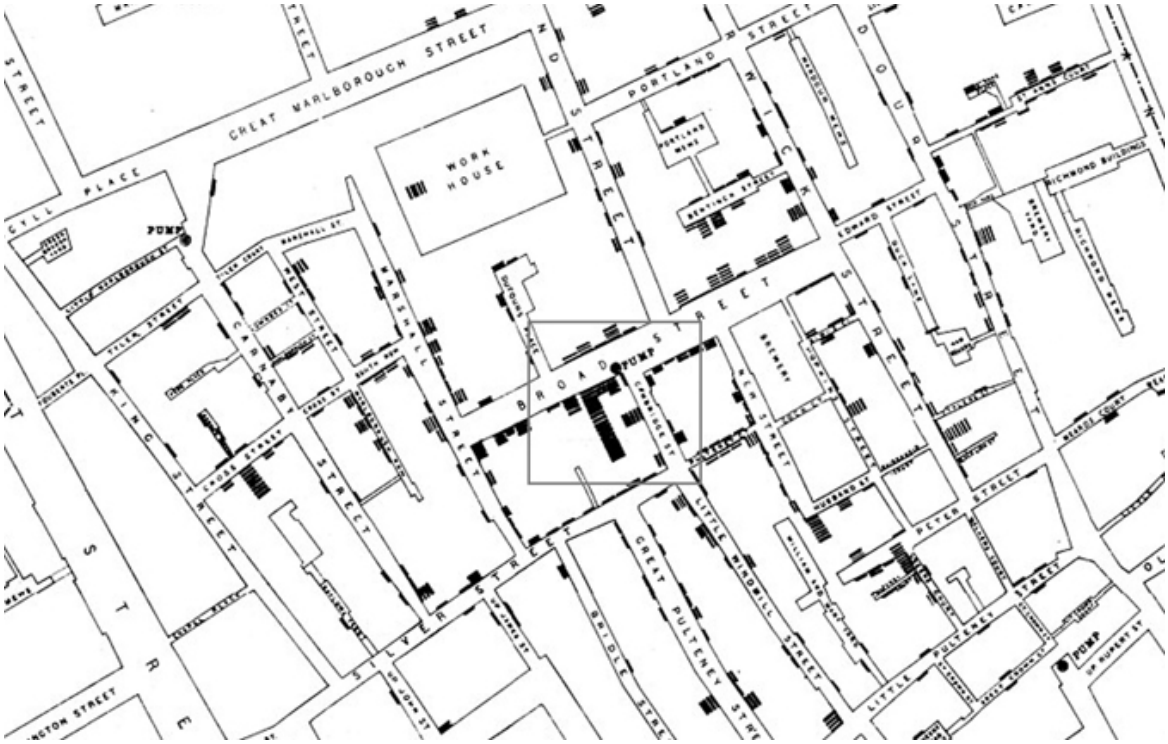

FIGURE 2.1 A map showing the location of cholera cases, indicated by stacked bars. ⏎
Creator: John Snow (1813–1858). Public domain.

With the map, John Snow was able to show that the cholera deaths were clustered around a public water pump, which was used daily for drinking and washing. John Snow also spoke to residents, who provided a corroborating source of data, in that the water from the water pump had an offensive smell. In addition, workers at a nearby brewery had not caught cholera, despite being in the centre of the outbreak. Further interviews revealed that these workers had not drunk water from the pump, but instead drank beer, and thus were spared from the outbreak.

Despite a chemical and microscopic examination of a sample of the water from the pump not conclusively proving any danger, the data used by John Snow persuaded the authorities to disable the pump. When the water pump was disabled, he was able to show there were no new cases of cholera or cholera-related deaths in that area, proving a causational link.

This usage of multiple sources of data, particularly the plot of deaths on a map, led to the insight that cholera spreads through contaminated water, and influenced public health and led to improved sanitation facilities in the 19th century.

ABRAHAM WALD AND SURVIVORSHIP BIAS

In World War II, the Allies wanted to protect planes from enemy artillery. They would do this by adding armour to planes, which had the drawback of increasing fuel consumption due to the added weight. In addition, adding armour to the planes would slow them down and make them less manoeuvrable, increasing the chances they would be shot down. There had to be enough armour to protect the planes, but not so much that it would make return trips impossible or increase the likelihood of being shot down.

Researchers had to figure out where armour could be best placed to protect the planes. To do this, they collected data by investigating planes that came back, particularly those that had bullet holes in them, and plotted the accumulated positions of bullet holes on a plane diagram, providing a visualisation of where planes were being hit the most. Their reasoning was that armour should be placed on the areas of the plane with the most bullet holes, since that is where it would do the most good.

Abraham Wald, a statistician, had an alternative suggestion. The data collected – positions of bullet holes on returning planes – was useful, but it was being viewed through the lens of a faulty assumption. Where the bullet holes were was not important, but rather, where they were not.

The bullet hole data was sourced from planes that had returned, or, to put it another way, planes that had survived being shot. The planes that had been shot down represented missing data, which would have told the Allies the best positions to place armour. That data was, obviously, not available. But it could be extrapolated from the surviving planes. Abraham Wald suggested armouring the locations where bullet holes were not present, reasoning that since planes hit in those areas had not returned, those were where armour would be most effective.

This faulty assumption is known as survivorship bias and will be covered in more detail in later chapters, with examples specifically related to the video game development. Speaking of which, we should look at some past decisions of developing video games with data.

VALVE EVOLVING A LIMITED EXPERIENCE INTO AN AWARD-WINNING EXPERIENCE

In 2007, Valve released a game called Portal. The objective of the game was to solve puzzles using portals that players could deploy and traverse through. Puzzles can be difficult to objectively evaluate, especially for the designers who created and refined them; is a puzzle's challenge just right? Is it trivial to solve? Or will players get stuck and frustrated? The ideal would be to have players

expend some cognitive effort on puzzles that provide just the right number of hints, such that players feel challenged during, and satisfaction upon completion. But once you know the solution to a puzzle, how do you evaluate that puzzle's difficulty objectively?

Valve used playtesting, a method which involves players playing a game during development. The method is useful to gather feedback from players on whether they understand how to play the game, how fun the game is, and anything else the developers might want to focus on. For Portal, the feedback from players was positive, with a twist. Players felt that what they had just experienced was fun, but wondered what these puzzles were leading towards, and couldn't wait to play the actual game.

Unfortunately, those 14 puzzles they experienced were the actual game. Clearly, something was missing. Using this feedback, Valve decided that the game needed something to tie all those puzzles together, something to act as a foil for the player, something to provide context and a motivation to progress. They came up with GLaDOS (Genetic Lifeform and Disk Operating System), an AI that would serve as both a reason for the puzzles to exist and antagonist. The released version of Portal had extended gameplay beyond those initial 14 puzzles, where players escape the artificial puzzle rooms and eventually have a final confrontation with GLaDOS.

GLaDOS was received well by players, and won several awards in 2007, including Best Character award. Portal itself was critically acclaimed and sold millions of copies. All of this might not have been achieved had Valve forgone playtesting and simply released the game in its original state.

UBISOFT USING HEATMAP DATA TO FINE-TUNE UNINTENDED FRICTION POINTS

Ubisoft develops and publishes a series of action-adventure games in a franchise called Assassin's Creed. Each Assassin's Creed game is set in a historical setting, like ancient Greece, Britain, Japan, and more. The player (generally, but not always) plays as an Assassin fighting against Templars who seek peace via world domination.

Players carry out missions where they, as an assassin, navigate terrain and buildings through parkour, use the environment to stay hidden, and either perform assassinations or engage in combat. It is possible to die or "desync" in these missions, resulting in having to start again from the beginning of the mission or from a checkpoint.

During playtesting, Ubisoft would make use of heatmaps, which overlay data points of interest based on where they occurred on a map where the mission took place. If these heatmaps sound familiar, it is because they are an evolved version of the map John Snow used to plot cholera cases. Much in the same manner, Ubisoft plotted player death locations on the mission map, using different colours to indicate the volume of deaths in a location, with green being a low number of deaths, yellow a medium amount, and red a high amount.

Designers can use heatmaps to make decisions on whether the design intent is being met, depending on the context. As an example, for early missions, it is expected that players may be getting used to the game, understanding how to control their player avatar, and getting to grips with the controls. Unless the game's design philosophy is for the player experience to be difficult and punishing, it is expected that earlier missions should help ease the player into the experience, so they will keep playing.

Clusters of red dots on the heatmap indicate that multiple deaths occur in those locations. This could be intentional, perhaps a difficulty spike and a challenge for players to overcome. Or perhaps it is unintended; there are too many enemies, or the difficulty spike is far too steep compared to what the player has experienced so far, or those areas rely on new mechanics that players have not been taught yet. Either way, designers would understand whether those deaths are expected – for example, if a difficult boss was introduced in that area. On the other hand, if nothing particularly special or difficult was implemented for that area, there could be an unforeseen issue causing players to die more than expected. In which case, the deaths data visualised as heatmap surfaces this issue, allowing designers to explore the cause and fix it.

In the above case, the design intent was not for players to die in those areas, and the mission was tweaked to be easier, with the aim of improving the player experience and reducing frustration. This exercise in using data would have been carried out across multiple missions using multiple heatmaps to ensure players were experiencing what was intended by game designers.

DATA INSIGHTS DON'T ALWAYS RESULT IN BIG CHANGES

While these changes were to the benefit of the players and the company, the extra work that the changes required may seem intimidating. Adding a voiced antagonist meant writing extra dialogue, crafting new areas, and designing a final battle against the antagonist, all of which would have to be playtested. Gathering

data on where players die across multiple missions means gathering that data, analysing it, comparing it against what the design intent is, fine-tuning the levels, then repeating it all again to test and confirm. All of this adds significant development time. Is this what it means to use data in video game development? Not necessarily.

Valve was working on another game, Left 4 Dead, a multiplayer cooperative zombie shooter game, where up to four players could play together cooperatively. During playtesting, players had trouble finding teammates, especially during tense situations when they were being swarmed by a horde of zombies. So, Valve made a tweak allowing players to see the "x-ray" outlines of other players, even through walls and buildings. While this was a small change, it greatly improved the ability of players to find each other.

Another example of how data can influence or inform small but effective changes would be Ubisoft and subtitle options in their games. Subtitles are a way for players to be able to have an additional channel of information on-screen when playing the game. For players who have hearing disabilities or trouble understanding accents, subtitles can aid with understanding what is being said. Initially, the option for subtitles was off by default, but Ubisoft noticed that in one of their games, Assassin's Creed: Origins, more than 60% of players were going into the options screen to turn them on.

In the subsequent game, Assassin's Creed: Odyssey, they inverted it such that subtitles were on by default, presumably reasoning that it was more efficient for 40% or less of players to have to turn them off, than the other way around. Instead, only around 5% of players went into the options screen to turn off the subtitles, indicating that around 95% of their players either wanted subtitles on, or were indifferent to them being on. It was a small change, but one that improved the game experience ever so slightly for the vast majority of players.

REFERENCES

John Snow, 1854. *Cholera map*: https://commons.wikimedia.org/wiki/File:Snow-cholera-map-1.jpg

John Snow, 1855. John Snow on the mode of communication of cholera: https://archive.org/details/b28985266/page/n3/mode/2up

3 Decision-making without (or against) data

DOI: 10.1201/9781003735335-3

It is very much worth mentioning that data doesn't have all the answers. Sometimes the creative direction won't have any data available to support it, or sometimes, it will have the opposite. Sometimes, data can be a detriment as much as it can be an enhancer.

FROMSOFTWARE GOES AGAINST THE TIDE

In the mid-2000s, the gaming industry was filled with success stories based on making games more approachable. World of Warcraft, a Massively Multiplayer Online (MMO) game, was released in 2004 and became a huge success, with multiple studios shifting their strategies to create MMOs in an attempt to mimic World of Warcraft's success. World of Warcraft was not the first MMO, nor even the first successful MMO, as there were others before it such as EverQuest and Dark Age of Camelot. However, World of Warcraft became massively popular, due in part to the Warcraft franchise, but also because Blizzard made MMOs more approachable.

ACCESSIBLE TO APPROACHABLE

Approachable: In the past, the act of designing and developing games such that they were easier for new players to play and enjoy the core

experience with as little friction as possible was known as making the game more accessible. Since then, the term “accessible” has shifted away from this definition, and is instead associated with enabling players to engage with games, regardless of their capabilities, disabilities, or circumstances. The term “approachable” is instead used when referring to efforts to make games easier to pick up, play, and enjoy. Note that this does not necessarily mean making the game’s difficulty easier, but rather making it easy to understand and enjoy, particularly for newcomers. A challenging game that is approachable can still be easy for new players to get into and enjoy, while requiring skill to overcome the challenges.

MMOs like EverQuest were less approachable than World of Warcraft would be, due to a combination of having harsher death penalties, more lengthy downtime between battles as the player was forced to recover their health and/or mana, mandatory grouping for most content, and more. World of Warcraft streamlined a lot of this, with more forgiving death mechanics, much less downtime so players could get back into the action quickly, enabling a solo experience for a large part of its non-endgame content, and more. World of Warcraft, in effect, was more approachable than EverQuest and other MMOs at the time were, enabling a wider range of players to be able to pick it up, play it, and enjoy it. And they did, exceeding Blizzard’s expectations and making World of Warcraft the most popular MMO in the world at the time.

Other game developers noticed and followed suit, not just with the aforementioned MMO clones, but with the awareness that making their offerings more approachable enabled more players to enjoy (and buy) their games. Removing friction and punishment from games was desirable in the wake of World of Warcraft and other similar successes.

For the most part, the games industry was trending towards removing friction and making games less punishing. Enter FromSoftware, with an intentionally punishing and frustrating game called Demon’s Souls.

Today, FromSoftware is well known for its Souls (Demon’s and Dark) games, for Bloodborne, for Sekiro, and for Elden Ring, the last of which as of this writing has sold over 30 million copies, with over 13 million of those copies sold in the 1st month of release. Such is their popularity that there is

even a term for this new subgenre of intentionally difficult action role-playing games, called “Souls-like”. Multiple other game studios have developed or are developing Souls-like games in an effort to emulate some of FromSoftware’s success.

However, back in the late 2000s, Demon’s Souls was viewed as a risky bet. So risky, that the then Sony President, Shuhei Yoshida, had this to say about the game in an interview with Gameinformer.

INTERVIEW QUOTE

“For my personal experience with Demon’s Souls, when it was close to final I spent close to two hours playing it and after two hours I was still standing at the beginning at the game. I said, ‘This is crap. This is an unbelievably bad game.’ So I put it aside”.

Sony published the game in Japan but declined to do so for the rest of the world. FromSoftware partnered with other publishers to do so, and Demon’s Souls ended up selling 280,000 copies in 5 months. While nowhere near the same level of blockbuster games like World of Warcraft or Call of Duty, Demon’s Souls sold four times the projected number of units, becoming a moderate success. Notably, Sony would partner again with FromSoftware some years later to develop and publish Bloodborne as an exclusive for the Playstation 4.

Why Demon’s Souls became such a success is interesting and could fill up multiple chapters on its own. But that is beyond the scope of this book, so I will try to keep it down to a few paragraphs. As mentioned before, in the late 2000s, games were trending towards being more approachable by having less friction, less punishment, and including more onboarding or more tutorials to teach players. If the story or background setting was important, effort was made in the form of cutscenes, voiceovers, and other tools, to ensure the player was kept informed as possible.

Demon’s Souls did none or very little of this. It didn’t hold the players’ hands. It was punishing in the form of death mechanics that resulted in players potentially losing progress to increase tension. It had minimal explanatory cutscenes, instead relying on players to explore the story and

lore through their own efforts. A way to contrast Demon's Souls with other games of the time would be this; other games had a constant drip feed of enjoyment and excitement, while Demon's Souls instead had a loop of frustration where players would die repeatedly on a boss or encounter, either getting better through experience or through strengthening their character, eventually overcoming the frustration loop and experiencing a huge rush of enjoyment. And then that loop of frustration into victory would repeat, over and over. From Software presumably believed in this design intent, that there was a market out there filled with players who wanted this player experience. And they were right.

From Software went on to refine the formula they introduced with Demon's Souls across multiple games over the years, increasing sales and prestige with each subsequent game. They had no data that would have suggested that this new Souls-like subgenre would sell tens of millions of copies and inspire others to follow suit. If anything, the data would have suggested they do the opposite of what they did, to reject their creative vision. Sometimes you may not have any data to support your vision, or have data that actively pushes against it, and you have to take a gamble.

Even with data, all decisions are a gamble, just with varying degrees of risk. You will never have enough data to make a risk-free decision that guarantees the best user experience, or the most sales. And data-driven decisions do come with a cost; they take more time than decisions made without data. So, if there is always some risk, if there is added cost and effort, why do we need to bother with using data?

The simple answer here is; there is no right answer. At the end of the day, the answer is going to vary depending on you, the reader. Do you have a strong, defined vision for your game that you believe in? Are there any aspects you are unsure about that might need validation? Do you have stakeholders or backers that require assurances that the project is going in the right direction? Are you trying something new and revolutionary that may need multiple iterations of feedback and refinement to perfect? Do you just want a bit of guidance from player behaviour or player feedback to help realise your vision of a great game?

Whatever it is you need, hopefully some or all of the following chapters will help.

REFERENCE

Interview with Shuhei Yoshida on Demon Souls: https://gameinformer.com/b/news/archive/2012/02/10/shuhei-yoshida-interview.aspx

4 Gathering or using data with minimal resources

DOI: 10.1201/9781003735335-4

If you're reading this section, then you are at the start of your data journey. Perhaps you're part of a small indie company, developing your first video game. Perhaps you're employed at an established company, with one or more video games already released, and want to use data for the improvement of one of your existing games, or the development of your next. Or you might be working alone, designing and coding a video game project for university or to self-publish on a platform like Steam. You might already work with data and want to expand your repertoire, for example, you are a data analyst interested in UX research, or a UX researcher interested in data analytics.

Whatever your needs are, this section will detail the gathering and analysing of data from a position of little to no resources. So, no existing data warehouse, no playtesting lab, no team of dedicated analysts or researchers, no external agency and their costs, just you and whatever support you can get from your team. It will also cover using data from two phases of making video games: during development and post-launch.

PLAYTESTING DURING DEVELOPMENT

When you think about players playtesting your game and providing you with direct feedback, you might think of a playtesting laboratory comprised of multiple rooms, set up with computers or consoles, chairs or couches, enabling players to play your game during development, with cameras for

recording and one-way mirrors so you and your team can observe players' reactions. You might be thinking of a recruitment pipeline, which you'll use to find suitable players, have them sign consent forms, ensure that some sort of compensation system is set up for their travel and their time, whether that be in the form of direct cash payments, vouchers, or even free games. You might be thinking of the legal and confidential aspects of all of this, GDPR, recording personal data, non-disclosure agreements to keep your game confidential, and more.

And perhaps all of this sounds like a lot. You don't have the budget, the facilities, or the resources for any of this. You might even be working from home, without easy access to a company office.

Good news! You don't need any of that to get feedback on your game during development. Whatever your existing workspace is, be it in an office or working from home, whatever you use to play your game will work for others too. You don't need any fancy recording equipment, just yourself and a way to take notes.

For players, the most important criteria when it comes to playtesting are fresh eyes and some affinity towards the type of game you are making, as recruiting someone who dislikes games or dislikes the genre of game you are making will be counter-productive for getting useful feedback. You could ask a co-worker who hasn't worked on the game directly, someone from Finance or Human Resources. Or someone who has worked on the game directly, but not the section you need feedback on. If that isn't possible, then you could ask a family member or a close friend, someone you trust and fits the criteria.

Whomever you recruit, gathering playtesting data can be as simple as sitting a player in front of your game and watching them play, ideally with minimal interruptions to avoid biasing their experience. Afterwards, you can ask them questions or provide them with a survey so they can give feedback about their experience.

All of this can be done without a large budget or dedicated facilities, but the entire process does require some consideration to get the best quality of data. First, you need to figure out what it is you want to get out of the playtesting.

Are you worried about players understanding your new game mechanics or your new take on an old formula? Will players have trouble interacting with your game, whether that be via touch screen, controller, or keyboard

and mouse? Are players getting stuck? Will they find the story you're trying to tell interesting? Will they enjoy the game? Basically, what do you want to discover, what do you want to validate?

Understanding your research needs allows you to plan the playtest accordingly. Worried about the players understanding the game? Have them play your tutorial or First Time User Experience (FTUE) to evaluate this. Worried about the story? Plan to ask questions about whether they understand the premise – it's difficult to enjoy a story if you don't understand what's going on – and whether the narrative is compelling, whether climactic moments are hitting the right emotional beat. Worried about whether they enjoy the gameplay? Ask about this, and more importantly, for everything, ask why. Players may enjoy aspects of your game for different reasons, based on their preferences, familiarity with other games, and expectations.

Why not just ask about everything you need to know? Because playtesting can be intensive for both you and the players. You need to be focused the entire time, prepping to make sure the build is ready, observing players experiencing your game, taking notes to review later or to ask about, asking them questions and follow-up questions, all while being as dispassionate as possible while they provide feedback. And you will need to do this several times across several days, in order to gather qualitative data from different players. This ensures that you are not overly reliant on feedback from a single person, but on a mix of several players. It is more time-consuming, but it reduces the risk that the data you gather is skewed. You will also need to be agile during the playtest, prepared to intervene if players are unable to progress through your game and what to say, prepared to ask new follow-up questions depending on their answers, and prepared to remove or replace questions depending on appropriateness.

All of this might seem quite theoretical, so let's look at examples of interacting with players during playtests.

EXAMPLE 4.1: DON'T ASSUME YOU KNOW WHAT YOUR PLAYERS ARE DOING

Players will surprise you. During a playtest I ran on a game's First Time User Experience section, I observed one of the players traversing

the same area over and over. I suspected he might be stuck but did not interrupt for a few more minutes to be sure, in case I was wrong or he discovered the objective on his own. When he did not, I intervened to ask an open-ended and non-leading question, “How has your experience been so far?”. His reply was interesting; he was enjoying it. I dug deeper by asking if he would mind elaborating on what he was enjoying, and he mentioned that he had found an item that was slightly hidden a while ago. He was the kind of player who enjoyed exploring in nooks and crannies for hidden items or powerups, due to the sense of bonus unexpected advancement it gave him. As such, he was having a good time exploring the area and checking for more hidden items, and once satisfied he had discovered everything, he would move on.

Had I instead asked the player “Are you stuck?” (example of a leading question), this might have led to a completely different answer such as “No, I’m fine”, closing off the insight discovered above.

EXAMPLE 4.2: DEEP-DIVING INTO THE REASONS BEHIND THE PROBLEM

In another playtest for a different game, I had a similar observation; a player was spending a lot of time in the same area, walking past the objective multiple times. I repeated the same procedure as in the previous example, giving him time before intervening and asking the same open-ended question. His reply was the polar opposite; he was frustrated and unsure about what to do. Without mentioning that he had walked past the objective several times, I asked him about what he was trying to do. He was aware that he had to find a certain location (the objective) but had no idea where it was. Further probing showed that he understood the story beats that had led him there, and why he needed to find the objective, he just could not find it. I asked what he had tried to do to resolve this, and he pointed out all the locations he had visited, citing interesting landmarks that might indicate the objective. As none of them were the objective itself, I pointed out the objective, which was a faint glowing marker on the map. He was shocked by this and explained why. In his mind, the marker looked like an environmental special effect that was solely cosmetic in nature, and

he reinforced this by pointing out several other similar effects that were there for that very purpose, to add to the atmosphere of the setting. He had also tried visiting some of those areas with similar effects, and since there was no result from doing so, he assumed through pattern recognition that all such effects were not relevant to his goal. Part of the reason for why the objective marker blended in with the environment was because the art direction for the game was to have the objective markers not stand out as much, to ensure players would be immersed in the game. The feedback and footage of this player (and others) struggling to find the objective, however, convinced the team to make changes accordingly, while still holding true to their vision. They made sure that objective markers, while still blending in, would have no similar non-objective counterpart visually, so players would not suffer from the mistaken association.

The above examples are things you have to keep in mind and focus on during the playtest. What about your players? After all, they will be playing a game, which is what people do for fun. Nothing taxing about that, right? Well, players will also have to play your game while under observation, then answer a series of questions, both of which can be quite intensive and tiring, especially if the play sessions last for a long time. If you are running long sessions, consider budgeting time for the players to take breaks, and let them know this option is available before playtesting starts.

Even with breaks, it is inevitable that players will get fatigued, especially if you are testing a large slice of your game and/or have a lot of questions. Player fatigue contributes to a heavier cognitive load, potentially impacting how they play your game and less thoughtful answers. So, more focused objectives based on your research needs will mean less questions, less chance of fatigued players, which in turn means you'll gather better data.

Finally, if you are planning to conduct any playtesting for your game during development, I leave you with this; always have an open mind when watching your players play. Sometimes they will play the game exactly as you intended, and sometimes they'll find ways to play that you didn't expect, bypassing challenges you've set up or experiencing the game in a different way. It can be tempting to speak up, to fix it, to get them to play the way you intended; you have their best interests at heart, after all. But at

the end of the day, what you want is for your players to have a good experience, to have fun. If they are playing the game in a different way than you expect, consider whether or not this is acceptable. Are they having fun? Does their different way of playing oppose the fundamental design intent of your game, or does it let them achieve the experience you want in a different way? Have your players discovered a different way to have fun that you could incorporate into the design?

WHAT IF MY GAME ISN'T FAR ALONG ENOUGH TO PLAYTEST?

Everything we discussed until now has involved players experiencing your game in some way, which assumes that your game is playable to a degree. Perhaps there are placeholder art assets, or perhaps there's no voice acting, or perhaps sections of the game are missing. But regardless, players can play a slice of your game and provide feedback on their experience.

But what if your game isn't playable at all? What if you're at a crossroads between two or more design directions and want to utilise player feedback before progressing the game further? Thankfully, playtesting a slice of a game isn't the only way. There are still multiple low-cost avenues to gathering data, even without a game to play.

Whatever game you are working on, whatever the genre, whatever the setting, there is likely a competitor out there that has tried it already. It does not matter if the competitor's game was a success or a failure, as there are lessons you can learn from both outcomes. You can play the competitor game yourself, taking notes on what works well and what doesn't, and most importantly, what lessons can be applied to your game. Are there aspects to be emulated or avoided? Are they implementing similar features or systems in a different way, and if so, how does it compare to what you were going to implement? This is a low-cost solution, requiring only the price of the competitor game and some time on your part.

A drawback with playing competitor games yourself is that it can be difficult to ignore the emotional attachment you have with your own game's design. This emotional attachment may affect your evaluation of the competitor title, either positively or negatively, skewing your analysis of the

findings. Much in the same way that it is best practice for surgeons not to operate on their relatives due to emotional attachment causing loss of objectivity, so too can it be for you, and objectively comparing your game with another's. One way to deal with this is to recruit players (co-workers, friends, family) to playtest the competitor game and obtain their feedback, with a focus on learning what works and what doesn't, so that you can import or avoid accordingly.

What if your game is unique, the first of its kind? What if there are no suitable competitor titles to learn from? You can still playtest with a paper prototype. This is exactly what it sounds like: designing your game and playing using pieces of paper (either actual paper or digital versions within Figma or Adobe XD) with the appropriate notation and user interface (UI). This is especially suitable for games in particular genres that fit well with paper placeholders, such as collectible card games, strategy games, and other games that utilise only a few screens. However, even in games like shooters or action games where their fast-paced action cannot be easily captured by paper prototypes can benefit from this. For example, if there are progression systems such as skill trees or in-game shops with their own dedicated UI panels, these can be simulated and playtested with paper prototypes.

Paper prototypes are useful because the act of building them alone is helpful, in that it forces you to think of the intended player experience long before the feature or section you are working on is coded and visualised. In making a paper prototype, you have to make your design playable, implementing theoretical ideas that, when played, may turn out to not work the way you envisioned, without even needing to put it in front of players.

Paper prototypes enable you to iterate quickly, as tweaks and redesigns don't require as much time as coding and compiling new builds. This means that when testing it with your players, you can iterate during the playtest itself, based on observations and feedback. Are players struggling to understand your paper prototype due to ambiguous wording? Reword on the fly and try again. Need to change a mechanic or the numbers behind the scenes? Cross them off and try out your tweaks.

An excellent example of paper prototypes in action can be seen in how the game Hearthstone was made. Blizzard, a company noteworthy for great games like World of Warcraft, StarCraft, Diablo, and others, had a small team working on a card game project that would eventually be known as

Hearthstone. Due to circumstances, many of the team had to be allocated to another project for a time, leaving only two members of the team to work on the game.

The remaining two members turned disadvantage into advantage by using the time to iterate on different versions of Hearthstone. As Hearthstone was a collectible card game, they could use paper cards to experiment and play to find out what worked and what didn't at a rapid pace. After much tweaking and iteration of game mechanics, the Hearthstone they ended up with was effectively the game that launched.

Paper prototypes, despite their name, don't have to be in the form of paper. Interactive prototypes can be built in tools like Figma or Adobe XD, enabling players to interact with them on your computer, or even their own, through a shared link. This is useful especially if you work remotely (or are in the middle of a global pandemic) and can't get others to come play with a physical representation of your game on pieces of paper. It can also be helpful if your company is split across multiple offices, as shown in the example below.

EXAMPLE 4.3: FINDING ISSUES EARLIER ALLOWS FOR BETTER AND CHEAPER SOLUTIONS

During pre-production on one of the titles I worked on, the UI designers wanted to convey a new feature through a series of UI panels, built in Figma during ideation. Because implementing the UI flows from Figma into the actual game would take significant time and budget, they wanted to get it as right as possible during this period, since it would be cheaper and simpler to make changes that way. They also needed insights quickly (within days), as the project was moving at a fast pace.

To ensure that the flow and resulting player experience was as good as possible, I recruited team members from a different office who were not working on the project, since they would be a set of fresh eyes. I set up video calls with them, with links to the interactive Figma prototype provided, and asked them to share their screens during the call. I also invited the UI designers to join the call (with their web cameras off) so they could observe players interacting with the flow. This allowed us to

observe player behaviour through their interactions and ask questions about where and why they were stuck. By doing this, we were able to identify friction points in the UI panels and enable the UI designers to fix them in time before they were implemented into the game. This didn't just make the eventual game better, but also saved us both time and money overall, since it is much more costly to undo or redo features that have been coded and implemented into a game than it is on paper or a Figma prototype.

We've spoken a great deal about low-cost methods of gathering the data during the development of your game, through playtesting, competitor analysis, paper prototypes, and early access. Next, we'll look at data we can collect after your game has launched, to improve it or your next game.

GATHERING DATA AFTER YOUR GAME RELEASES – BUILDING A DATA PIPELINE?

In an earlier chapter, we showcased examples of companies like Ubisoft using behavioural data to inform and validate their decisions. A simple example given was Ubisoft determining that just over 60% of players would go into the options menu to enable subtitles, which were disabled by default. Based on this data, Ubisoft changed the next game to have subtitles be enabled by default, thus benefiting more of the players. The expectation was that just under 40% of players would have to repeat the same actions to disable the subtitles they did not want, which was still a net positive. However, this resulted in only 5% of players disabling the subtitles, indicating more than 60% benefited. This was a better-than-expected outcome due to a simple action, but it was not necessarily simple data collection.

To obtain this data, a company would have to program the game to send back information about players changing that setting. This data would have to include some form of user id, the date and time of when the action was taken, whether the setting was enabled or disabled. The company may want to track players' gaming sessions as well, to prepare for future questions.

For example, do players who disable subtitles play as much, more, or less than those who enable them? Essentially, do subtitles correlate to players playing more, indicating a better player experience? To answer this question and more, considerations must be made against what is needed and the cost to fulfil those needs.

Even if a company knows exactly what behavioural data they want to collect and has added suitable code for their game to collect and send, how does this data go from the player's device (PC, console, phone, tablet) to the company? They need to build a data pipeline.

For best practice reasons, we discuss later in Chapter 6 that the data pipeline cannot be as simple as sending data from the game directly to the company's database. First, the game should send the data to a staging area, usually a platform like Amazon Simple Storage Service (S3). Data from thousands or millions of players can be stored here, while automated processes that you and your team will have to build will clean and transform the data. The transformed data will then be loaded into another destination, usually a database, where further processing may be needed to structure the raw data in a more human-readable manner.

As you can see, even this simplified explanation of the process is a lot just to get the information from the game to you, and it will be covered in more detail in Chapter 6. For now, let's assume there isn't enough time or budget or need for such a complex system. You want whatever data you can get, whatever is easy to access, to help inform decisions. Thankfully, there are lightweight options for gathering behavioural data, with the caveat that most of these are only viable after you have launched your game, so they will only benefit you in supporting that game or in developing future games.

EXTERNAL SOURCES OF DATA

Back in 2003, Valve launched a digital distribution service and storefront called Steam, which eventually became the biggest platform for personal computer (PC) games. Nowadays, players can purchase PC games, download them, and be playing within hours or even minutes, depending on the speed of their internet connection. Steam makes it simple for players to purchase and play games, all on a single platform. It has functionality to

pre-download or preload games prior to launch, enabling eager players who pre-ordered to play their anticipated games as quickly as possible. Steam also offers discounts regularly, providing great deals for players who are willing to wait for lower prices. It also offers services like the ability to add friends and communicate with them via text, achievements, profile pages players can use as a showcase, and more.

Steam isn't just appealing to players, but to game developers and publishers as well. In Steam, Valve offers a digital solution for companies to sell games to players, expanding their reach beyond physical retail stores. As a fee for this service, Valve takes a 30% cut of the price of each game sold, leaving companies with a 70% share. This was seen as a better deal than using retail stores, which take a larger cut of the price than Valve. There is also an additional cost for a publisher or developer to provide retail stores with copies of their game, as physical copies have to be manufactured, stored, and shipped to retail stores across the world, whereas digital copies cost virtually nothing in comparison. Effectively, selling a game through physical stores means that the publisher or developer would only net around 50% of the price per game sold, whereas they would net 70% when sold via Steam.

So, Steam is a popular platform selling millions of games, and as with anything popular, there is a desire to track how much was being sold. In 2015, a website called Steam Spy was created by Sergey Galyonkin. The site was able to estimate the number of sales of games on Steam by polling a sample of user profiles and calculating what titles they owned. Using data sampling and extrapolation, Steam Spy was able to estimate overall sales of games on Steam. At the time, I measured the exact sales figures of games I had worked on against Steam Spy's estimates, and found them to be 90–95% accurate, depending on the game.

For nearly 3 years, Steam Spy became an invaluable tool for monitoring how other games were performing, providing us direction on what to do next. For example, if a competitor title was selling far more copies than us, that would be a sign to investigate what they were doing better than us. Was their game offering a better player experience? Did they have a feature that we were lacking, or a better version? Were they doing better marketing for their game? Conversely, if a competitor title was selling far fewer copies than ours, that might warrant an investigation into why. Perhaps they had a feature or creative direction that players hated, one that we were thinking of

implementing in our next game, and thus should reconsider. Steam Spy's data was also useful for investigating the sales performance of games in genres we might be interested in, to understand who the market leaders were, which then helped guide what we should research and what we should focus on.

I've referred to usage of Steam Spy in the past tense, because in 2018, Valve made changes to Steam that made it more difficult for Steam Spy and other similar sites to estimate the sales figures of games. This measure was effective, as sales estimates of titles I had worked on deviated significantly from the actuals, and Steam Spy's figures were no longer as trustworthy as before. While Steam Spy still exists, the figures it reports now are wide range estimates. For example, as of the time of writing this book, it reported that Hades II had sold between 2 million and 5 million units, which is quite a significant range. This doesn't mean Steam Spy cannot be used for the benefits listed above, but it does mean that some caution must be used when considering which games to evaluate.

Steam Spy is not the only service I've used. In past years, companies like Sensor Tower have emerged to provide data insights on free app downloads, paid app downloads, highest grossing app, and more. Like Steam Spy, Sensor Tower isn't 100% accurate, but it provides a good "ballpark" estimate for how mobile games are performing. If you are working on a game for mobile or tablet, Sensor Tower provides data and analytics on mobile apps for the price of a subscription, which can be a low-cost alternative to building your own data pipeline or conducting your own analysis.

Another useful (and free) source of data is SteamDB, which, as its name implies, is a free database covering Steam's games using publicly available data. It displays concurrent player counts and price history for Steam games, allowing you to view how players engage with games and how other companies plan their pricing strategy (more on that later in Chapter 6).

If your newly released game is on Steam, you can use SteamDB to monitor how many players are playing your game concurrently, without needing to build in any tracking of your own. You can compare how players are engaging with your game – when do they tend to play more? On particular days? At certain times? This data can tell you if players are playing your game after work (daily peaks in the evenings), or dedicating blocks of time for gaming (weekends). These peaks can determine your

post-launch strategy. If you want to launch an update for your game to provide players with fresh content to experience, and you know your players tend to play on Friday evenings and on the weekend, you can aim to launch your update on Thursday, giving players time to find out about the update and download it before their big play sessions.

The concurrent player count on SteamDB also enables you to track how effective your updates are. Is there a large spike in player numbers after your update? How does it compare to previous updates? Player numbers tend to decline over time, as players grow bored, feel satisfied, move on to other games, and so on. How effective are different updates in keeping players engaged and playing your game? Which types of updates have longevity, and can you use this to inform your future update strategy? Analysing this data enables you to understand which of your updates are effective, and which are not, prioritising what works and what doesn't to keep your players engaged and enjoying your game. While it may not be as in-depth or extensive as data from your own pipeline, it's free.

We've spent some time going over how to collect data at a low cost, but sporadically, doing what can be done where possible. In the interest of building towards a mature data-informed organisation, we'll need to start focusing on best practices and building a foundation of extensive data-informing decisions.

But before that, we need to talk about bad data.

5 Bad data examples and consequences

DOI: 10.1201/9781003735335-5

Bad data is worse than having no data. I was hiking in Japan a few years ago on my own, heading to the site of a historical castle ruin. It was out in a rural area, and I wasn't getting the best signal, so Google Maps was being finicky. I reached a split, with one path going left, the other right. There was a sign in the middle of the split, written fully in Japanese kanji, which I was not fluent in, but I could make out the character for "castle". Below that sign was a small wooden arrow, pointing left. So, I went left. And then I walked for over an hour before my phone got a good signal, and Google Maps told me I was going the complete opposite direction. When I made it back to the split, I tentatively touched the wooden arrow, and it spun easily on the nail it was hanging from. I pointed it to the right, and went back to my hotel, as it was getting late. Bad data – the incorrect arrow – had wasted my afternoon, although I did get some exercise out of it.

Had the sign not been there, or had the arrow not been there, I would have had no data. Perhaps I would have gone left (wrong choice), or right (correct choice), or decided not to take the risk of either path, backtrack and go somewhere else instead (safe choice). Either way, my decision-making process would have been completely different. With no data, I had a chance of making the correct choice, a safe choice, or the wrong choice. With bad data, I could only make the wrong choice.

We've touched on bad data in an earlier chapter, when we spoke about John Snow and cholera cases. In the 1800s, the miasma theory was a now-abandoned medical theory that diseases like cholera were caused by a

noxious form of air originating from decomposed matter. Breathing in this "bad air" was thought to be the cause of epidemics.

This miasma theory has origins as far back as the 1st century, and it wasn't until the mid-19th century that investigations by people like John Snow began to make headway in disproving this theory, to eventually be replaced by germ theory.

However, before it was debunked, miasma theory was an excellent example of how bad data can impede the ability to make effective decisions. Treatments for illnesses thought to be caused by miasma revolved around preventing or removing the miasma. Examples of such preventions included burning incense to produce a smell that would counter the bad air, purifying the air by keeping pleasant-smelling herbs such as lavender on one's person or in the lavatory. While these made for a more appealing environment to be in, they did nothing to actually prevent diseases.

In the 17th century, plague doctors wore outfits to protect themselves while treating plague victims. The outfit's items included a beak-like mask, an overcoat, gloves, boots, hat, and hood. The beak was designed to hold herbs or flowers or other strong substances to filter out the bad air. Even the eyeholes were covered in glass to shield the wearer's eyes and face from the miasma. It was thought that wearing this mask would protect doctors as they tended to sick patients. Notably, because the outfit also included a long overcoat and gloves, this had some minor benefits due to preventing bodily contact, exposure to fluids from a sneeze or cough, and flea bites which could transmit the plague. However, these benefits were outweighed by the fact that plague doctors were entirely ignorant of them, and thus fluids such as blood, mucus, and pus would accumulate on their outfits, protecting the doctors but making them a vector for the disease as they went from patient to patient. Once the doctors were away from the "miasma" and thought themselves safe, they would take off their contaminated outfits without any regard for the dangers of doing so carefully.

All this time and effort was funnelled into solutions for miasma, all mostly wasted, because there was little chance of them being effective since the bad air theory was based on bad data. Armed with the miasma theory (bad data), people of that time were investigating solutions in the wrong direction. Without it (no data), the time and effort wasted on miasma solutions might have resulted in discovering germ theory and better

preventive methods sooner. Or perhaps not. Perhaps the miasma theory would have been discovered anyway. But with no data, there was a chance. With bad data, there was none.

In order to build a solid foundation of data-informed decision-making, we need to understand the difference between good data and bad data, to be able to recognise and avoid the pitfalls, and to resist the temptation to take shortcuts. And above all, to recognise mistakes and learn from them.

BIASES

Bad data tends to originate from biases, which are inclinations to be for or against someone or something. Everyone has some form of bias, and they are usually in the form of a mental shortcut. For example, a mental shortcut that may occur when looking at someone who is physically attractive would be to assume that because they are attractive, they must have other positive attributes, such as being smart, kind, funny, or thoughtful. This can also work the other way around through a bias known as lookism; someone who is physically unattractive may be judged by the viewer as having other negative attributes. Obviously, there is no rational merit to these assumptions, but this bias and others like it exist, because they allow our brains to make quick and easy judgements, without having to spend too much cognitive effort.

There are many, many kinds of biases, but for the purposes of this book, we'll look at the more common ones that plague data analysis and UX research.

SURVIVORSHIP BIAS

Cavemen paintings

When you read the term "prehistoric human", what comes to mind? If it's a person primarily living in a cave, then you are already affected by survivorship bias. Prehistoric humans primarily lived in open-air settings,

close to sources of water. Caves provided temporary shelter and refuge from harsh weather or predators, rather than a permanent home.

The image of a caveman is closely associated with prehistoric humans, because a great deal of our knowledge of prehistoric humans comes from cave paintings and artefacts preserved in caves. Any kind of paintings or artefacts left behind out in open-air settings would have been eroded by the elements over time, whereas the same protection that led prehistoric humans to seek shelter in caves also ensured that cave paintings and other cave artefacts would have a better chance of lasting throughout the centuries. In other words, our sample of data related to the activities of prehistoric humans is skewed heavily towards what has survived in caves, hence the term survivorship bias.

We've touched upon an example of survivorship bias previously, when Abraham Wald identified said bias in the analysis of planes returning from battle. He refuted the idea that areas on planes with the highest density of bullet holes should be armoured, correctly identifying that this analysis was tainted by survivorship bias. Planes with the highest density of bullet holes survived and returned in order to have their data collected, with the keywords there being "survived" and "returned". Planes that did not return represented the missing data, and this missing data tells us that armouring areas with zero density of bullet holes was the best way to protect planes.

The "survivors" in survivorship bias aren't limited to literal life or death situations. Another example of survivorship bias is rooted in stories of individuals whose dedication to pursuing their dreams at all costs led to incredible success. For example, Bill Gates, Steve Jobs, and Mark Zuckerberg are all incredibly successful, having founded or co-founded Microsoft, Apple, and Facebook, respectively. All three men also have something else in common; they dropped out of college to pursue their dreams and achieve this success.

Survivorship bias might lead you to the conclusion that dropping out of higher education to pursue your dream will lead to fame and riches. And this does happen! There are multiple success stories like this, not just in tech and business, but in sports, in entertainment, and more. Rags to riches stories, underdog beating the odds stories. What all these stories are missing is the multitude of unmentioned failures. The college dropouts who pursued their dream and didn't go on to found Microsoft or Apple or Facebook. Maybe they had moderate, minor, or no success in their fortunes after. We

don’t know, because there are no widespread stories about them. The survivorship bias here is only looking at the dropouts who are successful, while ignoring or being unaware of those that were not and falsely believing based on this incomplete data that dropping out to pursue your dreams has a high chance of success.

Decisions within the video game industry are as vulnerable to survivorship bias as any other. Very early in my career, I worked on a mobile game project for an established video game franchise, where both the genre and the platform of the new game were unfamiliar territory to the company. Due to this unfamiliarity, the leadership team wanted multiple sources of data to prove that the new game could succeed, before they would allocate more budget. One of these data sources was an evaluation of how well other similar games were performing, in order to get an idea of how successful a game in this genre could be. To do this, the creative director of this game used the Google Play store to investigate how other games of a similar genre were performing, based on their downloads.

Google Play does not report the exact number of app downloads, but instead displays rough estimates, e.g. 5M+ downloads, 10M+ downloads, and so on. The creative director was able to find several games in a similar genre that each had 5M+ downloads on Google Play. None of these games were from established franchises, or from known video game developers, and no one on the game team had heard of them before. This was the evidence that the creative director used to prove to leadership that our game would succeed, as it only needed a few million downloads based on projected costs and revenue. If all these other no-name games could achieve millions of downloads, our game in an established franchise with a known brand, made by a company with a pedigree of best-selling video games, would surely be capable of achieving the same.

While I did not realise it at the time (and should have), this was a clear case of survivorship bias. The Google Play store search (and any service that provides search functionality) will always try to prioritise giving you a good experience based on your search request, so that you’ll be satisfied and use it again. Here, a good experience means that your search should not only just be pertinent to what you are looking for but also be of a high quality or popular with other people. This means that any searches for games without naming specific titles will be extremely likely to surface games that are successful, usually with a high number of downloads. What

the search is unlikely to surface is games that were unsuccessful. You'd have to go out of your way to find these unsuccessful games, and in some cases, you may no longer be able to, as companies usually stop supporting such games and remove them from storefronts to reduce ongoing costs, such as server costs. In addition, Google and Apple will also remove apps that have not been updated in 2–3 years. In short, unsuccessful games are less likely to be surfaced by the search algorithms and may no longer be on the storefronts to find, whereas successful games will be.

This survivorship bias meant that the creative director was not looking at a broad range of successful and unsuccessful games in the genre we were diving into. Instead, he was only looking at the winners, the survivors, and the games that had succeeded, some of which were not part of any known franchise. Had we been aware of this survivorship bias, we could have investigated more thoroughly and used other sources to check if this genre was truly filled with mostly games that were successful, with few failures. While this would not necessarily have led to the cancellation of the game we were making, more thorough data would have provided much better context to the challenges of developing it, rather than the false confidence that the game was likely to succeed, given our pedigree and the apparent chance of success given the performance of no-name games. That false confidence was misplaced, as the game failed to reach a large audience, and was shut down after a few years, following in the footsteps of the other non-survivors.

A lot of this was discussed in hindsight, after the project ended. At the time, no one, least of all myself, brought survivorship bias up, because we all missed it. Had it been brought up, perhaps things would have been different. Perhaps not. It was a useful experience in how survivorship bias can lead to misinformed decisions, which became useful some time later.

One of the games I had worked on performed well critically, but not commercially. Because of this, company leadership was interested in a sequel, but only if it could be made commercially viable. As such, there was a strong desire to find new ways to make the sequel perform better, either by lowering costs, increasing sales, or a mix of both. While some costs could be saved due to a sequel being able to reuse assets, the newly developed in-house game engine, and optimised processes due to the development team being more experienced with one title under their belt,

some of these savings would be eroded with inflation and increasing development costs.

A consideration at the time was to pivot the sequel into an episodic format, similar to what Telltale Games used for their *The Walking Dead* game. *The Walking Dead* game was based on the popular TV show (itself based on the popular comic book) and told a narrative story through episodic releases, where the player's key decisions would be recorded and carried over from episode to episode, impacting the narrative.

The Walking Dead game was very successful, both critically and commercially, winning multiple awards and selling millions of copies. There was a desire from leadership to emulate this success, using *The Walking Dead* game sales as a benchmark for forecasting the units and revenue for our potential sequel.

FORECASTING BRIEF

In addition to investigating player behaviour through their actions in-game, part of my duties within the analytics team included using historical sales data to help with forecasting the potential performance of our upcoming games. With large game development companies, especially those that are funded by a publisher, forecasting is useful as it provides leadership with a rough estimate of how much revenue a game could generate, usually in the form of a range of values from conservative to realistic to optimistic. This range subsequently helps to inform budget approval, whether that be for development or marketing, with the ultimate goal of ensuring that a profit is generated. We would then present to leadership with several scenarios for how well the game might perform based on historical data or competitor data, using worst-case, base-case, and best-case scenarios.

Forecasting can be as simple as using the financial performance of the previous title, if you are making a sequel, and accounting for some uplift in revenue based on improvements. Or it can be as complex as requiring a completely new set of data to develop a new game in a new genre for a new franchise.

Mindful of the lessons learned in survivorship bias, we sourced data on how other episodic games had performed commercially, and in doing so confirmed that *The Walking Dead* was a market leader for episodic games. While this did not preclude the potential sequel for our game being a massive success, it would mean that to fulfil leadership's request to use *The Walking Dead* sales as a template, we'd be forecasting our sequel's performance based on the top-selling game in that genre and format of the time. Essentially, we would be betting on having the absolute best-case scenario, focusing on the sole winner of episodic games, and ignoring other lower-performing games of that genre. In the end, the episodic sequel was deemed too risky and was not approved.

SAMPLING BIAS

When gathering any kind of data or conducting experiments, it is rarely possible to query the entire user base or population of a sufficiently large size. For example, if you wanted to discover the pop-culture preferences of the population of Europe, it would be infeasible to poll every single European. If your game has over 3 million sales, and you wanted to understand what all your players thought of your game, it would be just as impossible to ask for their feedback. The cost of such an endeavour, the feasibility of carrying out the task, or the time required to gather and analyse the data accurately would all be serious blockers. An exception to this is in analytics, where it is possible to create a data pipeline that enables you to track the behaviour of all your players.

Outside of analytics (and sometimes even within, for cost reasons), sampling is a common method to gather data from a fraction of your population. There are two important factors to remember when sampling to ensure that your sample is roughly representative of your population. First, your sample should look as much like a mini-version of your population as possible. If your population consisted of 50,000 women and 50,000 men, any sample you drew from the population would ideally have a near 50:50 split between women and men. If you wanted to query hospital staff that was evenly split between doctors, nurses, and support staff, and your sample only contained doctors, that would not be a representative sample, as it would be missing the views and behaviour of nurses and support staff.

If you were able to simply pick directly from your population, selecting at random is a good way to ensure your resulting sample is a mini-version of your population. This ensures fairness and avoids any personal bias from creeping into the selection process. In reality, the selection process will never be fully random. For example, you may want to get feedback from players who are new to your game, so selecting entirely at random would be counter-productive for this, as you'd get both new and existing players. Instead, you'd first filter with a characteristic in mind, which in this case would be "players who are new to the game", and then select randomly from the resulting segment of your population. If your feedback process (e.g. a survey) contains the filter, then you might instead select random players from your entire population to fill out the survey, and have a question within the survey filter out those who are new to your game based on their answer.

Second, your sample has to be of a sufficient size that you can be confident that their views and behaviour roughly reflect your population. A population of 50,000 women and 50,000 men will have a multitude of views and preferences. For example, some women and men will have ice cream as their favourite dessert. Others might prefer apple pie, or cake, and so on, resulting in a diverse distribution of desserts. If your sample has 1 woman and 1 man, it has a 50:50 ratio matching your population, but if you were to ask members of your sample what their favourite dessert was, you'd have only two favourites at most, or potentially a shared favourite, which would not match your population's preferences. A sample of 500 women and 500 men, however, would be a closer match for the distribution of favourite desserts within your population.

So, what is an appropriate sample size? Unfortunately, there is no "correct" answer for this. The larger your sample, the more likely it will be representative of your population, but correspondingly so will the cost and time associated with sourcing this sample. A generally acceptable size is around 1,000, assuming your population is larger than that number, as a sample size of 1,000 has a margin of error of around 3%.

In very simple terms, the margin of error with regards to a sample size describes how much the sample might differ from the population. Using the previous desserts example, let's say that you recruit a sample of 1,000 people, and 50% of your sample say that ice cream is their favourite dessert. Because the margin of error is around 3%, the real percentage of your

population that has ice cream as their favourite dessert will be 50% plus or minus 3%, resulting in a range of 47%–53%. This means that based on the results from your sample, you can be reasonably confident that somewhere between 47% and 53% of your population would prefer ice cream for dessert.

If we decrease the sample size to 100, the margin of error grows to around 10%. If we had a similar finding of 50% of the sample preferring ice cream, you would be less sure what % of your population prefers ice cream, since the true value would be somewhere between 40% and 60%.

If we instead increased the sample size tenfold, to 10,000 people, this would result in a margin of error of around 1%, giving you a range of 49%–51%. Notably, while this is slightly more accurate than the 1,000-size sample, it also requires a tenfold increase in your sample size, and correspondingly, the cost of sourcing it.

This is why there is generally no "correct" answer for sample size; the larger the sample, the more representative it will be, and the more accurate your results, but with higher costs and diminishing returns.

While sampling has advantages such as lowering time and cost needed for the study, it does have disadvantages. We've discussed how and why the sample should be as representative as possible of the population, but the reality is that sampling bias can and often does occur.

Sampling bias is when a sample is selected in such a way that said sample is not a good representation of the population. The biased sample either over- or under-represents certain portions of the population, leading to a skewed mini-version. We've already seen an example of this earlier, as survivorship bias is a form of sampling bias. We'll look at another historical example below.

Car crash test dummies

Cars as a mode of transportation are fairly ubiquitous around the world. They are used by all kinds of people, mothers and fathers, sons and daughters, brothers and sisters, and women and men. A variety of tests are performed to ensure that cars meet safety standards, one of which is collision testing or crash testing. Crash testing involves having the vehicle impact another object, which can range from a wall (frontal impact) to a

pole (small overlap impact) to another vehicle (moderate overlap impact), and more. The goal of these tests is to gather data on how the occupants (driver and passengers) of a vehicle are affected by the crash, to best implement measures to reduce the risk of injury and death.

In the past, test subjects took the form of cadavers, animals, or live volunteers. Each of these test subject types had their own drawbacks. Cadavers, while obviously in the form of a human, suffered from, well, being dead, and thus did not react to impact in similar ways as a live person would, due to factors such as rigor mortis. Animals were, at the time, an acceptable live simulation, particularly pigs who shared a great degree of anatomical structure with humans. However, this was (and still is) animal cruelty, since animals feel pain and could not consent to being used in this way. Live human volunteers were in some ways the best representation at the time, but as a data source naturally suffered from a lack of willing candidates and also shared a universal drawback with the other two sources, the lack of standardisation for consistent testing.

Nowadays, anthropomorphic test devices, or crash test dummies, are used. These dummies are able to record data on what happens to them during a simulated crash, such as the speed of impact, the forces they experience in the form of crushing, bending, folding, and more. Naturally, this data is very useful for researchers to understand and analyse what happens to people during a crash, without having to use actual people. Crash test dummies can also be mass produced, allowing for repeated test variations to be carried out while standardising the attributes of the participant, a desirable trait when testing.

In the past, crash test dummies of a height approximate to the 50th percentile of adult males were used. The height of males if plotted onto a graph would look like a bell curve, meaning that a crash test dummy representing the 50th percentile matches the real-life outcome quite well for the largest percentage of males, and less so for the outliers.

SOMETHING TO CONSIDER

Throughout the previous paragraph, a key word indicates where the sampling bias for crash testing has originated from. Have a think about what it is before continuing to the next paragraph.

We know from real life that driving isn't an activity exclusively carried out by men, with the split likely being closer to even between men and women. Yet, for most of crash testing's history, the crash test dummies have been male, with average male attributes. On average, women are shorter than men and weigh less due to a condition known as sexual dimorphism.

SEXUAL DIMORPHISM

Sexual dimorphism is a condition where different genders of the same species display different characteristics. For example, lions have very obvious sexual dimorphism; the males have a distinctive mane and are larger, while the females are smaller whilst being the hunters.

Sexual dimorphism in humans matters for car crashes, because size differences can mean that protective measures like seat belts and airbags may not be as effective when deviating away from the average size (shorter, taller, fatter, etc.). Treating male crash dummies as representative of the world population, a clear case of sampling bias, means that subsequent safety measures implemented are not as effective at protecting women as men, endangering lives needlessly. Thankfully, this has been changing in recent times, with Swedish engineers developing a crash test dummy based on the body of the average woman in 2022, and other countries following suit.

What about video game testing? I was once asked to run a playtest using influencers and esports players. Influencers are people who promote products or services via social media, and with regards to video games, this usually means streaming the game they are playing on Twitch, YouTube, or other similar social media platforms. This can be non-interactive (the influencer plays the game, usually competitively, without engaging with their viewers), semi-interactive (influencer thinks aloud while they are playing), or interactive (influencer reads viewer posts while playing and responds to them). Video game influencers can sometimes play games less for their own enjoyment, but rather to entertain their audience, since a larger audience means more money in the form of subscriptions and donations. Influencers can gain a huge audience and significant amounts of donations

by being entertaining to watch or even simply playing a highly anticipated game.

Esports players are the video game equivalent of sports athletes. Much like how a football player or baseball player undergoes training to become an expert in their chosen game so they can play professionally, e-sports players must do the same. This involves playing in leagues (usually team-based games like *Counter-Strike*, League of Legends, etc.) or participating in tournaments (either team-based or individual, and spans a wider variety of games like Street Fighter or Smash Bros) where players can be sponsored and/or win cash prizes.

Circling back to the playtest, you can see that influencers and esports players are not representative of the gaming audience as a whole, since most players aren't playing to entertain others, nor are they making money by playing games. Now, if your game is targeted solely at these cohorts, then that would be a different story, but as of 2026, you would also have quite a limited audience. Because of this non-representative sample, I queried what the goal was for this playtest and was told that it was partly to introduce the game to influencers and esports players who might go on to become champions of our game on social media, and also to gauge how these players might feel about our game, informing some of our future marketing efforts. Consider that influencers who play video games can have tens or hundreds of thousands of viewers, and that esports players also tend to stream when they are playing, so they can be potentially good sources of advertising for your game.

From the answer I received, it seemed like stakeholders were aware of the limitations of the playtest and would not use the data as a representative sample for how our potential players would feel about our game, so the playtest went ahead. Since this story is being told in the context of sampling bias, you can probably guess what happened next. Needless to say, after the playtest was over and data collected, the net result of the findings was that both influencers and esports players enjoyed the game. Overall, we gained some useful feedback on how these players would react to our game, as well as aspects that could be improved to help them engage with their viewers (and correspondingly increase the marketing reach of the game).

Unfortunately, parts of the report were then recontextualised into a second report indicating that our playerbase was enjoying the game and would be likely to play it on launch. I later learned that this second report

was created in order to show that the game was performing well as part of the regular validation process with the leadership team, with no mention of influencers or e-sports players in this second report. Putting aside whether or not those cohorts would have enjoyed the game on launch, this usage of this data is an excellent example of intentional sampling bias in action. It led the leadership team to have unfounded confidence in the game, believing that our players as a whole would enjoy it, when in fact an extremely small slice of our players had reported this.

Why was this second report created? Unfortunately, this particular game had been performing badly during testing, with low interest from our target audience, and those who did try it did not play it for very long. Because of this poor performance, there was a strong desire from leaders on the development team to seek out data that proved that their creative direction was working and the game was doing well, buying them time to hopefully resolve its issues. It worked for a time, but ultimately the game failed to resonate with players and was cancelled.

While the method used to generate the second report was sampling bias, the tendency to seek data to fit a desired narrative (which is a very easy trap to fall into!) is known as confirmation bias.

CONFIRMATION BIAS

People have personal opinions and beliefs that drive them. This is good, because it allows for a variety of ideas and solutions. But when it comes to data, it can be dangerous to let personal opinions drive the data gathering and analysing process, as this can lead to a desire to look for or interpret information in a way that reinforces existing beliefs. It can also lead to a denial of proven facts that would contradict those beliefs. This tendency is known as confirmation bias.

An example of confirmation bias in play would be the continued existence of the Flat Earth theory. Hundreds of years ago, it would be understandable to believe the Earth was flat. In the absence of the discovery of gravity and how it works, a flat Earth makes sense; else, how would objects stay on the ground? It would also have made sense because visual stimuli are an important part of people’s everyday lives, and the horizon

looks flat. But this is today, not hundreds of years ago, and we have absolute proof that the Earth is not flat.

Yet the Flat Earth theory persists, and a YouGov survey in 2019 revealed that 3% of British people think that the Earth is ‘probably’ or ‘definitely’ flat. Since the British population in 2019 was around 67 million people, this means roughly 2 million people in Britain believed the Earth might be flat.

Flat Earth theory persists for a variety of reasons: distrust in authority, conspiracy theories, echo chambers, and confirmation bias. The latter exists in Flat Earth believers actively seeking out information that confirms the theory, while ignoring evidence to the contrary.

Confirmation bias exists in everyday life, in people choosing to read articles or watch channels that support their political beliefs while rejecting opposing views, or in science, via researchers who want to prove a theory choosing to seek out data that confirms their hypothesis. Confirmation bias is particularly dangerous, because it can lead people to seek out echo chambers that constantly reinforce their beliefs and insulate them from diverse views.

One of our projects in development was shaping along nicely, with internal developer playtests reporting that the game was fun to play. We wanted some fresh eyes on the gameplay experience, but did not yet have any internal UX Research resource, so we hired an external agency to conduct the research for us, and granted them access to a playable build of the game.

After a few weeks, the agency sent over their report, which had players describing the game as being reasonably fun to play, perhaps not quite as good as the competitor titles that were already released, but something that could be improved over time. Overall, players ranked the game as a B-minus, and brought up deficiencies where our game was lacking compared to competitors. This seemed promising, as the game was still in development, so with time, we could improve that B-minus into something much better.

Unfortunately, leaders in the development team had a different reaction; they disagreed with the game’s rating. In their minds, the game was already on par, if not better than competitors, so this report was not to be taken seriously. None of the feedback was actioned, and we were not allowed to use the agency again. I asked about this, but received no official answer beyond an offhand remark, “We’ve been making games for years, what

does that agency know?" (Some fault for this attitude and decision likely belongs to sunk cost fallacy, as the game had been in development for quite some time, with an already spent budget to match).

The game was eventually released to lacklustre results, far below what was needed to break even, let alone make a profit. The development team attempted to revamp it, but ultimately even the revamp was not successful, and all development efforts were eventually ceased.

SELF-SELECTION BIAS

We've talked about ensuring your sample is representative of your population and being mindful of sampling bias. However, there are biases that can be impossible to avoid, only minimised, depending on how you are obtaining your sample. Online surveys, for example, are by their very nature voluntary. You could share the survey link via social media, send it directly to players' emails, surface it in-game, and in every possible scenario, players can and should be able to ignore it. While it is possible to avoid this by making an in-game survey mandatory in order to continue playing the game, I would advocate against this extremely strongly, as this will lead to player dissatisfaction and the data being noisy.

NOISY DATA

Noisy data is a subset of bad data, and contains a large amount of meaningless information (noise) alongside the useful information (signal).

With regards to a survey, useful information is provided when people read and think about the questions, before submitting their answer.

Meaningless information occurs when people misunderstand the questions, or worse, don't read them and simply select answers at random to complete the survey as soon as possible. This noise distorts the data, making it difficult to identify signals, and in extreme cases, can overwhelm any useful information you might have.

Forcing players to fill out an in-game survey increases the likelihood that some of them will fill out the survey at random in order to get back to the game as soon as possible. While this gets you more completed survey submissions, it comes at the drawback of potentially sufficient noise to negate the purpose of the survey, getting useful information.

While ensuring your survey is voluntary guarantees less noisy data and a better player experience, it does have the drawback of introducing self-selection or volunteer bias. As the name indicates, it is a bias which can introduce error into your research whereby the people who choose to complete your voluntary survey may not represent a fully representative sample. Your target audience will always include players who are motivated to provide feedback, and those who are not. Because voluntary surveys will primarily receive responses from players who are motivated to do so, whether that be for positive or negative reasons, a sample sourced this way will always be unrepresentative to some degree.

This is true not just of surveys, but for playtest recruiting, focus groups, or any mechanism by which players must choose to participate in providing feedback. Self-selection bias is unavoidable, but we can mitigate it as much as possible through various methods derived from the why behind unmotivated players.

Unmotivated players have a variety of reasons for not wanting to provide feedback, ranging from a lack of interest, believing that it is a waste of time due to lack of action taken, concerns about data security, or simply not having the time to do so. However you plan to recruit players, whether it be for a survey, a focus group, direct interviews, or a playtest, here are some ways to mitigate these reasons.

Provide some form of compensation, either in the form of cash, vouchers, in-game currency, or even free food if the activity is held in-person. This helps mitigate the lack of interest in providing feedback; just be diligent with your recruitment criteria, as the compensation can attract non-representative players who don’t care about your game. Recruitment criteria changes depending on the goals of research; if you’re testing the onboarding experience for new players, then the criteria might be gamers or non-gamers who have never experienced your game. If you’re testing changes to a long-standing game mechanic, then the criteria might be experienced players of your game to gauge and understand their reaction to the changes.

Short, simple, and to-the-point questions

When running a study, it can be tempting to ask as many questions as possible, to try and answer every concern you might have about your game. This is not advisable, for more reasons that are discussed in later chapters about UX Research. In the context of reducing self-selection bias, surveys, interviews, focus groups, or playtests that take a long time can be a turn off for players lacking free time. Having short and simple questions helps with reducing the time needed, as does focusing only on questions that matter to you at the time. For surveys in particular, keeping them short and to the point allows you to advertise a short completion time to players, e.g. “This survey will only require around 2 minutes of your time”, which can help increase participation.

Simple questions have additional benefits in that they are more likely to remove ambiguity and reduce the risk of players misunderstanding what is being asked, as well as lowering survey fatigue, which can occur when players have to answer questions for minutes on end, leading to mistakes or a desire to quit before completing the survey.

Consistent behaviour of taking on data and taking actions

Convincing disillusioned players that their feedback matters can be a difficult and long process, but is worthwhile (both for you and them). Feelings of disillusionment usually arise because players have provided feedback before, only to see no action taken, on a repeated basis. Obviously, not every bit of feedback can be actioned, sometimes because the action costs too much, and sometimes because the feedback itself is not actionable due to being technically infeasible.

Sometimes the feedback is actioned but not publicised. Doing so can help a lot with players feeling like their voice matters. Even when action cannot be taken, it can be worth mentioning why it could not be taken, telling players that they are still heard even if action will not be taken at this time. Publicising what actions you’ve taken (or not taken) based on feedback can be publicised via patch notes, or a developer post on your

website, or on social media, something to make sure players feel that they are heard.

Disclaimers about how data is protected

This may be required by your legal department anyway, but it is good to add here regardless. Having a disclaimer at the start of your activity can help reassure players that their personal data or anything else they might say will be protected, only used within the company to improve the game, and not used in any other efforts such as marketing.

Statistical weighting

If your player population is 50% female and 50% male, but your survey results are skewed 25% female and 75% male (or vice versa), then you can apply weighting multipliers to address the imbalance. In this example, you'd treat every answer from female respondents as having 2x the weight, while male responses would receive 2/3rds the weight, to skew the answers to more closely match your population's actual distribution.

LAST WORD ON BIASES

This chapter has covered a small sample of the many biases out there (covering them all in any detail would require the entire book!), and ways to mitigate or avoid them.

However, the most important step you can take in avoiding bias is to recognise that it exists.

After months or years of working on your game, it can be difficult to divest yourself of the emotional attachment you have, the positive and negative feelings you possess. Just be aware that this is something that happens to everyone, and even if you do run afoul of bias from time to time, you'll have hopefully learned from it and adjust accordingly for the next data analysis or UX research study.

Now that we've covered some examples of bad data and how to avoid them, we can start building on the fundamentals for extensive data analysis

and UX research. The next chapter will start with analytics.

REFERENCES

History of the plague: https://pmc.ncbi.nlm.nih.gov/articles/PMC7513766/

YouGov data for conspiracy theories that Britons believe in: https://yougov.co.uk/politics/articles/22839-which-science-based-conspiracy-theories-do-britons

6 Introduction to analytics

DOI: 10.1201/9781003735335-6

INTRO

So far, we've discussed what data is, how to use it sporadically, and the risks associated with bad data. If you've made it all the way here, you're likely interested in working with data to aid with decision-making, throughout the development process and beyond. Perhaps you're interested in the analytics side – sales data, key performance indicators (KPIs), what players are doing in your game – or the UX research side – usability testing, focus groups, longitudinal studies, or both. Whichever you are interested in, you'll need solid processes for gathering, storing, analysing, and reporting data.

If you're interested in analytics, then the next two chapters are for you! For UX research, Chapter 8 onwards will focus on building it up from non-existent to structured maturity.

The capability to track what your players are doing in your game is a boon for decision-making and improving the player experience. Since games are built from code that defines what players can do and experience, all that is needed to track what they are doing in your game are data hooks.

Data hooks are code within a game that is used to track what players are doing, what is happening within the game world around them (such as score updates, changes in inventory, and more), and background data that players are not actively exposed to (such as your game's technical performance). They are capable of recording exactly what actions players take, and when, to the millisecond. The data that can be collected can range from 100% coverage (data from all players is returned) to a good approximation (data from a random sample of players is returned).

WHY SAMPLE YOUR IN-GAME DATA?

Given the choice of returning all data and being fully representative of what players are doing, or returning data from a sample of players for an approximation, why would we ever want to do the latter?

The answer comes down to cost and time. Simply put, the act of transferring data from your game to some form of storage has both pipeline and storage costs associated with each gigabyte of data. The more data you have, the higher your costs. Sampling means you only record a fraction of the data and lower your costs, with minimal impact to accuracy if the sampling is carried out correctly. For example, let's say your game has 10 million players. You could have the game return certain actions all players take and store them in your database, with a cost that scales with the number of players and the number of their actions you wish to track. Alternatively, you could have your game only return all actions from 10% of players, selected at random, which would result in significant savings. A simple way to sample like this would be to take the player's numerical user id and have the game only return data from players where the last digit of the id ends in 0.

Cost is only one part of the answer. The more data your game has to return, the more time it takes to send and process that data. With 10 million players' worth of data, this can result in billions of rows of data having to be processed on a daily basis.

In terms of speed, these hooks also return data with minimal time lag, ranging from seconds to minutes to hours. This enables you to track how many players are concurrently playing minutes after your game launches, or how long they played yesterday, or how many completed your game within the first week. Even after a decade of working with data, it's still exciting to launch a game and watch as clusters light up on a screen displaying a world map, showing the geographical locations of where players are playing your game, and to see the charts and graphs updating with aggregates of player actions.

But, we're skipping ahead too fast. How do we get data from the game to storage to analysis?

DATA PIPELINE

A player starts up your game, plays it for some time, makes progress, then shuts it down. You want to know when they started playing, how long they played for, what they accomplished, what they did last before stopping, when they stopped, and more. To do this, the game needs to record those player actions and transmit them if the game is connected to the internet, or store them to do so later if not.

The data can be sent to a commercial analytics platform that will do the work of collecting, transforming, storing, and aggregating the data for you, usually with a visual dashboard component to aid with analysis. Examples of commercial analytics platforms are Google Analytics, Microsoft Azure PlayFab, and Amplitude. Alternatively, you can develop your own platform to handle this. While more complicated than using an off-the-shelf platform, it does have the benefits of giving you full control, customisation, and ownership of your data and can potentially be cheaper in the long term. Finally, no matter whether your data is stored on a commercial or self-owned platform, someone (usually data analysts or scientists) can query the data to analyse it.

STARTING WITH THE BASICS

What data should the game return? What player behaviour are we interested in? The answer is of course, and usually, everything, all of it! But we need to start somewhere, and ideally start small.

A simple yet essential place to start would be the data points of when players start up your game and close it down. This would, at minimum, return the attributes of date, time, game, session id, and user id. The wrapper around these attributes is known as an event, and since these events deal with the players starting and ending a game session, they are usually known as session start and session end events.

USER IDENTIFICATION (USER ID)

A user id is generated internally by your system, either a numerical identifier or a Universally Unique Identifier (UUID) that allows you to identify your players anonymously.

A user id for data analytics should ideally not use some form of personal data, like a player's name or social security number or national insurance number. Due to the General Data Protection Regulation (GDPR) law passed in Europe to help give individuals control over their personal data, having user ids linkable to people is undesirable.

What do we get from implementing these session start and session end events? Well, from here you can derive a number of useful measures for insights and decision-making.

Daily Active Users (DAU) and Monthly Active Users (MAU) can be derived from the combination of the user id and the date attributes. These KPIs can provide you with complimentary data for your game sales figures. For example, are players playing more on the weekends? How many are still playing a month after launch? What about several months after launch? If your game still has a high number of players several months after launch, that would indicate good long-term engagement.

Player session length, and subsequently average session length, can be determined through a combination of the session id and the associated time attribute. The session id attribute here is particularly useful, as without it, you would have to manually try to associate when a player starts a game session and when they end it, which can create errors if a session start or end event is missing. For example, a player might start up the game, creating a session start event that is sent all the way to your database, but their game might crash unexpectedly before they quit the game, and the unexpected termination of the game would not normally result in a session end event being sent. Having a session id attribute helps avoid subsequent errors, as you would calculate session length using session ids that have start and end times, and exclude those that lack one or the other.

Session lengths are useful to gauge how players engage with your game; do they play it in short bursts? Do they binge for long periods of time? Do different groups of players engage with your game in differing session lengths? This information can help guide future content for your existing

game or future games. For example, if most of your players play for short sessions (say, 10–20 minutes at a time), then this may indicate a preference for bite-sized content. Future content you might develop for your game as an update should be tailored to this; tasks that can be completed within a short span of time to give your players a sense of satisfaction with every session. On the other hand, if most of your players binge your game for hours on end, any future content you make should instead be tailored to enable long, uninterrupted sessions.

Retention is a measure that tells you what percentage of players return to your game, days or weeks or months after they first play it. Retention is a good approximation for player engagement, as good retention strongly indicates that a high percentage of players are enjoying your game enough to play it again and again. Tracking retention is particularly important for Free to Play (F2P) games, as these kinds of games generate revenue by encouraging players to pay for in-game micro-transactions over a long period of time. Once, this was not as important for games that need to be purchased (premium games), since players pay money upfront. However, since the 2010s, companies have started relying more on paid downloadable content (DLC) or micro-transactions within their premium games to generate additional revenue, in exchange for providing more content to players. As such, tracking retention has become an important business metric for premium games as well as F2P games.

There are two kinds of retention, classic (bounded) and rolling (unbounded). Classic retention describes the percentage of players who return exactly X days after they first started up the game. The value of X varies, but classic retention measures are usually based on 1-day, 7-day, and 30-day retention. For example, if player A started up the game on a Friday and played it again the day after, they would contribute towards Day-1 retention. In comparison, if player B started up the game on a Monday and did not play on Tuesday, but played on Wednesday, that player would not contribute to Day-1 retention, since they played again 2 days after their first session. (With regard to retention, the 1st day a player starts up the game is considered day zero for that player.) Classic retention is not the best measure to accurately represent players returning to the game overall, but it is an excellent measure for comparison purposes and benchmarking. This is because classic retention provides a standardised snapshot of when players

return to play your game, and is stable for each cohort measured, once Day-X has passed.

Let's demonstrate using a scenario where your prior game sold reasonably well, with your launch players having a Day-1 retention of 80%, indicating that 80% of them returned to play it the next day. Now you've released a sequel that has sold just as well as the prior game, but launch players have a Day-1 retention of 50%, indicating that half of them returned to play your new game on the next day. Because you have data from both games, and because Day 1 has passed for launch players for both games, these retention values will never change, unlike rolling retention (more on that later). Now, without the prior game's data, we might think that a Day-1 retention value of 50% for the sequel is acceptable. But, with the prior game's data as a benchmark, we can see that the sequel isn't performing as well at keeping players engaged, at least not enough to have 80% of them return the next day. Perhaps the sequel isn't as good as the original, and fewer players of the new game feel the need to play it again the very next day. Perhaps the problem is technical, are players complaining about more crashes or worse performance in your sequel? Perhaps it's just as simple as the release day itself; your prior game was released on a Friday, while your sequel was released on a Tuesday. It's more convenient for players to return to play on a Saturday than on a Wednesday.

Classic retention data isn't limited to just your games. I was working on a game that was going to launch into a very competitive market, as there were multiple successful games already in that genre. The negatives of doing this are clear (red ocean versus blue ocean strategy), but a positive is that there was a lot of data out there on how existing competitors performed. We had an idea of what expected Day-1, Day-7, and Day-30 retention values were for successful games in this genre, giving us targets to aim for 1, 7, and 30 days after launch. If our game didn't perform well (which sadly happened), we could use the comparison retention figures to understand how effective our efforts to improve the game were.

RED OCEAN VS BLUE OCEAN STRATEGY

A red ocean in this context refers to a market space that is heavily contested, competing to attract customers' attention and money

towards similar products. This competition is so fierce and cutthroat that it “turns the waters bloody”, hence the term red ocean. A non-gaming example of a red ocean would be international airline companies, each offering lower prices or better perks to get people to use their services to travel to other places.

A blue ocean is one where there is little or no competition; someone or some company has entered the market space first with a new product or a new differentiator that makes their product stand out above the rest. Without the competition, the waters are untainted by the effects of competition, evoking the image of a blue ocean. A non-gaming example of a blue ocean would be Apple creating the iPhone, a phone offering that stood above the rest of the market (at the time) with its large touchscreen (lacking the need for a keyboard) and App Store (providing a huge variety of useful tools that non-touchscreen phones could not replicate).

Consider Microsoft’s Xbox 360 and Sony’s Playstation 3 in the late 2000s. Both consoles were competing for players’ attention with high-quality graphics, blockbuster AAA games, and online offerings.

For their next console, Nintendo could have created a new console with similar high technical specifications to compete with the Xbox 360 and the Playstation 3. But that would be entering a red ocean, having to fight for the same players, and having to spend money on expensive parts to ensure their new console would be comparable from a tech perspective. Instead, Nintendo created a console that was underpowered (relative to Xbox 360 and PS3) called the Wii, which used motion controls as its main gimmick, instead of fancy graphics.

Motion controls made it much easier for people who were not traditionally gamers to play video games, with approachable games such as Wii Sports allowing anyone to play tennis by swinging a game remote (called the Wii Remote) like a tennis racket. With the Wii, Nintendo was tapping into an uncontested market of non-gamers which Microsoft and Sony could not hope to reach with their traditional console offerings, essentially creating a blue ocean for themselves. Notably, Microsoft and Sony would offer their own motion control peripherals a few years later.

So classic retention is a good tool for comparing engagement, but it is not a fully accurate measure of how players engage with your game, since it only measures if players return on Day-X, not before or after. For that, we want rolling retention, which describes the percentage of players who return to play the game on Day-X or later. For example, Day-7 rolling retention describes the percentage of players who return to play the game 7 days or more after they first started. A player who first started a game on 1st January 2020 and played again on 8th January 2020 would be counted as part of Day-7 rolling retention. Another player who also first started on 1st January 2020 but did not play again until 15th February would also be counted, as that player did return to play the game again over 7 days later. In comparison, a player who started on 1st January and continued playing every single day until 5th January but never played again after that would not be considered part of the Day-7 rolling retention.

Rolling retention is good at describing the continued engagement of your players, correlating well with how fun they find your game to be. For example, if 1 year after your game launches, you measure your Day-30 rolling retention and come up with a value of 90%, that means that 90% of your players have continued playing your game at least 30 days after they first played it. Conversely, if your Day-30 rolling retention value was instead 1%, that would mean that 99% of your players stop playing your game at some point within the first 30 days. This could mean that your game has longevity problems, or it could mean that players are satisfied with their experience within that timeframe. You would check on this by examining Day-1 and Day-7 rolling retentions, to see if that 1% value changes significantly. If it does not, then the former is more likely. If it does – and to a satisfactory value – then your game is engaging for a short period of time, but not more than 30 days, meaning that certain choices may not be viable for your game, such as releasing DLC a few months after launch, since most players will no longer be playing your game.

A point of note with rolling retention is that because it is unbounded, there are some caveats to consider. For example, if you are measuring Day-7 rolling retention 10 days after your game is launched, only players who started the game within the first 3 days would qualify. In addition, the players who started playing on the 3rd day of launch would only be able to return to play on their 8th day, not their ninth or tenth, since those 2 days fall in the future of when you take your measurement. As such, rolling

retention tends to fluctuate in value when you take repeated early measurements, and stabilises over time.

As an analyst, I've used both types of retention for the purposes described above. And as a UX researcher, I've used it as a signalling mechanism, since low retention (rolling or classic) is a red flag, indicating that research might be needed into the why.

Finally, these session start and session end events allow you to measure how much time players spend in your game, simply by summing up all the session lengths each player has to get a total playtime per player. These total playtimes can be used in multiple ways. What's the average playtime of your game? Is it higher or lower than expected? If you were to group these playtimes by bands (1-hour, 10-hour, etc.), that would enable you to visualise what percentage of your players have played only a little, and what percentage have played more than expected.

These simple yet essential session events enable quite a few ways to measure how players are engaging with your game. DAU/MAU and retention in particular are KPIs for most games, and widely used for measuring the health of games and as part of KPIs.

As an example, I worked on a game that had been in development for years and had been exposed to thousands of players through a series of alpha and beta builds that they could play from their homes. During this time, we were able to use measures like DAU, average matches played, and retention to evaluate whether players were enjoying the game more, indicating that our efforts were improving the game. Not all measures had to go up; for example, if DAU remained flat, but average matches played and retention increased over time, then that would be an indication that we weren't increasing our active player base, but were giving our existing players a better experience as they were playing more matches on average and returning to play more than before. A flat DAU might mean we needed to improve our efforts to get more players to sign up to play, perhaps through improved marketing efforts.

When players were exposed to the first alpha build, the results were promising. We had a good influx of players interested in playing the game in an unfinished alpha state (so the game would have a mixture of polished and unpolished sections and would have some degree of bugs). Both the retention and average matches played KPIs were below what the game needed to achieve when it was finally completed, but acceptable for its

alpha state. It had a playable core loop, but not enough content to keep players engaged with that loop for a significant period of time.

WHAT IS A CORE LOOP?

The term ‘core loop’ refers to the essence of a game, which will vary from game to game. For example, in Tetris, the core loop is solving lines using different-shaped blocks. In Pokemon, the core loop is the act of catching pocket monsters, improving them, and duelling with other trainers. An alpha build of a Pokemon that only allows you to catch monsters but not duel, or vice versa, would not be said to have the full core loop of Pokemon gameplay.

In Tetris, the content mostly repeats the process of solving lines endlessly, with blocks dropping faster over time to increase difficulty. In Pokemon, the content is in the amount of Pokemon you can capture, develop, and evolve, exploring and unlocking parts of the world, and duelling other trainers to become the best there ever was. Limited content in an early development build of Pokemon might mean there are only a limited amount of Pokemon to capture, or a small slice of the overall map to explore, or a low amount of duels you can engage in.

Since the first alpha build of our game lacked content, it was natural that average matches played or retention would be lower than launch targets, since players would naturally grow bored with limited content. The goal was to use the data from the initial alpha builds to validate features that were working well, fix or revamp those that were not, and add content over time via successive alpha and beta builds. Correspondingly, players would enjoy the game more as we improved it over time, which would be reflected in the KPIs, especially retention.

Unfortunately, this was not to be. KPIs fluctuated, sometimes positively, but mostly negatively, such that classic retention, which was initially at half of the launch target, dropped to a quarter of that target. Player feedback was mixed, with some players loving the game, but others being lukewarm towards it, and many others showing their disinterest through actions rather than words, by dropping out and never returning. Despite significant effort

over a period of many months, we never achieved our targeted retention figure, which was an important pass/fail check for the game. We had data indicating that similar games to ours that had classic retention above the target figure might succeed, but games with classic retention below that figure always failed.

What happened to that game in the end? The company owners set a final goal of meeting the KPI targets within three months, and while the development team did their best and managed to improve the player experience such that retention returned to where it was at the start of the first alpha build, this still fell far short of target, and development on the game was cancelled.

EXPANDING THE DATA SCOPE

With the session start and end events, we have data on when players launch the game and can derive KPIs to gauge engagement. But when players are engaging with your game, what do they do? If your game has different kinds of modes, e.g. single player or multiplayer, do they engage with one or the other, or both? What aspects of your game your players engage with is important data for understanding what's working well, what needs support, and what potentially could be dropped without affecting the overall player experience negatively.

Game modes

For example, while working on a sequel to a successful game, the development team wanted to explore how the sequel could be bigger and better. One avenue of exploration was enhancing and expanding the multiplayer aspect of the game, which had only received token support previously. Their hope was that enhancements for the sequel's multiplayer mode would improve the player experience and subsequently generate positive review scores and word of mouth. There was also a separate investigation into whether enhanced multiplayer would bring in more new players, but that was a matter for market research, rather than analytics. To justify the budget such improvements would require, the team wanted to

know if improving the multiplayer aspect would improve the experience for our existing players, and how many players this might affect.

We could not answer the first part of the question, but for the latter, we had already implemented a data event that tracked what modes players were using in the prior game. This event had an attribute describing whether players were playing the single player or multiplayer component of the game, and this, along with the user id, date, and time attributes, meant we could:

- Calculate the number of unique players who have ever engaged with multiplayer.
- Measure how much time these players spend in the multiplayer mode.
- Understand if players repeatedly engage with the multiplayer mode over a period of days or months or years, or if it's just a one-off.

What we learned was that only a tiny percentage of players had ever started up the multiplayer mode in the game. And of that tiny percentage, most only played a single multiplayer match. A fraction of a fraction of players engaged with the multiplayer mode regularly, with the vast majority never trying it or only trying it once.

Returning back to the development team's question: would improving the multiplayer aspect improve the experience for our existing players, and how many? Our data could not prove or disprove any enhanced experience for existing players, but it could indicate that any effect would be minimal. At best, a tiny minority of players would have a better experience with the enhanced multiplayer. While it's always worthwhile to improve the player experience even for a small number of players, there was the opportunity cost to consider; there were multiple other avenues the development team could take to improve the sequel, and there was limited time and budget available.

We did not refute or confirm the validity of improving the sequel's multiplayer mode as an avenue for making the sequel more successful; there was insufficient data to do so. We could only provide a small piece of the answer with the data we had, which will happen more often than not, as it's rare that data will be able to provide all the answers needed. As mentioned earlier, there was the possibility that improving the multiplayer

mode could attract new players, and even convince previously disinterested existing players to engage.

To be able to gauge if this upside was feasible, the multiplayer improvements would have to not just cater to existing engaged players but also fix issues that turned away existing disengaged players in the first place. What improvements were the team considering? Did they know what the players wanted out of the multiplayer mode, what turned them away from it? Did they know what was unappealing about it such that the vast majority of players never even tried it? Was it perhaps a discovery issue, did players not know about the multiplayer mode?

Unfortunately, these were questions that the company was not well equipped to answer at the time. We had feedback from players via community forums and social media, but no research capacity to answer questions like this. This lack of data and the low engagement with the multiplayer mode led to that aspect of the sequel being kept as it was, with minimal budget allocated. Analytics data can tell you many things, but not necessarily what players want, what they don't want, or how they will react to brand-new changes.

Settings

Video games also have settings that players can change for their own needs. These settings span a wide variety, such as prioritising better graphics or better performance (how smoothly the game runs on your device), changing the language, enabling or disabling subtitles, the game's difficulty, the control schemes and keybindings, camera movement sensitivity, and more, all in the name of empowering players to customise their experience as they see fit. Tracking what settings players change, and what settings they keep, can be very useful for decision-making.

Quite some time ago, our games supported both stereo and mono options for audio settings. Because every feature and setting in a game has to work correctly and to the players' expectations, this meant that even maintaining old settings had a cost in the form of QA checks and code updates to align with updates to the game engine.

The development team were considering ways to lower costs by removing features that players no longer used and asked if it was still worth

supporting the mono audio setting, as the sound design team felt that it was unlikely any players were still using it. We checked the data across multiple games, and their assumption was correct; less and less players were selecting the mono audio setting with each subsequent game, such that less than 0.03% of players were using it in our latest game. This gave the team the confidence to go ahead and remove the setting entirely, secure in the knowledge that only the tiniest fraction of our players would be affected, and they would likely switch to stereo or surround sound options, as that was the trend across our games.

Data around what settings players use can help with more than optimising the budget; it can also help to optimise the player experience in small ways.

One of our game franchises had two different control schemes, one legacy, the other brand new, recently implemented based on what competitor titles were using. The goal was to give players more choice and let them play the game however they wished; they could use the legacy controls they were familiar with from previous games, or switch to the new competitor control scheme. To gauge how successful this initiative was, we tracked the adoption rate of the new control scheme. From the data, we could see that most players stuck with the legacy controls, at least initially. We kept tracking this over several years and several sequels, and over time, the balance shifted with each subsequent game. Eventually, the competitor control scheme was the one that the majority of players used, and one of the first things players would do in the latest sequel was to go to the options menu and switch to the newly preferred control scheme. Based on the trend and this finding, we made a recommendation to set the competitor control scheme as the default, since that was what the majority of players preferred. The minority could of course change it back to the legacy controls; we wanted to continue to give them freedom of choice in how they played, but minimise the effort needed by a majority of players to do so.

Transaction events and data

We haven't discussed implementing data for sales of premium games, simply because it's unlikely you'll ever have to. Whether your game is sold in physical retail stores or on an online platform, the data infrastructure work for tracking sales has likely been carried out by the platform holder.

Retail stores will send you or your publisher the number of copies your game has sold and the revenue generated, as will online platform holders like Steam (Valve), Playstation (Sony), and Xbox (Microsoft).

However, as video games are trending more and more towards including paid DLC and micro-transactions, we should discuss data hooks for transactions within the game. Much like many of the events discussed before, a transaction event will include the attributes of user id, date and time, session id, and so on, to identify the who and the when. Other attributes would include the amount paid, the currency used, the item purchased, as well as the quantity and discount received, if applicable.

For free-to-play (F2P) games where the revenue is generated entirely by micro-transactions within the game, transaction events will contribute to the most important KPIs for your game, in addition to MAU/DAU and retention.

Data from the transaction events can tell you how many players buy anything in your game, no matter how large or small, providing you with an important KPI conversion. Conversion describes the percentage of players that convert from being solely F2P players to paying players, indicating how effective your game is at encouraging players to spend. More importantly, it lets you track the effectiveness of your efforts to get more players to spend, whether that be by improving the player experience and/or the attractiveness of your offerings.

A DIFFERENT KPI DEFINITION FOR CONVERSION

Conversion can also describe a different KPI. In marketing, conversion refers to when an advert shown to a person leads to that person engaging with the goal of the advert, whether that be making a purchase, downloading an app, or signing up for a service.

For advertising in the context of games, a conversion is usually when an ad campaign successfully gets prospective players to purchase or download your game.

Conversion alone doesn’t give us a full picture of player spending habits. F2P games tend to have a wide variety of items for players to purchase,

with some items costing less than a dollar, while others might cost well over a hundred dollars. Knowing how much players have spent on average can be expressed as Average Revenue per User (ARPU), which is the amount of revenue spent divided by the number of players who have played your game in a given timeframe. If that timeframe is set to a per-day basis, then ARPU becomes Average Revenue per Daily Active User, or ARPDAU, which is a useful measure to track players' daily spending behaviour. This can be used to measure changes; for example, you may be able to correlate updates you make to your game with a change in the ARPDAU.

If the timeframe used is not daily, but is instead the entire lifetime of your game, then ARPU is also known as Lifetime Value (LTV) of your players, since a simple way to calculate LTV is your total revenue divided by total number of players.

LTV is useful as a tool for cost-benefit analysis. Let's say you spent $1,000,000 on an advertising campaign for your game that brings in 10,000 players. Was that ad campaign successful? Well, if your data for your game tells you that the expected LTV of an average player is $10, then it was not a success, as you spent $1M to eventually get $100K. On the other hand, if the LTV is $200, then the ad campaign was a success, as you spent $1M to gain $2M, eventually. LTV allows you to judge not just how successful your advertising campaigns were, but also evaluate how profitable any prospective ad campaigns might be, depending on how many players they are estimated to bring in.

Another useful measure is Average Revenue per Paying User (ARPPU), which can be derived by dividing revenue using only the paying players. This will naturally result in a higher value than ARPU, since all the non-paying players are removed. ARPPU is useful because it is more sensitive than ARPU to changes in your monetisation strategy, especially if you have significantly more non-spenders than spenders, which in my experience is usually the case with F2P games. For example, if the user base for your F2P game is 99% non-spenders and 1% spenders, an update to your game that affects revenue might cause a noticeable increase in ARPPU, but not in ARPU, as all the non-spenders would be unaffected.

Knowing which players spend and which don't can be helpful as it enables you to monitor if there are any differences in how each of these segments engages with your game. Do the spenders spend because they are having a good gaming experience? Or are they spending in order to get a

good experience? If it's the former, does this mean the non-spenders are having a bad experience?

You can also expand on this segmentation between spenders and non-spenders by segmenting the spenders themselves. In any F2P game, there will be those who spend a little, those who spend a moderate amount, and those who spend more than all the rest combined. The Pareto principle, which states that 80% of effects originated from 20% of causes, generally applies to F2P games, where the top spenders can represent 2% of the total players, but generate 90% of the game's revenue.

Segmenting the spenders helps you identify if you have players who spend far beyond the rest, potentially tens of thousands of dollars on a constant basis. If revenue is the main concern – and there is a balance to be had between focusing on revenue and focusing on the player experience – then catering to these extremely high spenders who provide the majority of revenue can be an efficient way of generating more.

CONSIDER PLAYERS WHO CANNOT AFFORD TO SPEND TOO MUCH IN YOUR GAMES

If you do have extremely high spenders who pay tens of thousands of dollars into your game, the hope is that they're wealthy and can afford such a luxury without any impact to their finances.

Having non-wealthy players who spend most or all of their disposable income into your game is not desirable, from ethical and longevity standpoints, and something to strongly consider.

We've discussed adding new events to expand the scope of your data, but another way to do so is to expand on the events you already have. The session start and end events tell you when players start up your game, how long their sessions are, and when they stop playing altogether. As established earlier, we can do a lot with that data, but there are more useful attributes we can add.

For example, if your game is multi-platform (PC, Playstation, Xbox, Switch, Android, iOS, etc.), adding a platform attribute to the session start event can give you an idea of what platform players are using to engage

with your game. If any of the platforms your game supports has barely any players, that might inform whether you drop support for future titles, particularly if the cost of supporting that platform through coding and QA is more than the revenue brought in. If the time component is added to the analysis, you can also understand if certain platforms are becoming more or less popular with regards to your game, and alter your strategy accordingly.

In a similar vein, adding the country attribute to your session start events can tell you where your players are. A player's country can be determined using the IP address (which would be required in order for the game to send the data in the first place). While this may not be a perfect estimate due to factors like virtual private network (VPN) masking players' IP addresses, or players relocating to another country, it's a good enough estimate. It's best to return the approximate country rather than the IP address, as the latter is considered personal data under GDPR.

Knowing what countries your players are from has similar benefits as knowing about the platform they play on; it helps you identify emerging markets and tailor marketing efforts accordingly. It also informs you on matters relating to localisation.

LOCALISATION

Localisation in video games is the process of ensuring that the game makes sense for players in specific countries or regions. The bare minimum is to translate the text, which requires the least time and cost. Translating the audio is far more costly, as it requires new voice actors to record dialogue in other languages.

Localisation isn't just limited to translating text and audio from one language to another, but also ensuring that cultural differences are translated over correctly. For example, 13 is considered an unlucky number in Western culture and is often used in pop culture to denote misfortune, or set an ominous atmosphere. However in China, 13 is instead considered a lucky number as the pronunciation can mean "assured growth". Instead, the number 4 is considered unlucky in China, as the pronunciation for 4 can also mean "death". As such, if a game originating from a Western studio used the number 13 to set an ominous atmosphere, that number might be swapped for 4 during

localisation if the game is released in China, to ensure players get the intended experience.

As you can see from this small example, not only does good localisation require experienced translators and voice actors to convert text and audio, respectively, a great deal of consideration has to go into translating cultural differences as well.

Because good localisation requires time and money, understanding which countries your players are from can help with decision-making. Some of the games I've supported have had widespread text localisation for multiple languages, while more strategic decisions were made on audio localisation. A good deal of this strategy was informed by understanding where the majority of our players were from. For example, using session data, we would count the number of distinct players our game had, grouped by country. Using this list, we could see the top five countries with the most players, and any of those countries that didn't have English as their primary language usually had full audio and text localisation implemented, with the remaining non-English countries in the top ten having text localisation support. This enabled us to use our localisation budget wisely; it makes more sense to localise your game for a country where 10 million people are playing your game, compared to a country where only ten thousand are playing it.

We covered tracking different modes (single player and multiplayer) earlier in the chapter via adding a new data event. As our game also had difficulty settings, allowing players to tailor their experience to their desired level of challenge, a useful way to expand this event was to add an attribute tracking the difficulty setting players selected. Using this, we could determine the proportions of our players that chose easy, standard, and hard settings, and further determine what the success rates were for these proportions. We learned some interesting lessons here; one might expect that the first-time players who selected the easy setting would have higher success rates for completing the game compared to the other difficulties. In reality, it was the opposite, first-time players who selected the hard difficulty setting had the highest success rates, followed by those on standard difficulty. Digging deeper, we discovered that most of the players on the easy setting either stopped playing the game partway through or

restarted a new playthrough on a more difficult setting. As we did not have UX research resources available at the time, we did not have the why behind these actions, but could hypothesise.

We looked at other factors, such as how long the easy difficulty players played for before they quit, and compared this against the data for the standard and hard difficulty players who also quit. A much higher proportion of the easy difficulty players quit very early on, within the first 2 hours of play, which happened to be the time limit that Steam, the platform our game was on, had for auto-refunds. Any players who requested a refund with less than 2 hours of play could get one through an automated system. So, of the easy, difficult players who quit, most of them did so within the first 2 hours, likely to obtain a refund, potentially because the game experience wasn't a good fit for them. In comparison, the majority of the standard and hard difficulty players who quit only did so after roughly 10 and 20 hours, respectively, potentially indicating that they were enjoying the game for some time, especially the hard difficulty players. Finally, we looked at the data of the easy difficulty players who switched to a harder difficulty, and found that they played for significantly longer or went on to complete the game, potentially indicating that while they enjoyed the game, the easy setting made it too trivial to keep their interest.

The development team took several actions based on this data. They added a simple "recommended" text to the standard difficulty, providing guidance to brand-new players on which difficulty would be best. They also enabled players to change the difficulty without having to start a new playthrough (to avoid players wasting their time) and set a text pop-up in the 1st hour of gameplay to inform players of this functionality. As some players were still completing the game on easy difficulty, they decided against increasing the difficulty; it was deemed best to keep the easy setting at a trivial challenge level for players who wanted or needed that experience. We monitored the data after these changes were implemented, and while it had minimal effect on the first-time standard and hard difficulty players, the first-time players who selected easy difficulty had lower quit rates, with more of them shifting over to other difficulties instead.

Once you've made headway into using analytics data, even on a sporadic basis, this may spur more interest from the development teams to collaborate on adding more data hooks to get even more data. In quite a few cases, members of the team had approached us to understand how they can

best implement data hooks for useful data. This is extremely beneficial for everyone involved – as a data analyst, it can be difficult to fully understand every aspect of the game in as much depth as designers do, and programmers who actually code in the data hooks will know the limits of what is possible to track, and what is not. For the game developers, they benefit from being able to take advantage of an existing data pipeline and your expertise on best data practices. Overall, this collaboration tends to result in a better, more standardised way of integrating analytics into games.

A nice bonus of collaborating with other members of the development team is that they'll start to wonder about the data and make detailed data requests, some of which we have covered above. Such a request came on for a game I was supporting. During the early days of launch, a majority of players stopped playing after about 10 minutes, on a game that was intended to have dozens of hours of content. The team needed to find out why, and to give them an answer, we turned to descriptive analytics.

7 Analytics methodology examples

DOI: 10.1201/9781003735335-7

METHODOLOGY EXAMPLES

ONBOARDING

Previous examples have informally covered descriptive analytics, which is the interpretation of your historical data to understand what has happened, and to identify patterns and trends in your players' behaviour.

Depending on the scope of your game, the length of your onboarding or tutorial can vary from as little as a few minutes or as long as a few hours. Some game teams choose to make the tutorial optional, so players can ignore it if they feel they don't need it. Others make it mandatory, with some going so far as to have the tutorial be the first thing players experience, to ensure that they are prepared for the challenges ahead.

Our game had a mandatory tutorial for first-time players, which lasted for about 15 minutes. Because of this, the development team had already suspected that the majority of players who dropped out during the first 10 minutes were doing so because of the tutorial. During this 15-minute tutorial, players had to carry out around 30 actions, with instructions and guidance throughout. With each action taken, the game would record it and eventually send it out all the way to our database. The data attributes included, among other things, the user id, when they carried out the action, the name of the action, and the numerical step of the action out of a progression of 1–30. Because this data hook was already in the game, we could use these attributes to query the database, visualise players'

progression through every step, and identify exactly where they were dropping out within the tutorial.

COULD WE NOT JUST ESTIMATE?

We have the exact times when players stop playing in the tutorial. So why go to all the trouble of coding in data hooks for the game to record the tutorial steps, returning them, storing them (all extra cost), and writing queries to process the structured data so that it can be visualised? Why not just use the exact times players stop to estimate the approximate action in the tutorial?

Simply put, players play at different speeds. Some may rush through the tutorial, some may take their time reading the text and taking in the details, and most fall somewhere in-between. Since players vary so much in their behaviour, estimating which point in the tutorial where they stopped based on time played is unlikely to yield accurate or useful results.

To visualise the results, I plotted each action sequentially in a bar chart, with each bar representing the percentage of players who completed each action or step, out of all players who had started the game. For example, the bar representing the first action in the tutorial had a value of around 99.7%, indicating the percentage of players who had completed that first step. While it may seem strange that 0.3% of players start up the game and don't take even one action in the tutorial, this is actually fairly normal. Some players might start up the game to make sure everything is working, then shut it down with the intention to play again in the future, but never do. Some players go even further, buying games that they end up never playing for a variety of reasons. On Steam, around 33% of players' games go completely unplayed, meaning they never even start up those games.

The resulting bar chart had 30 bars representing each successful action players took in the tutorial, and depending on player behaviour and the corresponding data, could have shown us multiple different patterns. Had we seen a constant 2% decline with each subsequent bar, this would have meant around 58% of players drop out before the end of the tutorial (29

bars times 2%, since we exclude the first step). It would have also meant that the problem wasn't any one step, but the tutorial (and potentially the game) as a whole.

This wasn't the case, as there were some very slight declines in most steps, but not enough to be a problem overall. The problem came from a 40% drop between steps 23 and 24, indicating that 40% of players decided to stop playing the game before – or because of – the action required in step 24 of the tutorial. The bad news is that while we knew exactly where the problem was, we didn't know exactly why. That step required players to complete a battle, which itself had multiple sub-steps or sub-actions players needed to carry out to complete the battle. Unfortunately, no code had been added to track and report on these sub-actions within the tutorial battle. As such, we still had no idea what the problem was. Was it some step or mix of steps within the tutorial battle that was causing players to quit? Was it the battle itself? Perhaps players didn't like it when compared to the rest of the tutorial, as it was quite different from what came before. We had a lot of theories on what the problem could be, as well as possible solutions, but each solution required some significant work to implement and differed in result. If the problem was that players didn't like the battle, improving the steps within the battle would not help. If the problem was that some actions in the battle were too difficult to carry out or understand, revamping the battle to improve the player experience would not help.

Compounding this lack of exact information was budget, or rather, the lack of it. There was only minimal development time available to make tweaks to the game and apply a single patch; after which support for the game would halt, as the team were meant to move on to a new project. Hence why some solutions were not possible; for example, we did not have the time to code in hooks for the tutorial battle and then wait even longer for new players to experience the tutorial so we could collect their data. Or A/B testing with two different solutions to experiment and see which solution resonates more with players. (If A/B testing is a new term to you, more on that shortly later in this chapter.)

Why not run a quick UX research study to ask players directly? Aside from the aforementioned budget issues, the UX research team did not exist yet at the time for the company I was in. This was one of the many times I wished I had the why as well as the what, but not the last, as later examples will show.

This lack of certainty on what we could do combined with lack of time and budget meant that no solution could be agreed on, and with the patch deadline fast approaching, ultimately no fix was implemented for this problem, and the game continued to lose a large portion of new players during the tutorial for the rest of its lifetime, until it was ultimately cancelled.

SOMETIMES THERE WILL BE NO CLEAR AND EASY FIX, OR SOMETIMES IT WILL BE TOO LATE

This was an example where we had detailed data, but not enough to make a difference, and as a result could not make an informed decision to improve the situation. Time and budget constraints further exacerbated the problem.

I'd love to state that all you need is more data, and the above will never be a problem, but the reality is that there will be scenarios where you don't have enough data, or there is no clear solution, no matter how much data you have. The best you can do is ensure that you learn from it, and try to do better next time by having suitable data hooks in place, or discover the flaws earlier in development.

Balancing

Another game I worked on had launched to positive reception and commercial success. However, there were complaints from players through social media that a particular mode of the game was unbalanced, with certain units being far more powerful than others. As this mode was a competitive multiplayer mode, with the intention for players to be able to play in matches against each other, ensuring good balance was important. Good balance for a competitive game means that there are multiple viable strategies a player can use to achieve victory, with no one strategy being significantly better than all of the others. Poor balance means that there are only one or two dominant ways to play, with all other ways leading to certain defeat. An unbalanced competitive multiplayer game means that the

matches become stagnant, as players chasing after victory will devolve into only using one of a few dominant strategies.

Upon learning about this, the development team immediately asked us (the analytics team) if the claims were valid. This game returned a lot of data about what players were doing, including their engagement with both the single player and multiplayer modes, whether they won or lost their matches, against whom, what units they used in matches, and more. As such, I had a lot of data to dig through and investigate thoroughly.

The first potential indicator that the multiplayer mode might have poor balance was at a macro level. The number of players engaging with the mode was declining over time, far more drastically than those engaging with the single-player mode. This was only a potential indicator, since an alternative reason for this could be that the multiplayer mode had less longevity or was simply not as fun as the single-player mode, regardless of balance.

The next step was to investigate the claims at a micro level. If the mode was as unbalanced as reported, it would be indicated by certain strategies having an abnormally high win rate. In this game, players created an army by selecting a combination of units that they could take into a match to battle against other players' armies, and these units could not be changed once in the battle. Since the game collected data on what units players brought into battle, I could list armies (combinations rather than permutations, since the order of units does not matter) alongside their win rates and frequency of battles in which the army was used. That last measure, frequency, matters because for this to be reported as unbalanced, the strategy must not just have an abnormally high win rate, but also be used frequently. A strategy that has a 100% win rate but has only ever been used a few times would not be an indicator, since it would not be deployed against enough players on the losing side to generate a sufficient number of complaints. Furthermore, there is no guarantee a low-frequency strategy is even a winning strategy, given the sample size. Because of this, I grouped together a full list of all armies used by players in the weeks since launch, used a filter to exclude armies that had low usage rates, and sorted the filtered list results by win rate.

BALANCING BASED ON WIN RATE

Perfect balance in a video game is impossible to achieve, nor is it desirable. By perfect balance, I mean something like flipping a coin and guessing if the outcome is heads or tails. No matter which side you call, the outcome is always a 50/50, perfectly balanced. But not very sustainable as an exciting and engaging game.

There's no skill involved with flipping a coin, which is not true of competitive video games, or any competitive game, for that matter. In the game I was supporting, selecting units for your army is just the beginning; players still need to be able to direct and utilise those units in battle against the opposing player's army. A player's natural talent and experience affects how that army performs in battle, such that an army in one player's control might have a 55% win rate, while the same army in another's might have only 40%.

The goal with balancing is not to achieve a perfect 50% win rate, but to aspire towards it. If individual units and armies' win rates in a game were consistently between 45% and 55%, that would be a well-balanced game. Some armies would have a very slight advantage, but not enough to be a concern. Even a 40–60% range, while not ideal, would still be acceptable and something to improve over time without the need for drastic action.

In the resulting list, most armies fell into a 40–60% win rate range, but several armies stood out, each having a win rate of over 80%, used in tens of thousands of matches. There was no correspondingly low win rate army with an under-20% win rate, which is understandable, as players don't reuse unit combinations that perform badly, and thus these combinations are likely to be excluded by the low usage filter. It was clear that these high-win-rate armies were the likely culprit for player complaints around the unbalanced multiplayer mode.

I investigated further, approaching it from two angles. First, while these armies had an over 80% win rate, they were still losing nearly 20% of the time. What were they losing to? Because we had so many data hooks for multiple aspects of what players were doing in the game, and what outcomes they experienced, I could query the database and create a list of all matches where players controlling these armies had been on the losing side. From there, I could find out what armies were on the winning side,

and it turned out to be the same overly powerful armies. It turned out that only an overpowered army could defeat another.

The other angle was to understand why these armies were performing so well. Was it a perfect combination of units that complemented each other? Did the unit combination act as a perfect counter for every other army? Was there a common factor between the armies?

It turned out that these armies all did have a common factor, a unit that was present in all the winning combinations. I reran my data query to return the list sorted by win rate, but this time focused on individual units rather than armies, and this unit was at the very top of the list, with a similar high win rate.

This unit had the unique ability to deal high amounts of damage from afar, whilst on the move, and could be upgraded to also heal itself whilst dealing damage. This alone was not the problem, as multiple units in the game possessed unique abilities to make them interesting to play with. This unit's unique ability was compounded by the format of the multiplayer game mode's maps, which were control point maps. The objective of a control point map is for a player to have their units occupy control points, which would grant points to the player over time. Normally, these maps have an odd number of control points (some games could have 3, 5, or 7, ours only had 3), and thus players who control a majority of control points with their units for a sufficient amount of time would win by having the most points. The maps were designed to be fairly small, ensuring that frequent combat encounters would occur, as players' units would clash often on the battlefield. This also meant that the control points were fairly close to each other, and units that could move quickly between points were valued highly, as were those that could strike at one control point from far away whilst defending another point. The overpowered unit excelled in both of these aspects, and any army with multiple copies of this unit could easily hold and defend two control points simultaneously, thus guaranteeing victory against players who had to split their armies to achieve the same effect.

I showcased this data to the team, who agreed with the findings and downgraded the capabilities of the unit in question, quite severely lowering the range at which it could deal damage. The complaints stopped almost immediately after changes were made, and army win rates settled in a comfortable range of 40–60%.

Heatmap for unit deaths

Not all problems have immediate answers. Sometimes you'll encounter multiple dead ends when sifting through the data, looking for the patterns that would explain what's causing the problem, especially when you have an ocean of data available.

DASHBOARDS

Dashboards are visualisations that show multiple related KPIs in one place, much like the instrument panel of a car. Dashboards are usually – but don't have to be – centred around a common section of the game. For example, you might have a business-level dashboard that displays the KPIs most related to a game's overall performance: DAU, MAU, and retention. For a premium game, this would also include total unit sales, daily/monthly sales in a line chart, and similar counterparts for revenue. For a F2P game, sales data would be swapped for new installs or downloads, ARPU/ARRPU, and conversion.

Dashboards allow anyone, not just data analysts or scientists, to easily view and understand important information pertaining to the section of the game they are interested in. The business dashboards described above would be of interest to most developers, especially so for business executives who need to gauge the financial health of the game at a glance, to understand if the company is on track to meet targets.

Dashboards can be static snapshots taken at a regular cadence, but are more likely to pull data real time from databases onto a suitable platform, such as Microsoft's Power BI or Google's Looker. This is done by writing structured query language (SQL) code (or similar query code, such as Kusto) that exports the tabular data from a database to the platform in a suitable format (usually charts) that allows the platform to visualise the data. These live dashboards can be refreshed at will by viewers and will usually have filters available to enable better self-service. Date filters, for example, enable a viewer to select only the date period he or she or they are interested in, rather than looking at the lifetime data of the game. Other filters, such as

country, platform, spender segments, and more, allow for similar investigation of what countries/platforms/segments are over- or under-performing.

Dashboards aren't limited to business affairs; visual summaries of how players interact with various aspects of the game are a powerful tool for enabling self-service for the development team. Some of the data requests described earlier in this section, tutorial progression and army balance, eventually resulted in dashboards being created for those sections, so developers could easily check on the status of player progression through the tutorial/onboarding, or keep an eye on the win rates of armies. These dashboards served to provide not just at-a-glance information, but also to potentially enable developers to get answers swiftly, especially for questions that are common and likely to be asked on a regular basis across multiple games.

The developers in charge of designing battlefield maps for one of our games had noticed an unexplained anomaly in the map dashboard and wanted to understand what was causing it.

The map dashboard tracked multiplayer map statistics on a per-map basis, including usage frequency, win rates for both sides, and average match length. As the maps for multiplayer matches were selected by the game at random, the frequency served as a check that the randomiser was working as intended. If all multiplayer maps had roughly the same usage frequency, all was well. If a map had much higher or lower usage, then something was wrong, either with the randomiser or another part of the game. Similarly, players were allocated to one side or another at random, so the win rate for each side of each map should be around 50%, give or take a few percentage points. Any discrepancy would indicate that a particular map was imbalanced, meaning that random chance might give players an advantage just for being allocated to one side of a map over another. The final statistic, average match length, gave us an idea of how long matches lasted on any given map on average, providing a number that helped validate design intent (e.g. larger maps should result in longer matches).

This last statistic had the designers concerned, as one of the maps had an abnormally short average match length, short enough that it would be barely feasible for players to complete a match in that time. I queried the

data to create a visualisation, bucketing the frequency of matches on that map by 5-minute time buckets, to see what was skewing the average match time downward. I chose 5-minute buckets as matches on average lasted for less than an hour, so this bucket size avoided over- or under-segmenting the data, ensuring that the resulting visualisation would not have an obscured trend due to too many (over-segmenting) or too few (under-segmenting) bars in a bar chart. This revealed that there were a significant number of matches in the lowest time buckets, 0–4 and 5–9 minutes, abnormally so, with the remaining matches distributed across other time buckets in a similar pattern as other maps.

So, we knew what was causing this – multiple matches on this map were ending far earlier than expected, across two-time buckets. But why? I had a discussion with the designers to see if there was anything unique about that map that players might not like, or something that it lacked compared to the other maps, but nothing stood out from a gameplay perspective. While the average win rate looked acceptable for both sides, that included the wins and losses of players quitting the map early, causing the opposing player to win for free. Perhaps there was a problem with the map's balance, causing players to quit out rather than play it. I reran the win rate query for that map, excluding the matches that were shorter than 10 minutes, but there was nothing abnormal about the resulting win rate for either of the map's starting positions.

Using the session ids and match ids of players who left the matches early, I investigated if the sessions themselves had no session end event or if the matches had no match end event. In a two-player match, if one player's game crashed or terminated early unexpectedly, the match would end with the other player winning, and that player's data regarding the match would be sent to us. However, for the player who crashed out, their game would not send us a session end event or a match end event. I suspected that perhaps the losing players were not choosing to quit out early, but were being kicked out due to the game crashing for them on that particular map. Unfortunately, this also ended up not being the source of the problem.

Multiplayer matches in our game were peer-to-peer, meaning that rather than a server hosting the match and players connecting to said server, one of the players was selected at random to host, with the other player connecting and the match playing out that way. Rather than crashing out, perhaps

players were being forcibly disconnected from the matches due to something unique to that map. We did not track unintended disconnects, as disconnecting and quitting were reported in the same way, so I asked QA to investigate if they could reproduce the potential cause, as they could automate multiple computers joining matches specifically on that map. They did not discover a disconnect issue and went one step further, replaying the map over and over to determine if some other technical issue was the problem, but found nothing.

So, we had ruled out players quitting unintentionally, either through crashes or disconnects. Players were choosing to leave that map and incur a loss, sometime within the first 10 minutes. We did not have much tracking within the match itself, as there would be too many player actions and commands ordered by players, with no meaningful way to visualise said actions. One data event we had implemented recently reported back on where army units died on a map, with attributes for the unit id, player id, match id, map id, date and time, and the x- and y-coordinates.

As it was a newly implemented event, we had the raw data for it, but no heatmaps visualised. I took this chance to do so for the offending map, as well as two other maps for comparison purposes, using a greyscale image of it as the background, and plotted a series of coloured dots using the x- and y-coordinates. Each dot was given a colour (green, yellow, red) with each colour denoting low, medium, or high density of unit deaths in that particular location. A cluster of red dots would indicate a high amount of unit deaths in a location, indicating that multiple unit skirmishes had occurred there over multiple matches on that map. The expectation for most maps is that there would be multiple clusters of red dots encircled by yellow and green ones, denoting common points of conflict over locations of strategic interest to players. These clusters would be connected by “roads” of green and yellow dots, denoting where skirmishes had occurred less frequently in the spaces between the strategic locations. The comparison maps’ heatmaps, when plotted, looked exactly as described above.

For the offending map, its heatmap stood out immediately as being different. There were only two extremely dense red clusters in two corners of the map, with the occasional light green cluster elsewhere. The map was symmetrical, and the red clusters were effectively mirrored. I investigated the offending map in-game and found that the red clusters were positioned at the base of large hills, with only one path up. After discussing with the

developers, we came to the conclusion that because the map was a deathmatch variant (a player could only win by wiping out the other player's army), it was likely that players were ordering their units to the top of the hills and settling them into a fortified position. If one player approached the other's fortified position, they would be wiped out easily, ensuring victory for the fortified player. If both players sat on their respective hills, the match would simply go on with no victor. The developers had only intended for the hills to be used as background scenery, and not optimal fortified locations.

PLAYERS WILL ALWAYS OPTIMISE THE FUN OUT OF A GAME

At their heart, video games are about problem solving. How to win a battle. How to gather resources. How to build up a civilisation. Part of the fun of games is figuring out how to do these tasks efficiently, optimising as much as you can. Sometimes players will do this even if the most optimal action isn't necessarily "fun".

For example, Destiny, which was a looter shooter – a game where the main goal is to progress and kill stronger enemies, who will drop better loot or equipment, which you can then use to kill even stronger enemies, in an ever-escalating loop – had a cave where enemies would spawn endlessly if players stood a certain distance away from the entrance. Once in the correct location, players could shoot at endless enemies running out of the cave in a straight line towards the player, akin to shooting fish in a barrel. It was an extremely efficient and carefree way to get loot, with players doing this for hours despite the monotony.

We hypothesised that players, upon discovering (or losing to) these fortified locations that guaranteed a win, would always default to going there, even if it meant stalemating the match and resulting in players simply quitting out of the map, once one or both players were entrenched. Since this was not the intended way to play this map, nor was the geography meant to be used in this way, the developers decided to remove the ability to access the

hills in an upcoming patch. This was accompanied by a visual change to indicate that the path up the hills was blocked and included patch notes that inform players of this change, to avoid any confusion.

We monitored the status of that map for a few weeks after the patch, and the average match length crept up slowly, which was a positive sign. When a filter was applied to limit matches played after the patch date, the average match length on that map was roughly in-line with the other maps' match lengths, and while there were still players who quit the offending map in the first few minutes, there were less and less of these as the weeks went by. The average match length for that map had crept up slowly, because it was an average across the lifetime of the game, so the earlier matches pre-patch were still skewing the match length far below the others. Overall, it looked like we had found the problem and fixed it.

Why did fixing this map matter? After all, our game had many maps, and one map having this problem out of many other maps would only be a minor friction point for those players who did experience it. Not a big deal if their overall experience is positive, right? The reality is that any player experience is going to have some friction, no matter how well designed your game is. If successful, it'll be played by millions of players, all with different preferences and reactions to your game. They will experience friction in your game, some of it intentional (provide a challenge for players to overcome to gain a sense of satisfaction), and some of it unintentional, like the map problem above. Removing as much unintentional friction as possible helps ensure the player experience is as good as it can be.

We had solved the problem, even though it took a while and a lot of dead ends during the investigation. However, not all problems need to be solved, as we'll see with the next example.

The why behind requests needs to be considered

Over time, we rolled out dashboards for heatmap deaths for all the maps in our games, which were received well by the developers, who found them useful for self-service investigations. In addition to the example described earlier, they gave the developers a good sense of whether chokepoints in

maps were working as intended (units should be dying there) and whether players were fighting over strategic points of interest.

For one of our upcoming games, a designer wanted as much analytics support as possible. In addition to the unit deaths heatmaps, he also wanted to add data events that would track the players' camera movements. This game had an isometric view, allowing players to view the battlefield top down from an angle. Players could move the camera across the battlefield, and also rotate and zoom. The designer was particularly interested in a heatmap that would aggregate where players rotated their cameras on each map.

Heatmaps are a useful tool for deaths on a map, as they display where deaths occur and quantify them through colour coding and intensity, visualising high and low conflict zones. Heatmaps can also be useful for visualising eye tracking data on websites, enabling you to see where viewers focus their attention the most and the least. This provides you with an indication of which elements of your website draw attention, and which don't.

However, the benefits of a heatmap that showed where players rotated their camera on the map eluded us. We (the analytics team) discussed it, but could not divine a use case for having such data. We had a sit down with the designer to better understand his intentions; perhaps he had thought of a unique solution to a problem he was facing.

The designer was concerned about the amount of environmental effects (such as flames and smoke) and terrain features (such as towers and hills) on the map. These effects and features provided points of interest and added a sense of immersion, but also took up space, particularly vertical space, meaning that at times they would obscure units. This in turn might impede players' view of army units on the battlefield and their ability to select them and give them orders in the heat of battle.

The designer wanted to have a heatmap for camera rotations, as his reasoning was that areas on the map with high intensity of camera rotations that were correlated with the locations of visual obstacles would allow him to make the case for removing said obstacles, to remove friction from the player experience.

While his intent was good, the solution would not have worked. When a unit dies on the map, the location of its death is a defined event, i.e. a unit died there. When a person gazes at an element of your website, that is also a

defined event, i.e. the gaze data tells you what they were looking at. But when a player rotates their camera, it is impossible to determine if they were doing so to look around an obstacle or simply rotating the camera to look around the battlefield. The camera rotation data, even with x-, y-, and z-coordinates plus the camera's viewing angle, could not tell us if there was a visual obstacle in the way of units, or where units even were at the time relative to the camera. All it would tell us is that players rotated their camera in that location; the presence of visual obstacles alone would not be enough to determine that they were actually blocking the players' view of units.

To do that, some very resource-intensive data events would have to be implemented, capturing where all units were at any given time during a battle, so that their locations could be plotted relative to the camera's location, and checking if the camera's gaze would be blocked by potential obstacles. Given that multiple units can be in multiple distinct locations that change every second, this would likely result in trillions upon trillions of rows of data being generated and sent, more than the rest of the game combined. Storage costs and query time would be astronomical.

After some back-and-forth discussion, the designer agreed that not only would the implementation of this be expensive, but it would also not yield any useful or actionable insights. In the end, the development team added an option to have an outline generated around units when they were obscured behind obstacles, enabling players to see them and select them without having to rotate the camera. It's always worth considering the why behind data requests; understanding the problem means you can come up with better solutions, or potentially not need a data-related solution at all.

We've gone through adding all kinds of data hooks and events to ensure you have as much data as possible, and have discussed examples of some data requests from developers. What about data initiatives of your own? Fielding requests from others to answer questions and solve problems is an excellent use of data, but sometimes exploring the data on your own initiative can provide answers to unasked questions, or solutions to optimise unoptimised flows.

Discounting strategy

Video game companies, like other for-profit companies, need to generate revenue and make a profit, some more so than others, depending whether they have to answer to a parent company, publisher, or shareholders. As with all companies, a fiscal or financial year has to be observed, which is a 12-month period used for budgeting and financial reporting. The fiscal year start and end vary from company to company, and even from country to country. For example, the United States has fiscal years running from 1 October to 30 September, while the United Kingdom and Japan fiscal years run from 1 April to 31 March.

Companies are usually incentivised to report a profit by the end of the financial year, as this profit is usually tied to bonuses. Because of this, companies will sometimes plan releases around the fiscal year dates, taking actions to ensure sufficient revenue comes in before the end of the year. In 2025 for example, 23% of all AAA games were released in February and March, meaning that 2 months out of 12 accounted for nearly a quarter of all AAA games released in 2025.

Why are a disproportionate number of AAA games released towards the end of the fiscal year? Why not earlier? Simply put, development teams need a lot of time and effort in order to release a polished, fun game. Sometimes a game has to be delayed due to unforeseen production issues or the need to revamp it, so the release date slips. When considering the player experience, it would make sense to delay the game as much as needed to ensure the best possible experience, but companies have to consider escalating budgets and profit margins. In my experience, company owners and publishers are more accepting of delays if they don't push the game's release date into the next fiscal year. As an example, for a fiscal year starting in April, a 6-month delay from May to November would not be great, but more acceptable than a delay from November to May, since the latter pushes the release of the game into the next fiscal year, meaning that it would be harder for the company to show that they had made a profit in this fiscal year.

Another way to bring in additional revenue before the end of the fiscal year is for companies to try to sell additional copies of their existing games using discounts. This works particularly well for digital versions of games since they have no physical production costs. Deep discounts of 50–75% can shift a lot of games, so even though the revenue per discounted game is

low, the volume of revenue generated can help with generating profit for the fiscal year.

Some years ago, we had a game release to critical acclaim and commercial success. Towards the end of the fiscal year, we noticed something strange; there was a huge surge in units sold, with a volume far beyond what we would normally expect outside of a deep 50–75% discount. We asked around, and it turned out that a decision had been made to heavily discount our newly released game by 90%, as we were just short of making a profit in that fiscal year. In terms of immediate short-term benefit, this decision made sense. Discounts, when used strategically, can take advantage of economic surplus to ensure both the companies and players benefit from selling and buying games at different prices.

ECONOMIC SURPLUS

Economic surplus refers to two things:

- Consumer surplus, which is the gain consumers (players in this context) obtain when they are able to purchase a product or service for a price that is less than what they were willing to pay.
- Producer surplus, which is the gain producers (video game companies in this context) obtain when they are able to sell a product or service at a market price that is higher or equal to what they are willing to sell for. In most cases, what they are willing to sell for must involve some profit, as producers (companies) do not wish to sell at break-even prices.

That last bit, break-even prices, applies more strongly to physical products that have a production cost even after the product has been designed, tested, and refined. Materials, construction, shipping, and other costs have to be applied, so there is a limit to how low the price can go for physical products. If the products aren’t selling at all, then selling at a break-even price or even at a loss is preferable to never selling them, but not ideal. While video games in the 20th century and early 2000s used to be primarily physical products, which needed a box, manual, cartridge/cd/dvd, transport costs, storage costs, and all the other costs associated, they have shifted significantly towards digital

distribution since then. Because of this, discounting digital games heavily can still generate some profit, no matter how small.

As with any other product, players aren't a monolithic group when it comes to prices. Some players are happy to purchase a game at full price, while others will only purchase at a certain discount level, whether that be 10%, 25%, 33%, 50%, and so on. While you could sell your game at full price only and try to obtain as much money from players as possible, this has two drawbacks. You'd lose out on revenue from players who are unwilling or unable to purchase at full price. You'd also lose out on expanding your player base; even if they purchase at a discount now, there is always the chance that they enjoy your game enough to become fans and end up purchasing sequels at full price in the future.

Essentially, using discounts enables you to capture consumer surplus that would be lost otherwise, a net negative to both you and to those players.

However, the important thing about discounts is to use them strategically. To use a simple example, imagine your player base is split into four segments of equal size; those that will purchase at full price, and those that will only purchase at 25%, 50%, or 75% discount levels. If you only sold your game at full price, you would lose out on 75% of your player base (for simplicity, let's assume that these hypothetical players never change their habits or preferences). Furthermore, you would lose out on around 60% of the revenue you would have earned had the discounts above been applied (and assuming each segment purchased at that discount level).

HOW DID I GET THAT 60% REVENUE LOST FIGURE?

Note: This section is used to explain the 60% revenue figure mentioned in the previous paragraph. If you don't like math (I do!), or really trust me (thank you!), or just don't want to read all these calculations, feel free to skip this box.

Abstract calculation:

25% of your player purchase at full price: 25% * 100% = 25%

25% of your player purchase at 25% discount: 25% * 75% = 18.75%

25% of your player purchase at 50% discount: 25% * 50% = 12.5%

25% of your player purchase at 75% discount: 25% * 25% = 6.25%

The percentages above add up to 62.5%.

If we only sold at full price, we would gain 25% and lose 37.5%.

This loss can be calculated as 37.5%/(37.5% + 25%) = 60%.

Non-abstract calculation with example $ figures:

Assume 1 million players in your player base, with a price tag of $60.

If you sold with appropriate discounts rolled out over time, and players purchased at the appropriate discount levels, you would get:

250,000 players purchasing at full price = $15 million in revenue

250,000 players purchasing at 25% discount = $11.25 million in revenue

250,000 players purchasing at 50% discount = $7.5 million in revenue

250,000 players purchasing at 75% discount = $3.75 million in revenue

In total, your revenue would be 15 + 11.25 + 7.5 + 3.75 = $37.5 million in revenue.

If you sold only at full price, such that only 250,000 players of your player base would buy your game, that would result in 250,000 * $60 = $15 million revenue.

Your loss in revenue by adopting a full-price only strategy (assuming 75% of your players never purchase at full price) would be (37.5–15)/37.5 = 60% of revenue.

The ideal would be to try to sell the game at full price for as long as possible, such that all or almost all players in the first segment who would purchase at full price are acquired. Then, the game would be discounted by 25% repeatedly for short periods of time, with the goal being to acquire all players who would only be willing to purchase at a 25% discount or better. Repeat until all four segments of players have purchased your game at 0%,

25%, 50%, and 75% discounts, and you have maximised your revenue (as opposed to losing 60% of it by selling only at full price).

Returning back to the 90% discount deal; while it did generate revenue to help fulfil fiscal year obligations, we were concerned that this might impact future revenue. Because the game had only been released recently (and was critically acclaimed), the expectation is that we would be able to use a standard staggered discount strategy, enabling players to purchase at discount levels that suited them over time, with the caveat that the deeper the discount, the longer the wait.

Our purchasing-level data included customer ids, meaning we could identify not just whether players had bought our new game, but our previous games, and at what discount levels. We had queried this data before, and knew that our user base did include segments that would pre-order/purchase at full price, or wait to buy at various discount levels. Over 50% of our players were also longtime fans and would purchase other games that we released, usually at similar discount levels.

We queried this data again, with the goal of learning the details of players who had purchased our newest game through this 90% discount deal. What we discovered was that over 95% of them had purchased from us previously, and nearly all of them had done so at various discount levels ranging from 25% to 50%. Unfortunately, this meant that while the 90% deal had brought in some extra revenue for the fiscal year, it had done so through existing players who had a history of purchasing multiple games from us at lower discount levels.

We also made sure to look at the data from angles that could paint the 90% deal in a positive light. For example, over 95% of the players who purchased were existing customers. That meant just under 5% were new customers who had never purchased from us before. Perhaps these new customers would go on to be valued long-term players and buy DLC or other games in our catalogue. Unfortunately, this was not the case; these new players did not go on to buy any of our other games, nor DLC for this new game. When looking at the new players' engagement, the vast majority tried our game for less than an hour before stopping, never to be seen again.

With the data above, we were able to show that the 90% discount deal was beneficial in the short-term, solely for the fiscal year, but negative in the long-term. As we could prove that the majority of players who had bought in this deal were existing players who were repeat buyers of our

games (at a discount, albeit much lower than 90%), this meant that future revenue from these players was now sacrificed for a much smaller sum. While we weren't able to undo the decision and subsequent consequences, these findings helped drive decision-making for the future, with an overall strategy of maintaining staggered discounts throughout the lifetime of our games, and only resorting to huge discounts several years after release. This ensured that our games captured as much economic surplus as possible, while providing players with deals that suited their needs.

A/B TESTING

Our games could detect if they had been launched for the very first time, which triggered several flags, including the First Time User Experience (FTUE), which directs the player into the tutorial unless they choose to opt out. Whether players opted out of the tutorial or not, they could choose to start their adventure with customised settings of their choice, or quick start into an adventure with pre-determined settings, allowing them to get straight into the action without having to worry about the details. If new players chose the latter option, the FTUE would use settings that the designers felt would provide the best possible experience for them.

Designers had a hypothesis that making the game easier using these first-time user settings would further improve new players' first experience with the game. If players had an enjoyable first experience due to lower difficulty, it would increase their engagement and retention. We wanted to use data to see if this hypothesis could be validated. Unfortunately, there was no way to do this with the data we had on hand, since the first-time settings had been static for some time, and there was no variant or benchmark to compare against. However, new players were buying our game on a constant basis, so what we could do was run an A/B test.

A/B testing is where you compare the performance of several versions of your game simultaneously to see which version has better engagement with players, which can be measured through KPIs like retention. In its most basic form, an A/B test uses two versions of your game, one in its default form, also known as the control group, and one with changes made, known as the variant. The control group, as the name implies, provides a baseline of KPIs for player engagement against which any changes made to the variant can be measured. Players should be selected at random when

determining who is allocated to the control group and who is allocated to the variant, to ensure that selection bias does not affect the results of the A/B test. The KPIs of a control group are also used for comparison rather than historical KPIs, because there may be global factors in the present that affect your game's performance that need to be controlled for.

As an example, let's say you did not have a control group and used DAU as a performance check between your game as it was (historical performance) and your game after changes are made (variant). If, at roughly the same time, a competitor title was released, and your game's DAU dropped, what was the cause of the drop? Was it the competitor title? Was it the changes to your game? Perhaps it was a mix of both? Without the control group, it is impossible to know for sure. With the control group, we could check if the DAU there dropped to a similar degree, a lesser degree, or remained the same, giving us an indicator of how much effect the variant's changes had in relation to the release of the competitor title.

Returning to the hypothesis on making the first-time user experience easier, we suggested patching the game and changing it such that half of the new players would get the standard first-time user settings (the control), and the other half would experience an easier version of the game (the variant). From there, we could measure which version performed better, via retention.

We worked with the designers to plan out the parameters of the A/B test, going over:

- How the new players would be allocated at random to the two groups, which we did based on whether their user ids were even or odd. This ensured random selection with no bias towards any of their attributes.
- The length of the A/B test, which we set based on the average number of weeks new players needed to complete a first playthrough of the game.
- The success criteria, dependent on whether there was a statistically significant increase in new player retention for the variant over the control group.

After the A/B test was over, we had to make an adjustment before comparing the difference in retention between the two groups. Some new players in both groups chose to switch their difficulties after quick-starting

their adventure and had to be excluded to avoid contaminating the results. After using statistical tests to compare the retention of both groups, we found that they were not significantly different, indicating that making the game easier for new players did not improve their first-time experience sufficiently to keep them playing the game more than the control group.

Despite the outcome here not yielding a positive result in terms of improving the game, the A/B test was still beneficial with the insight it gave us regarding difficulty. Through various UX Research studies some years later, I learned that a high proportion of our players valued the challenge our game provided, and toning down that challenge actually made their experience less engaging. This preference may have negated any gains we could have obtained from players who preferred a less challenging experience, giving us a potential why for the what indicated by the A/B test result.

Another benefit was that we had successfully set up a process for running A/B tests and could experiment with tweaking some of the other parameters for the new player experience to see if we could make their experience better. While we did not run any more A/B tests on this game, we did end up using A/B tests on a different project regularly, which yielded excellent results; while we did get negative or neutral results, we also gained positive ones and were able to improve that game gradually through multiple tweaks over a period of years, resulting in improvements in player engagement and revenue.

There are however some drawbacks to A/B testing; if the variant is worse than the control, then for a period of time, we would have provided the variant players with a worse experience. A/B tests also require quite some time to execute, both in the set up period and in the testing period. They also increase development costs, as two versions of the game must be supported via design, coding, and QA, even if only temporarily. Finally, because A/B tests prioritise incremental optimisation, it is difficult to tell if they are optimising towards a local or global maximum.

LOCAL AND GLOBAL MAXIMUM

Figure 7.1 shows what local and global maximums look like on a chart, as well as local and global minimums.

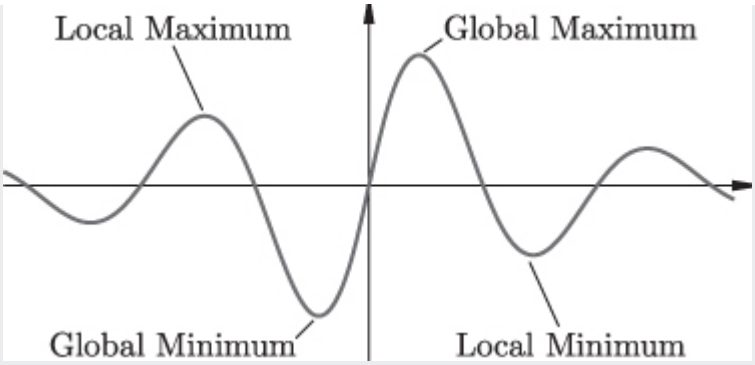

FIGURE 7.1 An image showing a graph of local and global maximums and minimums. ⏎
Creator: Inductiveload. Public domain.

If your game is on the cusp of achieving a global maximum with regard to performance, then optimising via A/B tests likely achieves that. However, if your game's performance is closer to a local maximum, A/B testing will not enable you to reach the global maximum. This is because when conducting A/B testing, the goal is to test if changes have yielded positive results (performance numbers go up), and use the successful variants. To reach a global maximum from a local maximum, you would have to accept negative results for a while (short-term losses, long-term gains), as per the chart in Image 7.1. Unfortunately, A/B tests cannot tell you if you are at a local maximum or a global maximum, and so it is difficult to know when or even if it would be appropriate to accept short-term losses for long-term gains.

Depending on the size of your user base, you may be able to run more than one variant, as long as sufficient players are able to experience each variant, and there is a clear success criteria that each variant fulfils. The next example will showcase a more complex A/B test with several variants.

A/B TESTING WITH MORE VARIANTS

The earlier A/B test example involved experimenting with ways to improve the first-time user experience. The impact of this is dependent on your game; if it's a brand-new game with no existing fanbase or a F2P game that

requires players to keep playing in order to generate revenue, a good first-time player experience is of utmost importance to keep players engaged. Compare this to an established game franchise, where a sequel is likely to bring as many, if not more, existing players than new ones; the first-time player experience is still important, but not as beneficial to existing players who already have an idea of what they are getting into.

Your players aren't monolithic; they have different attributes and needs. While it is not possible to cater to every single player's needs, it is possible to cater to segments of players that share similar attributes. To do this, it is important to understand how different segments of your players will react and engage with your game.

In one of the games I worked on, we had data on when our players were engaging with the game. One segment of players had high engagement, spending 20+ hours a week on average, most of it during the weekends. Another segment had lower engagement, roughly 2–5 hours a week, but they also played mostly during the weekends. Our game had an in-game shop that offered microtransactions for players to enhance their gameplay sessions, and naturally, these two segments of players used the shop mostly during weekends.

Based on these segments' play behaviour, we wanted to experiment with the items on offer in the shop. To that end, we set up an A/B test covering two variants. One of the most popular items in the shop was a buff or enhancement that would double the resources players gained for an hour upon purchase, enabling players to progress twice as fast in that period. Because the high engagement segment would have long sessions on the weekend (averaging around 5 hours), we wanted to experiment with two ways to enhance their experience. The first was a new double resource buff that lasted longer, matching their average session length. That way, the high engagement segment could simply buy this one buff and enjoy having double resource gain for one of their weekend sessions. While the new buff had a five-fold increase in duration, we didn't want the price to scale accordingly, and ended up setting it at only three times that of the hour-long buff, giving players better value for their money. Our hope is that this new buff would also give players more choice; by keeping the old buff as a purchase option, players could choose to buy one of (or neither of) the buffs to enhance their experience as they saw fit.

The second experiment was to see if we could not only give the high engagement segment a better experience and more value for their money but also encourage them to play more. To that end, we wanted to have the new buff last slightly longer than the average weekend session, to see if players who purchased the buff would be encouraged to play a bit longer.

We also wanted to do the same for the low engagement segment, who played 2-hour sessions on average during weekends. We wanted to see if a purchasable 2-hour buff that had double the value of the existing buff would be desirable to players at 2x the value but only 1.5 times the cost.

Rather than unleash multiple different buff purchase items in the shop at the same time (and risk overwhelming or confusing players), we decided to use A/B testing to trial them one at a time, targeted at different segments. The A/B test cases we settled on were:

Control: Shop unchanged for a portion of both segments.

- Case 1: A portion of high engagement players have access to a new 5-hour buff in the shop, with a cost three times that of the existing 1-hour buff.
- Case 2: A portion of high engagement players have access to a new 6-hour buff in the shop, to see if this new buff would increase their average playtime. The cost was also set at three times that of the existing 1-hour buff.
- Case 3: A portion of low engagement players have access to a new 2-hour buff in the shop, with a cost 50% higher than the existing 1-hour buff.
- Case 4: A portion of low engagement players have access to a new 2.5-hour buff in the shop, with a cost 50% higher than the existing 1-hour buff.

Since we could deploy updates for our game selectively to players, we allocated 33% of both segments to the control group, with the remainder split equally across their respective variants. Using the control case as a baseline, we could measure what % of players shifted over from the existing 1-hour buff to the new buffs, and whether the average session length in the week increased for players allocated to cases 2 and 4.

After running the A/B test for a month (giving players enough time to get used to the new additions in cases 1–4), we compared the results. In

cases 1 and 2, the high engagement players gradually shifted over to purchasing the new buffs, increasing revenue. Since the existing 1-hour buff was available to these players, the hope was that their game session experiences were also improved, since they could have simply chosen to purchase the original buff otherwise.

For cases 3 and 4, however, the low engagement players continued to purchase the original buff, with almost none of them shifting over to the new versions. We reasoned that because these players were not as engaged, it was likely they did not value the double resource buff as much as the high engagement players, and so the new buff was mainly viewed as 50% more expensive, rather than providing good value for money and a better match for how they played.

Finally, the average session length did not increase for cases 2 and 4, with both segments continuing to play as they normally had. We dug deeper here, since the average session length, being an average, might have masked any improvements, but there was no statistically significant difference between the session lengths of players who purchased the new 6-hour buff compared to those who purchased either the 5-hour or 1-hour buffs.

Having learned valuable insights from the A/B tests, we reverted all the variants and deployed a patch to all players that added a new 5-hour buff at three times the cost of the existing 1-hour buff, which overall increased revenue from the high engagement segment, and produced no drawbacks that we could detect, even a year later (a concern was that the longer buff would decrease engagement over a long period of time, since players would gather resources faster and progress faster, but this concern was not realised).

The examples and methodology presented so far have been reactive; players show a certain behaviour, and we react. Or we make some changes, measure how players react, and make decisions accordingly. While this is a great use of data, there is an additional facet that can improve the player experience and the longevity of your game: predictive analytics.

Predictive analytics with players who are likely to quit

Over time, we built up a detailed profile of how our players behave and react in our games. In some of our games, players would play all week long, mainly in the evenings. In other games, with different players, their gaming time would be focused mainly on weekends, as detailed in the example above. In our single-player games, some players would play all the way to the end, and never pick up the game again. Some would quit partway through, while others would take long breaks before returning, then taking another break, playing sporadically.

For the players who quit and those who played sporadically, we wanted to learn if there were any underlying factors we could resolve. By investigating these players (no new session events from them entirely or in a while), we identified a commonality for a large portion of them; they had experienced losing streaks due to dying in-game. There was no one defining streak number, but once these players had lost 3 or 4 or 5 or more times in a row, they either quit or took a break from playing our game. There was also no singular mission or level in the game where these losing streaks were taking place, so it was not a case of a difficulty spike or a common challenge being too difficult and needing to be balanced. As we saw no similar correlation between losing streaks and active players, this was a potential avenue for A/B testing and improving the player experience by removing the frustration of losing streaks.

We ran an A/B test where the variant version of the game had additional coding that triggered when players died twice in a row in a span of a few minutes or less; it would apply a small negative modifier to the enemies' health and damage output, making it slightly easier for players to survive against and defeat the AI-controlled enemies. If players died again despite the modifier, a larger negative modifier would be applied to the enemies, and so on. The goal here was to gently ease the challenge so that players would not feel patronised, while attempting to remove the frustration of losing streaks. Once players overcame the hurdle that was killing them and moved on to the next level, the modifier was removed, restoring the game's challenge back to the default.

The variant showed good results after a few weeks; we compared retention of the variant against the control, and it was significantly higher such that we ended up patching in the new coding in for all players. While this did not stop players from quitting or taking breaks (we hadn't solved all the friction problems), it did significantly reduce the number of players who

were experiencing losing streaks, which in turn improved the player experience and the game's retention.

ANALYTICS CONCLUSION

Congratulations! You've reached the end of the analytics section. If this is all you were interested in, I hope you found it useful. If you are looking for more, consider this; in all the examples we've discussed above, there has been a great deal of data on what happens, but much less data on why these things happen. Previous examples have shown that with careful planning and consideration, speculation can be narrowed down to only a few options, but there is another path to getting the why, as the next section on UX research will demonstrate.

REFERENCES

Release windows of AAA games: https://newzoo.com/resources/blog/what-the-data-tells-us-about-the-true-cost-of-traditional-release-windows#:~:text=Above%2C%20we%20see%20two%20significant,70%25%20of%20all%20AAA%20games.

Local maxima and minima: https://commons.wikimedia.org/wiki/File:Maxima_and_Minima.svg

8 Introduction to UX research

DOI: 10.1201/9781003735335-8

INTRO

The previous chapter detailed examples of behavioural data, what players are doing, how they react to changes, and using trends to aid in problem solving and decision-making. This chapter focuses on user experience (UX) research, which can generate both behavioural and attitudinal data, providing the what and the why. Like analytics data, UX research data can be collected and used through the development process of games, and beyond. UX research studies can provide data to aid with decision-making or complement other data sources (such as analytics data) to provide a holistic view of the player experience of games.

The desire to understand the why behind the what of our data and combine them was why I started my journey into UX research. It began with a Games User Research (GUR) conference which included various case studies from UX professionals working in the games industry. While most of the talks were interesting and gave us (the analytics team) new insights on how to get to the why behind our data, one talk in particular stood out, as it was given by a UX researcher from a company that made comparable games to ours. His talk was excellent, and one point stood out; this competitor company had a full UX research department, but no analytics department. This was especially interesting since we had the inverse situation, a full analytics team but no UX research presence.

CONFERENCES

Conferences are a useful way to network and gain information from other professionals working in the industry. Often, problems that you are working on have already been solved by others, saving you some time. Or, you might learn about problems you never knew existed, aiding you in avoiding said problem in the future.

My lead and I reviewed notes from the conference and supplemented them with use cases of UX research from articles and videos. With all this data, along with some of the "what but not why" data examples we had (many of which have been detailed in the previous two chapters on analytics), we went to the company CEO and COO to present our case for delving into UX research as part of our work. I wanted to be able to allocate a portion of my working hours towards developing UX research, with the goal of being able to run at least one professional-level study by the end of the year, at which point said study's results would be evaluated to see if this UX research initiative would be progressed further. This was approved, based on the analytics work we had done in the past and a desire to be able to make more effective decisions in the future.

GETTING STARTED

Learning about best practices and what others have done is good, but there's nothing quite like running a research study yourself for the first time. The goal was to go through the process from start to finish, implementing best practices to see how they worked in practice, make mistakes and learn from them, and get a feel for how it all worked. I reasoned that it would be better to run multiple quick and cheap studies to learn and iterate, rather than one long and expensive study, especially since the short-term goal was to learn about the UX research process, rather than getting useful results to inform decision-making, which would come later. For this first study, and several after, I worked with other members of the analytics team, borrowing their time and brainpower as needed. These next few examples will focus more on the process, rather than the results or decisions after.

First playtest

Planning, recruitment and logistics

We decided to start with a usability test, which is a study aimed at gathering data on friction points in the game. These friction points can include players not understanding game mechanics, having problems with the controls, confusing UI, or text, or simply getting stuck and not knowing how to progress. Good usability is an important requirement for having a good experience, since it's difficult to have fun if you don't understand what you should be doing. Usability tests also have a benefit in that they require less players to gather meaningful data, which was helpful given the exploratory scope of the first few research studies.

Rather than test a build of an upcoming, unreleased game, we chose to use an already released game. This simplified the process, since we did not have to look for a suitable development build or worry about bugs.

To keep things simple, we decided to test the earliest portion of the game; the tutorial, and to that end, recruited other employees who had no experience with our games. We recruited members of HR and Finance who played video games regularly on PC, but had not played this particular type of game before, simulating the experience of players new to the genre. While we could have recruited people who had never played video games instead, we wanted to keep things as simple as possible for this first playtest, since the game being tested was quite complicated even for people with gaming experience.

We had no UX research lab for dedicated playtesting, but we did have our own work computers in the office. We installed Open Broadcast Software (OBS) and set it up on my work PC so that it would record the game, the player's inputs via keyboard and mouse, a webcam recording the player's face, what they were saying, and a timer. Even though we would be observing and taking notes on each player's experience of the game, having OBS recording their actions would allow us to rewatch parts we wanted to review, as well as enable us to create clips showcasing issues that players were experiencing. Having their actions in-game, their interaction prompts via keyboard and/or mouse, and their expressions via webcam was useful supplementary data to their feedback. In some cases, the feedback itself became the supplementary data, as a clip showcasing players struggling to

progress was in itself very compelling data. A picture is worth a thousand words, a video even more.

Having decided on the goal, the players, and the playtesting venue, we had to consider how we wanted players to be exposed to the game, which can vary depending on what's being playtested. For example, if you're testing a very specific slice of the game, do you want to start from the beginning of that slice? Or do you want players to experience the process of starting up the game, watching intro movies (if there are any), changing options or controls to suit their preferences, and going through the tutorial first? The first method gets straight to the point, but it is not a realistic player journey. The second more fully represents what players would experience when playing your game for the first time, but adds extra time to the playtesting. All these things and more needed to be considered, but we'll go into more detail about that later in this chapter. For this prototype playtest, since the goal was learning about the UX research process and having players experience the introductory section, they could experience the game as a new player would.

The playtesting prep

We arranged for the players to be tested one at a time, to allow for detailed observation; it is difficult to watch several people playing at once, especially if we want to thoroughly observe their progress and pitfalls. We also budgeted buffer time between playtests, in case players were late. This would apply even if they were external players rather than co-workers, since we would need the buffer time to allow for mishaps to occur, whether that be public transport delays, traffic delays, or players simply being late. The buffer time means that players don't have to feel pressured or rushed to get through their session quickly, before the next player shows up.

Prior to playtesting, players were given a very brief explanation of what they had volunteered for. An example: *You'll be playing the introductory part of this game for about 60 minutes. You do not need any prior experience with this game, nor do you need to bring anything, as everything you will need to play will be provided.* This gave players an idea of what they would experience, so there would be no surprises, and also helped ensure that they would not attempt to play the game prior to the playtest (remember, this game was already released), since we wanted players who

had never experienced this game before, to simulate the new player experience.

Playtesting can be intimidating, both for the moderators and for the players, especially if the latter have never experienced a playtest before. Because of this, an important part of playtesting involves ensuring that players are comfortable and put at ease. This starts even before the players arrive; we made sure the playtesting area was tidy, with all surfaces and interactable objects given a once-over with antibacterial wipes. Pre-COVID, this was a nicety, post-COVID, it became essential. Since these first sessions were short, a bottle of water was provided in case players felt thirsty. When players first arrive, the sight of a tidy and clean playtesting area can help put them at ease.

As with any formal interactions, small talk can help make the setting seem less formal and intimidating. This will depend very much on the moderator and the player; some players are more chatty, some just want to get right into playing the game. What I've found works best over the years is to have some rough guidelines rather than a detailed plan, and adapt as needed. How was the trip over, what games are they currently playing, and so on.

Having seated the player down, it's helpful to let them know they can do whatever they need to get comfortable; adjust the chair height/lean, tilt the monitor, change the positions of keyboard and mouse, and so on. Water bottles are provided, and if they need more, they only have to ask, and the same is true of bathroom breaks. Once they are comfortable, it's time to get started with the playtesting.

We reminded them what the goals were, repeating the earlier statement about what they would be playing and the rough length of the session. We also informed them about the structure of the playtesting, so they would know what to expect; they would be playing the game while being recorded, followed by some questions about their experience. As we were testing with co-workers in this prototype playtest, they did not need to sign a Non-Disclosure Agreement (NDA), but they did need to sign a General Data Protection Regulation (GDPR) release form, as we were recording them, resulting in personal data that requires security precautions to ensure their data is protected to comply with GDPR. In the interest of ensuring that players were comfortable with the playtesting experience, they also could opt out of having their faces be recorded. Thankfully, OBS allows for

recording in a modular way; the game screen, webcam output, timer, and player inputs are all separate components that can be enabled or disabled for recording. Players who were uncomfortable with their faces being recorded could opt out of this (in my years of playtesting, only a handful chose this option).

NDAS AND THEIR EFFECTIVENESS

In this prototype study, players did not have to sign a NDA as they were co-workers, but in the vast majority of playtests, you will need players to sign a physical or digital NDA form, ideally as part of the recruitment process. Whether you are testing a game in development that has not been revealed to the world, or new features to be added to an already launched game, you need protection to ensure that players won't leak details once the playtest is over. In reality, NDAs in playtests are not effective in a legal sense; if a player did leak details of your game, taking them to court and suing them for breach of NDA is neither an effective nor desirable action. However, the NDA itself is effective as a psychological deterrent and makes your legal department happy (if you have one).

Because the very nature of playtesting requires a degree of critical feedback, it is important to make sure players feel comfortable in being able to do so. To this end, it is helpful to have a disclaimer that you, the moderator of this study, did not work on the game at all. This lack of stated investment on your part implies that you are not emotionally invested, allowing the player to be critical without having to worry about hurting your feelings. This is a standard disclaimer I've used in my playtests

WHAT IF YOU ARE INVOLVED DEEPLY WITH THE GAME?

The disclaimer works for me because I don't work directly on the games, in the sense that I don't design or code them. But what if you do? What if you're a designer or a programmer working directly on the game? What do you say then?

From working with game designers, UX designers, and others who have run playtests, the best you can do is tell the truth. You have worked directly on the game, but are interested in improving it, and the best way to do that is to learn about what works and what doesn't, even if the latter is quite critical. You welcome criticism in the name of making the game as good as it can be.

You can also lie and say you don't work on the game directly, though this can have mixed results, depending on your reactions to negative feedback and ability to take it in stoically, and your ability to lie. If players provide criticism and feel that it is being received badly, they may modulate their subsequent feedback, hindering your ability to get useful data.

> I have not worked on this game directly, and any feedback, positive or negative, is welcome. I won't be offended if you think the game is terrible, and I won't feel pride if you think it's great. You can say anything you like.

Finally, in the interest of ensuring that players are comfortable, do allow them the chance to ask any questions, and make sure they know they can ask questions at any time. With all of that out the way, playtesting can begin!

The actual playtesting

First, we ensured that OBS was up and running and recording, then launched the game for the player. In time, we would develop a remote system that enabled us to start up OBS recording and streaming, launch/close the game as needed, open up surveys in browsers, and more, but for this prototype playtest, everything was done as simply as possible.

That included how we took notes, which was as simple as using pen and paper to note down what the player was doing as they progressed through the first-time user experience (FTUE) section. These notes included not just where the player was struggling, but also areas where they had progressed with no issue. Noting down what parts of the FTUE worked well and what did not helped with building an overall picture of where the friction points

were and were not, and also served to answer any potential questions that the developers might have about how well each section of the FTUE worked. In addition, it may not always be clear what players are doing; are they taking their time to read through some text, or are they lingering on a screen because they are unsure what to do? Are they going back and forth in an area because they are exploring or because they are stuck? Taking down notes on these periods of uncertainty allows for more informed questions during the interview period.

Players may ask you questions during the playtesting, and what answer you give them depends on a very simple question: will the answer affect their understanding of the game and their experience? Questions like “Can I go to the toilet?”, “Is it OK if I change the sensitivity of this mouse?”, or “Is it OK to take a short break?” are all perfectly fine to answer, especially in the affirmative. However, questions like “What should I be doing now?”, “Is this the right answer?”, or “What happens next (in the game)?” have to be considered carefully. This is because answering any of these questions with the answer they want may affect their experience of the game or change what they choose to do next, which would not be representative of how the game would be played normally.

Let’s use the “What should I be doing now?” question as an example, as this happened in the prototype playtest, and examine it from multiple aspects. At minimum, this question likely means the player is stuck or unsure what to do, and is asking for help. You could give them this help, but this would leave some questions unresolved. Would the player have figured out how to solve the problem on their own? Would they have been stuck? By giving them the answer, we now lose out on this information, and subsequently have less data with which to make a decision on what to do to improve the game (or leave it alone). However, we don’t want to be too negative in our answer either, since we want the player to feel comfortable. There are a few ways to do this.

One is to phrase your answer in the form of a question, “What do you think you should be doing?” With this, you’ll get more in-depth information about what the player’s situation is, what they’re thinking, and whether they are stuck. The player might be on the correct path but unsure about it, and just wants validation. The player might just be stuck and their explanation on how they got to this position and what they are thinking may provide useful context for the problem. You could also ask the player to keep trying,

as if they were at home, playing the game without anyone to ask directly, and see whether they are able to figure out what to do, through aid or context from the game (good), or through brute force or trial and error (less good), or if they are unable to progress at all (worst). These outcomes would give you and the developers some information on the state of the game's usability.

We obviously don't want players to be stuck indefinitely, since that is a frustrating experience and limits what they will experience in the game. But we also don't want to help them too soon, for all the reasons outlined above. So what do we do? What's the correct amount of time to wait and watch players struggle, a balancing act between gathering information and not spending too much time on this one section? Unfortunately, there is no definitive answer or specific time frame for this. It will depend on the section of the game being playtested, and how trivial or difficult it should be to resolve. If it's a trivial problem, something that a player should be able to complete or understand within seconds, then you may only need to give them a minute or two. In that time, observe their behaviour. Are they visibly trying to solve the problem? Perhaps they aren't even aware of the problem, and are engrossed with something else. We want to keep interruptions to a minimum if possible, to avoid interfering with their experience.

If it's a complex problem, then you may need more time to be confident that you need to intervene. I've left players alone to struggle for several minutes, with some players eventually solving the problem on their own, while others were not able to do so. For the latter, you'll need to intervene in the interests of preserving playtesting time, note down that you had to intervene, and why. The why of this can be gained as part of the intervention, asking the player what they're trying to do and what their thoughts are. In the prototype playtest we ran, the player was meant to interact with three points of interest, denoted by glowing markers on the map. However, while two of the markers were within sight of each other, the third was some distance away, and the player had likely not seen it. After waiting a few minutes, with the player visibly exploring the immediate area, sometimes checking the same location over and over again, we intervened to ask what they were trying to do. They explained that they were stuck and were trying to figure out where the last marker was. They explained that since the other two markers were in close proximity, the

location of the third must also be close by. Since this explained why they were stuck, and why they did not explore elsewhere, we directed the player's attention to the distant marker so they could continue. Had this been a real playtest (and not a prototype to learn), the player's explanation and footage would have been useful for the development team to understand the problem and the best ways to fix it.

Before moving on to the interview process, we should consider how to go about ending the playtesting portion of the research study. We've talked about setting a suitable amount of time for players to experience the relevant slice of the game. However, this time is an estimate, and players can surprise you, either by taking much longer than expected or speeding through. It is likely that you'll need to be flexible from time to time to accommodate this, as we had to be with two players in the prototype playtest, and in many playtests since. One player sped through the FTUE, completing it in a fraction of the time expected. This player was very efficient at following the game's instructions and did so rapidly, such that we were concerned that the player was carrying out tasks given without necessarily understanding the lessons imparted. Because of this, and because of the extra time available due to their speed, we decided to have the player experience a portion of the game post-FTUE, to check if our concerns were valid. Thankfully, they were not; the player was able to play through the game without any issues, and feedback provided later showed that they had absorbed and understood what was being taught.

That worked well for the player who sped through the section being tested, what about the other? That player had the opposite experience; they struggled repeatedly with basic controls and concepts – we learned a lot from this player – and because of this, they had not made it even halfway through the FTUE by the time the playtesting period was over. While it can be tempting to try and extend the session to ensure the entire section was tested, we decided it was best not to, for several reasons. First, we had obtained a lot of useful data on why the player was having trouble, most of which stemmed from the same reasons – the player was not used to this game genre and the control scheme, so further testing would only reinforce this finding. Second, we had an agreement with the player that the playtest would last up to a certain amount of time, and going over that allotted time would be a betrayal of trust, which could lead to bad data as the player might feel frustrated at having to play longer than agreed, tainting their

experience. While it would be nice to have all players experience exactly the same portion of the game, particularly for comparison purposes, sometimes you have to know when enough is enough and move on to the feedback portion of the study.

The feedback

So, the playtesting period is over, you have a recording of the player's session, and hopefully a bunch of observation notes and a pre-determined list of questions you'd like to ask. Ideally, you'd be able to interview the player in a dedicated room, so that they can answer questions without bothering others and without fear of being overheard, the latter being particularly important if they want privacy or are worried about offending others with negative criticism.

As detailed earlier, we had no lab at this point, and so we used my own work PC and work space as the playtesting area. However, we did book a meeting room to conduct interviews in the interest of providing each player with a safe space for feedback. In the interest of player comfort, we also made sure each player was aware they could take a break, have a drink, or go to the toilet, before the interview started.

During the interview, it is important to be impartial and ensure that any questions asked are as neutral as possible, without leading players to a desired answer (consciously or unconsciously). For example, let's say you suspect a player might be struggling with a game mechanic, and think it might be because the wording used to describe the mechanic was not clear. A very leading question here would be along the lines of "You were struggling with [mechanic X], was it because of the wording?" This sort of question is both leading with regards to the cause and makes an assumption that the player was in fact struggling. A more neutral question to ask would be, "How was your experience with [mechanic X]?" This neutral and open-ended question makes no assumptions about the player's experience with the mechanic, allowing them to provide an unbiased answer. If the subsequent answer isn't detailed enough, follow-up questions such as "Why was that?" or "Could you go into more detail on that?" can be asked.

WHAT IS A GAME MECHANIC?

A game mechanic can be thought of as a rule in how a video game works, or rules that define how the player interacts with the game.

For example, in a board game such as Monopoly, rolling the dice to determine how many squares your avatar advances is a game mechanic.

In a video game like Pokémon, catching a Pokémon is a game mechanic, as are actions like levelling them up, using them in duels, and giving them new skills. In a turn-based strategy game, armies might have limited movement defined by some stamina or action bar, which would also be a game mechanic.

Actively listening to what players are saying – as opposed to simply taking down notes and moving on to the next question when they are done answering – is also important. Active listening doesn't just enable you to ask good follow-up questions to dig into the experience players are having, but also ensures that players are feeling like their feedback is being heard and matters. Passive, robotic listening, if noticed by players, can lead them to feel that you are simply ticking questions off a checklist and not listening to what they have to say, diminishing their desire to elaborate or think about their answers, leading to inadequate data. Active listening can also include non-verbal cues, such as the player fidgeting; perhaps they are nervous and need to be put at ease. Or perhaps the player has something they want to say that the questions have not covered yet; asking them an open-ended question about their experience as a whole may yield an answer.

Throughout the feedback process, it is important to be as open-minded as possible and follow lines of inquiry to a natural end, making sure you have a good idea of what the player experience was, how they felt about it, and why, as this will lead to much easier and thorough analysis.

Analysis and writing up the report

As the focus was usability, all observation notes and questions were focused solely on whether players were able to progress and understand game mechanics, with emphasis placed on any friction points discovered. So too was the focus of the report; we compared notes on player behaviour and feedback, and discussed how best to present the usability issues observed.

Should we showcase each player's journey and struggles, or aggregate the results? Should we prioritise frequency (how many players experienced a friction point), or impact (how much effect the friction point had on the player experience), or recurrence (did players constantly experience it)?

We would later come up with a framework with which to better prioritise usability issues, but in our inexperience, we chose to document everything in the report. The first report we created looked nothing like later ones; such was the experimental nature of this first study. We provided the full footage of each player's experience, with text hyperlinks that described the friction point and linked viewers to the exact time in the video where players encountered said issue. As per the experimental nature of this first study, the report was labelled "prototype", with wording to indicate the findings should not be used to make decisions on the game.

Having completed our first experimental UX research study, it was time to review the process; what went well, and what didn't. To that end, we share the prototype report with several developers to get their feedback.

What went well?

Using videos was both helpful and not; more on the latter in the "What went wrong" section below. It was helpful for developers to be able to click on a hyperlink and see the exact moments players experienced friction.

Our initial OBS setup worked well for an experimental playtest; it worked as we had hoped, enabling us to review player behaviour when needed, and was simple to set up and have it record gameplay footage in the background without bothering players. Over subsequent playtests, we iterated on our OBS recording scenes, adding useful components such as timers, information cards, keyboard/mouse/controller input trackers, and more.

The preparation work we did beforehand worked well, such as writing a short script for introducing the goal of the playtest to players, so they knew what to expect. The script also standardised the experience for all players, so we did not unintentionally give one player more or less information than others.

Finally, the prototype playtest itself went well, as we made quite a few mistakes and were able to learn from them, as you'll see below.

What went wrong/what did we learn and iterate on?

While the parts of the videos that showcased player struggles were useful, the videos in their entirety were not. Developers weren't sure if they were meant to watch through each player's entire experience (each video was an hour long), and the amount of footage available felt intimidating. We resolved this in future reports by using video editing software to isolate issues discovered, usually as 10–60 second clips. The raw footage of players' entire experience was available to view as a separate link, tucked away at the bottom of the report as supplementary data, should developers need it.

Another thing we learned from the prototype playtest was to have a set of standardised questions for each player, based on the needs of the playtest, ensuring that all players have a chance to provide feedback on the most important topics. Without a standardised set of questions, we received answers from some players on certain areas and not on others, meaning that we had a hodgepodge of feedback across various areas, which was still useful, but not comprehensive.

The issues listed in the prototype report had no priorities attached, and this lack of prioritisation would have made it difficult for developers to allocate resources accordingly, had the UX research test been real and not a prototype. Over time and multiple studies, we evolved the reports to include priorities for each issue based on three parameters: frequency, impact, and recurrence, with context sometimes acting as a modifier.

Frequency: How many players experience the issue? For example, if 2 out of 6 players experience an issue, that issue is less of a priority when compared to one that is experienced by 6 out of 6 players. The number of players that would classify an issue's frequency as high is subjective, but as a rough rule, we used 50% or more of players tested as the criteria. An additional factor to consider here is the actual exposure to the section of the game where the issue was experienced. For example, games tend to have mandatory sections and optional sections that players can experience. 1 of 6 players who experience an issue in a mandatory section would be treated as low frequency, but 1 of 2 players who experience an issue in an optional section would be treated as high frequency.

Impact: How badly the issue affects the player experience? For example, if the issue is related to understanding an essential game

mechanic, and the player doesn't understand it at all, that would be considered high impact. If the player doesn't understand it initially, but figures it out in an acceptable amount of time (what is acceptable is going to vary based on the complexity of the mechanic), then it would be considered low impact. If the issue is related to the player being unable to progress at all, then that would be considered high impact, whereas an issue where the player is blocked temporarily but is able to overcome it (again, the definition of temporary is going to be subjective), then that might be considered low impact.

Recurrence: How often the issue recurs for a player? For example, if the issue is one that prevents the player from progressing, it recurs infinitely since the player isn't able to get past it. If the issue is one where a game mechanic is not understood, then recurrence is more variable depending on how prevalent or important that game mechanic is. A game mechanic that multiple players do not understand is a high recurring issue if the mechanic is required on a regular basis to play the game. For example, in a game where the player is required to pick locks multiple times in the first few hours of gameplay to progress, a lack of understanding of how to do this would be a high-recurring issue. On the other hand, if lockpicking was required once, and then never again (one-time mechanic), it would be a low recurring issue.

These three factors – frequency, impact, and recurrence – determined how we eventually set the priority for observed and reported issues in UX research reports in a consistent manner. An issue that was experienced by most players and impeded their progress constantly would have high frequency, high impact, and high recurrence, and we would rate that as High Priority, using a red background to provide a second channel of information and allow for easy scanning. Conversely, an issue that only one or two players encountered with minimal impact on their gaming experience might be rated as Low Priority (yellow background), something worth devoting resources to fix if time allowed.

I mentioned earlier that context sometimes acts as a modifier for setting priorities. Depending on the playtest goals and the development team's focus, some issues might be elevated or lowered in priority. For example, if the development team was implementing a brand new mechanic that was going to be a key selling point for the game, then ensuring that this new mechanic had as little friction as possible would be highly important, and

this would lower the requirements needed for any issues around this mechanic to be rated as a high priority. Alternatively, the focus of the playtest might prioritise or deprioritise certain game mechanics or game sections depending on the state of the game build being tested. For example, if an important game mechanic in the build being playtested was not fully fleshed out or did not have enough onboarding implemented yet, any issues around that mechanic would not be representative of how players would react to it when completed. Such issues might either be given a lower priority or have a note alongside the issue explaining the circumstances.

Contextual situations like these are obviously subjective, and as such will vary from game to game and from playtest to playtest. When context is applied to the prioritisation of issues, it is always helpful to add an explanation for the context and reasoning behind the altered priority, not just for the developers, but for UX researchers who may need to refer to the report in the future. The same is true for the three-factor system of frequency, impact, and recurrence, as a good prioritisation system is not just consistent and easily reusable, but transparent, so the development team can understand how the priorities are defined, enabling them to process and prioritise feedback accordingly.

Over time, we also transitioned the report format from being video-based to being issue-based, using the system of priorities defined above. Furthermore, as we collaborated more and more with the development team and other stakeholders to define the focus of playtests and research studies, we began adding sections to the report to aid in readability and scanning for issues that mattered. For example, the UI team (designers and artists) were more interested in UI issues discovered, so gathering all UI issues in one section and prioritising them within allowed that team to focus on that one section. Usability issues regarding controls would go into a controls section, and so on. Each issue had a format: what we observed, what feedback we received, and our analysis of both, with supplementary images and/or video clips added in where needed.

I've jumped ahead quite a bit with the iterations and improvements, all of which were implemented over time and multiple playtests. We'll look at these improvements more thoroughly later on in Chapter 9. For now, we will move on to the second playtest.

Second playtest

Shortly after sharing the prototype report, we were asked by one of the development teams if we could run a playtest on one of our other games in development, as they wanted to get early feedback from external players.

Recruitment

Thankfully, we'd already had a discussion with our Finance and Legal departments about what was possible when it came to recruiting external players.

We were based in the UK and were advised by our legal team that paying money to recruit external players for playtesting could run afoul of UK employment laws. However, providing these external players with non-monetary compensation would be an acceptable alternative, especially if it was a one-off, i.e. we do not recruit the same players again. We chose Amazon vouchers as compensation, since Amazon is widely used in the UK, and would be an acceptable incentive, particularly given the constraints we had and the audience we were targeting.

As this was still the first year of building up UX research processes and procedures, we didn't have much budget to work with, so advertising for players was out of the question. While we had our own company website, traffic to the site wasn't high enough that we could be confident of recruiting sufficient players, especially since the budget constraints meant we could not reimburse them for transport fees. So, recruiting players from afar (more than an hour away by car or public transport) was not viable at the time.

Our legal team also advised that recruiting players under the age of 18 in the UK would be challenging, as under-18 players would need to be escorted by a parent or suitable adult, adding both logistical and budgetary challenges. While video games nowadays are suitable for a wide range of audiences, some games, like Roblox, have under-18s as a core portion of their audience. Thankfully, the games I was working at the time were not focused on children or teens, so it was an easy decision to not recruit any under-18 players.

Given the budgetary and logistical constraints we had at the time, we decided to recruit our external players from universities. Recruiting

university students had a lot of built-in benefits that matched our constraints; university students would generally be 18 and over and would also be more than happy to spend some time playing video games and providing feedback in exchange for Amazon vouchers. Thankfully, there were several universities close by, such that driving or public transportation were viable options for these students. As our company had dealt with these universities before, via community outreach in the form of guest speakers, internships, and job opportunities, they were happy to work with us on requests to recruit interested students.

We sent the universities a short blurb stating that a playtest would be held at our company offices for several hours, and students willing to travel down would be compensated with an Amazon voucher. The value of the voucher itself was tweaked based on the length of the playtest; 1-hour playtest vouchers were naturally lower in value than 4-hour playtest vouchers. The blurb included a link to an online survey (Microsoft Forms), which had some screener questions to help us select suitable participants who signed up. While we wanted fresh eyes on the game, we also wanted people who had played video games before, particularly this genre of video game we were developing. While it can be useful to recruit players who are unfamiliar with the genre or video games in general, that was not the purpose of this playtest. The game lacked a dedicated tutorial or onboarding at this stage, so the development team wanted to get feedback from players who were already familiar with the genre. The blurb also detailed what would be expected of participants should they be selected; they would have to sign a NDA and a GDPR release form to protect the confidentiality of the game and get explicit permission to use players' personal data.

Makeshift playtest lab

With that, the recruitment pipeline was all good to go. We also needed a space where players could playtest. Since external players would be part of the playtest, using my work PC was not an option, as that would grant them access to all kinds of confidential data, not just the development build of the gaming they would be playing. Like most companies, we had multiple meeting rooms, all of which could be booked. We asked the Information Technology (IT) department which rooms were booked the least, and appropriated one for several days to use as a makeshift playtest lab.

Setting up a makeshift playtest lab isn't too complicated, especially if the playtest only requires one player per session. At its core, a playtest lab needs to be an enclosed space with a suitable device (PC, console, smartphone) for the player to engage with the game comfortably. Since the development build was on PC, we requested a spare PC that had restricted access to our internal network, and set up a playtest area using the existing desk and office chair in the borrowed room, making sure we installed appropriate software like OBS. The final thing we needed to do was move remaining furniture (armchairs, sofa) off to the side and ensure there was a place for a moderator to sit close by and observe.

Other logistics and rehearsal

The plan was to send suitable applicants an email with the appropriate dates and times for them to attend, and our company address, with some instructions on how to reach our visitor parking if they were driving, or how to get from the train station to our office if they were using public transportation.

It can be helpful to conduct a rehearsal prior to the playtest date, so you can simulate what players might need and prepare accordingly. Mindful that this would be our first playtest with external players, we did that just, simulating what a player would experience from start to finish, not just within the playtest lab, but before they even arrive at the company.

The rehearsal showed us the following:

- Someone needed to be waiting for the players, who might show up early (or late). If a receptionist was available in the period before the playtest began, we enlisted their help, and if not, one of us would sit and wait in the reception area to let players in and escort them to the playtest area.
- Offer players a chance for a toilet break after a potentially long journey!
- Ideally, the playtest area (makeshift or otherwise) should be close to toilets, and ideally not require players to walk through areas that have other confidential content (most development floors will have material such as artwork or figurines that identify future games).

- Offers players a drink, ideally unopened bottles of water (to address sanitary concerns).

Playtest, feedback, and report

As this was the second playtest, the processes for the above were mostly unchanged from the first playtest. As we had treated the work colleagues in the first playtest as if they had been external players, there was no change in decorum needed during the playtest itself, since the procedures could be reused.

What went well?

Recruiting university students worked very well for the initial playtests in our first year, as they fit well with aforementioned budget constraints and enabled us to recruit cheaply.

The rehearsal flagged some tasks we could have resolved before the playtest began. These tasks would have been resolved easily on the day of playtesting, but doing so prior ensured a smoother experience for us and the players.

We made sure to get the contact numbers of selected players during recruitment, and provided them with a number they could use to contact us in return. This wasn't used often, but when it was, it was usually in the form of players being late or needing help finding the office, which was helpful for both sides.

What went wrong/what did we learn and iterate on?

Having players sign the NDA and GDPR release forms in person wasn't worthwhile. It takes time out of the playtest session to have players read through the documents and sign them, and then the physical documents have to be stored in a secure location, permanently. Finally, we needed players to sign both the NDA and the GDPR release form in order to be able to proceed with the playtest, so any players who changed their mind would have wasted their time travelling over. After speaking to the legal team, we settled on a much better solution, which was to use an electronic agreement service called Docusign. This enabled us to send the NDA and GDPR release forms to selected players so they could sign online, prior to

them attending the playtest, and any who elected not to sign could be replaced days before the playtest began. Storing these digital agreements was also significantly easier, and the service cost virtually nothing, as the company already had a Docusign subscription for other needs.

A player showed up with a crutch, as they had recently injured their leg and needed it to get around. Thankfully, our office was accessible enough for them, due to a side ramp leading to the entrance. Through sheer luck, our borrowed meeting room was close to the elevators, so the player didn't have to take the stairs or walk a great distance. However, we should have prepared for this, not just for injured players who are temporarily disabled, but also for players with permanent disabilities. We subsequently made sure to check if players had any accessibility requirements, so we could prepare for them beforehand. The same check also eventually included a dietary check for when we provided food during longer playtest sessions, so we could cater to players' dietary needs.

As useful as it was to recruit university students, once the demands for UX research escalated and our need to provide more studies and a wider variety of participants. If you recall sampling bias from the chapter on bad data, you might have already realised that recruiting university students works well initially, but if all you do is recruit university students, your sample is unlikely to be representative of your target audience, and the data will be skewed accordingly. For example, most university students who signed up generally had minimal spare cash, so their feedback on spending trended towards being frugal. Similarly, they had more free time, so their feedback towards large time sinks in games was more accepting compared to feedback from players who were employed.

WHAT IS A TIME SINK IN VIDEO GAMES?

Time sinks tend to be activities that take up a lot of the player's time, and are usually in the form of repetitive activities or activities that take a long time to complete.

Time sinks are usually implemented to keep players engaged for commercial reasons. For example, certain games, such as Massively Multiplayer Online (MMO) games, require a monthly subscription to keep playing. Because players pay not just for the game, but a recurring

subscription, there is an incentive from the developer/publisher to keep players playing (and paying) for as long as possible. This is done via a mix of interesting new content (story or gameplay) and time sinks that keep players busy. An example of this in MMOs would be completing quests that recur daily so players can increase their characters' reputation with an in-game faction, with reputation milestones linked to in-game rewards. These "reputation grinds" can take up a great deal of time, requiring weeks or months of the players' time to max out.

Another example would be games which have upcoming paid Downloadable Content (DLC). DLC is a way for developers to add content to a game after it has been launched, sometimes for free, but usually at a price. Players are more likely to purchase DLC if they are still playing the game when the DLC is released, so building time sinks into the game to keep them playing until the DLC launches can be profitable for companies.

So, while university students were a great source of players and feedback early on, they were not suitable in the long run. In addition – and this is something we learned over subsequent playtests – the process of recruiting players ourselves took up too much time. Step by step, it required the following:

- Contact the universities to advertise details of the upcoming playtest with students, usually via some internal university webpage.
- Going through each of the candidates' details to filter out unsuitable ones.
- Contacting suitable candidates about their availability and scheduling them in.
- Handling cancellations, no-shows, and ensuring we had replacement players to substitute in if needed.
- Dealing with on-the-day communications, for example if a player is late or lost.
- Sourcing and handling the voucher payments to players.

The above can be quite time-consuming, especially once the UX Research department was fully established, and we were fielding multiple requests

for research. This was exacerbated when we began running large-scale playtests involving 30+ players (earlier playtests had 5–8 players), as the scheduling and on-the-day communications tasks began to scale out of control in relation to our team size. We considered two possible options: hiring a full-time staff member for administration and recruitment or using an external agency. The next chapter will go into more detail on which option we went with, and why.

For now, we'll move on to the next playtest.

Third playtest

The development team were interested in not just friction points, but how players felt about the game. Did they enjoy it, was feature X engaging, did they use mechanic Y to overcome challenge Z as expected? Feedback like this is subjective, and so we would need to recruit more players to ensure the sample selected was as representative of our audience as possible.

Planning and recruitment

A general rule of thumb is that around 30 players is sufficient for sentiment-based feedback, as it is a good balance between minimising cost/time and having an acceptable sample size of players. We'll come back to sample sizes for large playtests later on, but if you're just starting out, 30 players is a decent enough sample if your selection process is suitable.

WHAT'S THE DRAWBACK OF A NON-REPRESENTATIVE SAMPLE?

If you've read the Bad Data chapter, you can skip this box. If you haven't, continue on!

A non-representative sample, especially with subjective opinions of what players like or dislike, can lead to bad data and worse decisions.

The example that comes to mind was a research report that was commissioned with an external agency. The top finding from that report was that 80% of players in one of our games favoured multiplayer, which started discussions between developers about how

to improve our multiplayer offerings significantly to take advantage of this. When looking at the sample of players recruited, something seemed off, as the age distribution reported was skewed in the opposite direction of our players' age distribution, which we knew from other sources.

I checked our analytics data, and less than 10% of players in the game actually engaged with the multiplayer mode, with the remaining 90% only playing the single player mode. Given this discrepancy and the misaligned age distribution, a possible reason was that the external agency had recruited a non-representative sample of our players, likely overrepresenting the multiplayer-engaged portion of our players significantly. Thankfully, with these findings and a query to the agency, we were able to clear up the mistake, and no resources were wasted based on bad data.

A point of note here: This doesn't mean that improving multiplayer would not have yielded positive results in either increased sales or sentiment. It simply meant that the data from our existing players did not suggest such an action would naturally lead to that outcome.

Back to the playtest; we would need around 30 players, but not all at once! Unless you are part of an extremely large organisation, the number of staff and the room size needed to host 30 players at once is quite daunting. Unless your game specifically needs 30 players to be playing at the same time, I'd suggest splitting up your playtest into multiple sessions.

Box: What if I do need to have X players at once, because my game has an X-player format?

If the number X is 15 or less, you can skip this box, as the following paragraphs will detail having at least 15 players per session.

If it is more, then I would assume your game has an online component allowing players to play with others over the internet. If your game is already available to the public (via alpha or beta playtests, open access, or it has already launched), I'd suggest recruiting a number of players that fits your research lab size, then having them

matchmake online with the public players. These public players would not be part of your study, but your recruited players would still be able to provide feedback on the experience of playing as part of a large-scale multiplayer game. If your game is too early in development such that it cannot be played outside of the company, then you could try to time your playtest with internal team play sessions, so company staff would fill out the quota needed.

The number and size of your playtest sessions will vary depending on your resources and needs. I've run sessions with 5, 10, and 15 players at a time. The more players per session, the less sessions you'll need, and the faster you'll be able to collect and analyse the data, leading to faster insights and quicker decisions. But more players mean needing a larger research lab, and potentially needing at least one other room available if you intend to run multiple focus groups simultaneously. You'll also need at least two moderators even for 5-player sessions, and once you reach 10+ players per session, having a third moderator or an assistant helps make running the playtest much more comfortable.

For this playtest, we did not have a dedicated UX Research lab yet, so we borrowed an existing meeting area where developers would gather to playtest our games. While this took advantage of an existing resource with minimal cost, it had several drawbacks. We had to make sure that security concerns were dealt with; there could be no confidential material present in the meeting area (there was, in the form of notepads filled with notes and confidential data), the PCs had to be checked to ensure only the relevant game was on it, and not any other in-development games (there were). Finally, we had to make changes needed to enable a comfortable playtesting environment for external players, and make sure that whatever we changed was reverted to its original state once the playtest was over. Since we were borrowing the space, it would not have been professional nor good manners to return it in an altered state, leaving others to undo what we had changed.

This all meant that while we had an existing playtesting area with built-in infrastructure, there was a lot of work to do before and after the playtest. We had to sanitise the area to remove all confidential material; we had to uninstall or remove confidential data on 15 computers, and we had to install

OBS and set up the correct scenes for recording player footage for those computers. And once the playtest was over, we had to undo all of it.

Aside from borrowing a larger playtest area and recruiting more players, most of the preparation was identical to what has been described before; we synced with developers to understand what mattered most to them in terms of data, collaborated with them to ensure they were aware of the methodology used, took on their feedback, and familiarised ourselves with the game as much as possible. (More on familiarisation in the next chapter.)

Observing strategically during the playtest

With 15 players per session, all playing at once, it would be impossible to observe everything they did or experienced during the playtest. We had recordings for that if needed, but knowing everything we could about the slice of the game they were going to play was important, because it meant we could observe players strategically. If a player was about to experience a focal section or had unlocked a game mechanic that the developers wanted to know about, we wanted to be able to identify this at a glance and observe that player accordingly. Since each player plays at their own pace, it was possible to observe most, though never all, of players experiencing key parts of the game. The recordings served as backup, in case we missed anything, but it saved time to be able to observe players strategically and take notes accordingly for future data analysis. The game also had some basic data hooks coded in, so some player actions were being sent to our database. During planning and familiarisation, we wrote data queries to fetch the data we needed – what key actions players carried out, how long it took, and so on, again reducing our observation and note-taking workload during the playtest.

Feedback

For the feedback portion of the data, we planned to gather that in two ways. The first was a survey that players could answer after fulfilling certain requirements; either completing the slice of the game that stakeholders needed to know about, or when they ran out of time. The survey initially used questions that were open-ended, before drilling down into more specific questions about their experience with certain mechanics, with all questions discussed, iterated, and agreed upon with the development team

beforehand. The second method involved focus groups to get in-depth answers on the most important topics.

WHAT IS A FOCUS GROUP?

A focus group involves questioning a group of players (generally 5–10) about their experience. Like an interview, a focus group can be used to obtain detailed information from players by asking ad-hoc follow-up questions based on their answers, something a survey cannot do. A focus group allows you to gain this information from players more quickly than individual interviews would, as well as explore their shared experiences and social dynamics (more important in team-based games).

Focus groups aren't without drawbacks; because they will involve multiple players with potentially differing personalities, some may be more shy to speak up, while others may be more dominant. To help with this, the moderator should establish ground rules from the start; feedback from everyone is welcome, one speaker at a time, no interrupting others, let everyone have a turn. If a player is dominating the conversation, interrupting others, or challenging their opinions, it is the moderator's job to curtail this gently, to ensure that other players feel safe to speak, but also to make sure that the dominant player doesn't feel like they are being silenced.

To facilitate focus groups, we needed two large rooms so we could split the players into groups of 8 and 7. While the borrowed playtest room could have fit both groups, it would have meant a very noisy session where the two groups of players could hear, and potentially talk over, each other. We booked and set up another meeting room to host the second group of players. (As with the borrowed playtest area, we had to make sure the booked meeting room was suitable for external players to be in, and revert it after the playtest was over).

We had one moderator for each focus group, and a third person available to help out where needed. For example, the third person could help out if someone in one of the focus groups had to go to the lavatory, or run

shutdown tasks on the now-unused computers, or assist a moderator with note-taking.

What went well?

As with the earlier playtests, preparation and rehearsal helped ensure the smooth running of this large-scale playtest. Having a third person around to help out with miscellaneous tasks also meant that the two moderators could focus solely on the playtest and collecting data.

The analytics data collected aided significantly with analysis and reporting. In playtests where we had one player per session, we could observe a player's every action and take notes, but in a 15-player playtest, this was impossible. The analytics data reduced the time spent watching the video footage, since we could refer to the query's output to sanity check what tasks the players had completed, their success rate, and the completion time. We could also use the analytics data in the report to summarise the performance of 30 players, what percentage completed the slice of the game being tested, what sections took the longest to complete, and so on, without having to refer to our notes.

The two methods we used for feedback, surveys and focus groups, were used for their strengths and to complement each other's data. The surveys contained a mix of understanding-based questions for usability and Likert scale questions to get an idea of their sentiment towards the game. Surveys, if crafted well, allow for players to answer multiple questions in a span of minutes, allowing you to get data on a variety of topics quickly.

LIKERT SCALE QUESTIONS

Likert scale questions are used to measure a player's levels of agreement or attitude towards various topics. A Likert scale question usually has 5 or 7 answers, with the middle option being a neutral answer, and the remaining answers being mirrors of each other. For example, in a 5-point Likert scale question asking about how much a player agrees or disagrees with a statement, the answers would be:

Strongly disagree

Disagree

Neither agree nor disagree

Agree

Strongly agree

The goal of mirroring is to ensure that players can equally select negative and positive answers, depending on their experience, or the neutral option if they did not feel strongly about it.

Likert scale questions can also have an additional answer outside of the scale, to give players who have no opinion (rather than being neutral) a selection that fits how they feel. The additional answer would be something like “I’m not sure” or “N/A”.

In comparison, focus groups let you dive deeper into the data by speaking to multiple players at once, at the cost of requiring more time for less questions than a survey. We used surveys for broad data across multiple topics, and focus groups for detailed data on the topics that the development team cared the most about.

What went wrong/what did we learn and iterate on?

To enable playtesting, a computer has to have the correct build of the game installed, OBS settings and scenes configured, video drivers updated for stability, and a bunch of other technical tasks. Each task on its own required a small amount of time, but all of them together, and on 15 computers, was very time consuming to set up, test, and sign off as being playtest-worthy. We eventually developed a program to automate most of it, using a control computer to run all of those tasks on every other playtest computer at once.

As mentioned earlier, because we borrowed an internal playtest area, we had to revert the computers once the playtest was over. This only took a few hours, but that was time taken away from analysis and insights, and when repeated across multiple playtests and multiple borrowed rooms, the time spent on this starts to add up. We eventually did have our own UX Research playtest lab, but while it was being constructed, we borrowed the internal playtest room a few more times. During this time, we discovered another friction point beyond setup and reverting; the room was in high demand, so scheduling playtests became difficult, especially since large-scale playtests needed 2–4 days for setup, playtesting, and reversion.

We also underestimated how long the focus groups would take, due lively ongoing discussions between players, and the focus groups overran their allotted time. We had thankfully budgeted extra time just in case, so from the players' perspective, everything was on schedule. Later, we discussed this problem and came to the conclusion that we had asked too many non-essential questions in the focus groups. From then on, we made sure to plan focus groups around having three essential open-ended questions that needed to be asked, with several optional less important questions that could be cut if we were running short on time. We also needed to be stricter with moderation, but that would come with time and experience.

So far, this chapter has run through three of the earliest playtests as examples, using mistakes and learnings thereafter as examples of what to do and what not to do. Now we'll look at the processes instead, a cumulation of learning from others (books, videos, articles) and from experience.

9 UX research methodology

DOI: 10.1201/9781003735335-9

SOME PROCEDURES IN DETAIL

This section will go over some of the UX research procedures in greater detail, going through methodology rather than examples.

LAB SETUP

Note: If you aren't planning to run playtests, but use surveys or interviews or focus groups, then you won't need a playtest lab. Similarly, if you are planning to only test remotely or use agencies to handle the logistics of playtesting, there is no need to allocate space and budget for a physical playtest area. Services like Parsec (for PC), or PlaytestCloud (for mobile phones) can help with remote testing, allowing you to view your players' screens completely remotely as they play.

When we first submitted the designs for the permanent UX Research lab, it was based on what we had learned over a year of running playtests using ad-hoc meeting rooms and borrowed playtest areas, plus our projection of what research needs would be over the next few years based on discussions with stakeholders.

We were receiving multiple requests for large-scale playtesting, so we needed a room that would be able to handle at least 15 players at a time. While we would sometimes test with less players at once (8–10), having the larger space and the capability to not use it is better than a smaller space and no capability to expand. We had a lot of games in development, so we also needed a smaller room for parallel playtesting when the large room was

occupied. The smaller room would be used primarily for one-player sessions and a venue for the second focus group in large playtests. Finally, while a good portion of our playtesting was on PC, we did test with consoles (Playstation, Xbox, Switch) from time to time, so we could repurpose the smaller room with a living room setup as needed.

Finally, we found that hovering behind players or watching from a few feet away could be unnerving for them. This was unavoidable when we had to borrow a space for playtesting, but we wanted a better environment for ourselves and our players. To that end, we planned two different solutions, one physical, the other remote. The physical solution was to have an observation room in close proximity to the other two playtesting rooms. Cables under the floorboards connected the PCs from the two playtesting rooms to a multi-TV setup in the observation room. This allowed us to observe players engaging with our games without having to hover over them, yet still be close enough to intervene or help as needed. The observation room had an extra bonus; developers who were interested in watching the playtest live could join researchers in the observation room, where they could discuss between themselves and with researchers on what they were seeing without any danger of players overhearing.

WHAT ABOUT A ONE-WAY MIRROR?

Whenever I bring up our observation room in conversations with people new to UX research, one of the first things they inevitably ask is, “Does it have a one-way mirror? Can I watch players without them knowing?”

I always answer the second question first; yes, you can watch players without them knowing. Players not knowing they are being watched is ideal, since we don’t want them to be self-conscious with the knowledge that multiple people are watching them play. We enable watching them play our games using a combination of OBS streaming their gameplay footage and webcams recording their faces and body language.

My answer to the one-way mirror question is always the same; there are multiple difficulties involved that aren’t worth the effort to solve. One-way mirrors aren’t a product of magic, but science, and require

specific conditions to function. For them to work, the room being viewed must be brightly lit, while the observation room must be dark, because one-way mirrors still allow light through in both directions. This creates a risk that any accidental bright lights in the observation room (phones going off, someone entering the room) would reveal observers to players, even if only briefly, unnerving players when the giant mirror on the wall temporarily becomes a window.

Another issue is that for a one-way mirror to work, the observation room must be adjacent to the playtest room, limiting our options with regards to infrastructure and room selection. Finally, even if the two difficulties above could be resolved, a one-way mirror would only allow observers to view, at best, half of the participants in any standard room. For example, the large playtest room format we settled on had PC units lined around the perimeter of the room, so any large one-way mirror would only be able to view players from across the room; the ones closest to the mirror would be blocked by the PC monitors. In theory, this would be possible with an unorthodox room structure; a very long room could have many PCs lined up against the wall, with the one-way mirror at their backs. However, it would be very strange to have a room like that available in any office complex, and the purpose of the mirror would likely be obvious to players.

This observation room solution was a bonus; had the space not been available, we had an alternate remote solution, as OBS can record and stream footage. We asked IT to set up a server allowing OBS clients on each PC to stream footage to that server, allowing researchers and other staff to watch players engaging with the game remotely from their work PCs. While this was a useful alternative that did not require physical space, it had the drawback that our work desks were some distance away from the playtest rooms, so in the event that a player was stuck and we had to intervene, we could not do so quickly. We also lost the informal capability to sit down with developers in the same room, observe players together, and discuss the issues and findings we were seeing together.

In the end, we managed to secure three rooms: a large playtest room, the observation room, and a smaller playtest room that did triple-duty as a focus group room and a living room for console testing, all within close proximity

of each other. We also took into account the location of said rooms with our proposal, as you'll see below.

This will vary depending on the company, but at the time, we had quite a few unannounced games in development. While players recruited for playtests did sign NDAs prior to arrival, there were concerns that players would see these unannounced games while in-transit to the playtest room. Walking across floors where the development team were working was considered a risk, since PC screens would likely show confidential content, artists would have physical models from licensed franchises as inspiration, and the brand team would have mockups on boards. Any curious player might notice these, potentially leading to a leak of a game that was years away from being announced.

ROUNDABOUT ROUTES TO AVOID LEAKS

In our first year experimenting with UX Research, there was a strong concern of players seeing games they were not supposed to see, such that we were requested to avoid taking players across any floors where the development teams were working. This was possible through a bit of a roundabout route to get to the borrowed playtest room on the second floor; we took players from the front door through the IT section to the far end of the building, up the rear stairs, then made our way back partway through the building to reach the room. A little awkward, but we were able to distract players from the strange route through small talk.

To remove this risk and quell fears, we requested that any rooms allocated for the UX Research labs would be ones situated away from the development floors and easily accessible from the waiting area.

The waiting area is important because we needed to give players a place to wait comfortably. Players may need to wait because they arrive early (this happens quite often) or because they arrive late and have to wait anyway, as moderators would be busy onboarding the early and on-time players, and would welcome the late players once the rest were settled in and playing. It's unlikely that players would be allowed to walk around in

your company building unescorted, and so all players (early, on-time, or late) need a waiting area so they can be escorted to the playtest area when appropriate.

We used our reception area for this, since there was usually a receptionist waiting there. We would inform the receptionist about the upcoming schedule of playtests and how many players would arrive, so the receptionist could be prepared to let them in and alert us. The waiting area doesn't have to be your reception area; any area where players can sit comfortably for up to 30 minutes will do. If your reception area is too small, you could escort players to a pre-booked meeting room temporarily and have them wait there, something we did when the reception area was under renovation. For playtests where you only have one player per session, the waiting area may not be needed. Simply prepare the playtest room ahead of time, and if the player arrives early and is willing, you can start the session immediately.

Why not have players wait in the playtest area? That is also an option if all else fails, although there are some problems with this. As mentioned before, players will arrive at uneven times, and will generally do so one by one, meaning that you'd have to constantly go back and forth from the lab to the building entrance to escort players one at a time. Depending on how early the first player is – and I've had some players arrive over an hour early – you might not be fully prepared with lab setup and need time to perform final checks, so having the earliest player in the lab at the same time can be problematic. It's also preferable to onboard as many players as possible into the playtest at once, rather than give an introduction speech up to 15 times. Finally, if players need to start playing at roughly the same time (a multiplayer game requires this, for example), then having them waiting in the playtest room can be boring, as they won't be able to play the game or interact with any of their devices to pass the time.

Why won't players have their devices? While they may have signed NDAs, as discussed earlier, legal actions in response to a player breaking their NDA are undesirable, and it's best to think of NDAs as a preventive measure. One way to enhance security is to request that players not bring their phones or smartwatches or any other device that is capable of recording into the playtest room. Saying "I played this unannounced game" leaves room for doubt (no evidence), having footage of said game that can be uploaded to social media is much more conclusive. Stakeholders can

have concerns about confidential game information leaking, and letting someone record your game in development and publish that on social media will justify those concerns, leading to a lack of trust.

Obviously, we don't want players to have to worry about their devices, so we had a set of lockers placed outside the UX research lab rooms. Players could store their devices in individual lockers and keep the key, secure in the knowledge that they alone could open their locker. As a bonus, players will often bring satchels or backpacks or other personal belongings, and they are free to store all their stuff without risk of it going missing, or worse, stolen by another player, something which I worried about but thankfully have never experienced.

So, we had the playtest rooms and supporting areas all planned out. When planning the interior design and infrastructure of each playtest room, we were heavily informed by our earlier efforts borrowing other rooms as ad-hoc playtest areas. Air conditioning had been a problem when we needed to fit 15 players, 2–3 researchers, and 15 PCs in a single borrowed room, all generating heat in a room that was not set up for that purpose. After a couple of hours of play, the borrowed room became stuffy and hot, necessitating that we keep the door open for additional ventilation. The communal dining area was nearby, meaning that developers chatting and potentially talking about confidential information could be overheard by players if they took off their headphones. If the playtest overlapped lunchtime, the noise from the dining area could be overpowering when the door was left open for ventilation. Overall, not a great environment for playtesting, and a reason why we requested the playtest rooms be situated away from employee communal areas as well as development floors. To solve the air conditioning issue, we made sure that each of the rooms had sufficiently powerful air conditioners, and windows that could be opened for additional ventilation.

Dimmable lights were also very useful for ensuring that the testing environment was as bright or dim as needed. For example, when we were playtesting a horror game, we could dim the lighting just enough to make it more atmospheric for the players, without compromising their vision of their surroundings. As an accessibility feature, it was also useful for players who had trouble with bright lights, allowing us to dim the room to suit their comfort level.

For seating, we knew players might be sitting down for several hours at a time, so they needed comfortable seating. We provided adjustable chairs in both playtest rooms, so that players could change the height, tilt, and armrests to their liking. For the living room arrangement in the small room, we made sure to provide comfortable sofas.

The living room setup mimicked the real thing as much as possible, but we also took care to ensure everything was lightweight and adjustable. If players needed to be closer or further from the screen, based on their own preferred setup, we could easily shift the furniture around.

The large playtest area could fit a total of 18 players if needed, with chairs and desks positioned in a perimeter around the room. Each desk had partitions in between them, so players could be immersed in their gaming experience without being distracted by what other players were doing on their screens. The large playtest room also had amenities that helped researchers run playtests more smoothly. During large playtests with 15-player sessions, researchers would walk around the room, ready to assist players if needed (if they needed a bathroom break, if they needed to ask for assistance, etc.) and to observe their playthroughs. We had two long ottomans in the centre of the room, so if researchers needed a break from standing, they could sit down and still observe several players' screens at once.

In the third playtest example detailed in the previous chapter, I mentioned learning about the lengthy time required to set up multiple PCs at once. Our setup process preparation before every playtest involved this checklist:

- Ensure graphics card drivers are updated.
- Obtain game build via Steam, Perforce, or other means.
- Check that the game build runs.
- Revert game build to first-time user state, if required (useful for testing FTUE).
- Check that headphones, mic, and web cams work, and that OBS detects all inputs/outputs.
- Check that OBS settings are correct, and that streaming and recording work.
- If the players need to start from a certain point in the game past the start, ensure the correct save file is present and can be loaded without

issue.
- If players need to fill in a survey, ensure that it is pre-loaded, so players can easily alt-tab from the game to said survey when needed. Using a private browser
- Ensure that eye tracking devices work.

And our wrap-up process, both between sessions and after all sessions are concluded:

- Ensure that OBS recording/streaming has been stopped.
- Ensure that the game is terminated.
- Copy over video files to relevant secure storage (bearing in mind GDPR).
- Once all video files have been copied over, delete the files on the playtest PC to ensure that personal player data is not being stored in multiple locations.
- Revert the game build to first-time user state, if required.
- Delete all save files so players start fresh, or revert to the correct save file if they need to start at a later point in the game.
- If surveys are being used, wipe the browser cache, so auto-fill does not occur for players in subsequent sessions. (The auto-fill setting should be disabled, but mistakes can happen. You can also put the browser in private mode instead.)
- If all sessions have been concluded, remove the game build.

Each step on its own takes anywhere from half a minute to a few minutes to carry out. All the steps combined, across 15 computers, made for a lot of repeated work, and even with 2–3 researchers splitting the duties, it could take a couple of hours for them to complete all tasks. To resolve this, a colleague wrote up a program that allowed a single PC (the control PC) to send commands to the other PCs in the room (the clients). By doing this, a researcher could use the control PC to issue a launch game command to all the clients, check that it had worked, then issue a terminate game command to all clients, condensing 30 tasks into two, without having to move from PC to PC repeatedly. When considering 15 PCs and multiple sessions and multiple tasks, this program saved us dozens of hours of work across multiple playtests.

We also made use of the control PC during the playtests themselves. Prior to having this functionality, whenever a player's game crashed and had to be reloaded, we would have to go to the player's PC, start up the game again, and restore their game status to roughly where they had crashed, if possible. This meant the player had to get out of their chair and stand back, a somewhat disruptive process. With the control PC, we could simply inform the player about what we were going to do, then reboot the game and carry out other pre-scripted actions needed, without the player or researchers having to move about.

Several of the PCs also had eye-tracking devices attached, which allowed us to track where a player's gaze was during gameplay. We only had a few of these, as the development licenses were expensive. It also wasn't worth having them on every PC in the large playtest lab; the eye trackers were much better in single-player sessions, where researchers could observe the entirety of a player's experience. We didn't want players to be able to see the eye tracking icon, since that would have been a distraction, but OBS has options for this; you can hide the eye tracking overlay from the player and have OBS record it as an element overlapping your game recording.

EYE TRACKING IN EFFECT

One of the games we tested had a new game mechanic, which allowed players to engage in duels against the enemy during a battle. Players were notified about these duels through a UI panel that appeared at the top of the screen, with an option to accept or reject the duel request. Multiple players missed this notification, with some only noticing it just as the request timed out. We had eye tracking enabled for this playtest and could see what players were looking at, or in this case, what they were not looking at.

In all cases, when the duel request appeared, players were busy with the battle, their gaze flickering back and forth from their units to the enemy's units, as they gave commands and watched the battle play out. For some players, the eye tracker indicator either never went near the duel request panel, and for others, it hovered over the panel just as the timer reached zero, too late for those players to react. This footage,

combined with player feedback from the subsequent interviews, made a compelling combination of behavioural and attitudinal data showcasing the usability issue.

The developers didn't want the duel notification panel to be distracting, particularly when players were engaged with the battle, and noted that some players did eventually notice the panel. Their solution was to extend how long the duel notification panel lingered and increase the timer to allow players more time to make a decision.

Because we didn't use the eye trackers for every playtest, we needed somewhere to store them when not in use. For this, we had a storage cupboard in the large playtest room, allowing us to store spare tools and peripherals. This cupboard was useful not just for the eye trackers; you don't know when something might break down or malfunction, so having spare mice, keyboards, controllers, web cams, and headphones in close proximity is worthwhile, rather than having to rush off during a playtest session to your IT department for replacements.

Finally, all rooms had potted plants to add a bit of life, as well as framed pictures of various video game consoles throughout the decades.

A lot of what has been discussed was pre-COVID, with a lot of changes during that period, and some after. The COVID-19 pandemic was an interesting time for us, as we had to develop our remote testing capability and abandon the physical playtest lab for over a year, since lockdown meant everyone was working from home. To enable remote testing, we set up virtual computers in the cloud, using Amazon Web Services, and uploaded game builds to them. Using a tool called Parsec, players could connect from their own computers to these cloud computers and play games in development from the comfort of their own home. This let us adhere to lockdown rules and allowed us to conduct playtests without having to use the playtest lab. As a bonus, since we could spool up cloud computers in any geographical region, this also expanded our pool of potential players.

Those were the benefits of our remote testing solution, but there was also a severe drawback. While the game builds on the cloud computers were protected (players had access to play the game, but were prevented from downloading any files from the cloud computer), the concerns about players leaking information about our games intensified, since players would no

longer be playtesting under the watchful eye of researchers in the lab. Instead, they'd be playing from home, potentially taking screenshots or streaming their footage onto social media. Because of this, once the pandemic was over, we reverted back to using only the physical playtest lab.

Even when the lockdown was over, we had to consider some new precautions. What would we do if a player brought COVID into the studio, or if we infected a player? We had to make some adjustments; first, when recruiting players, they had to confirm that they were not sick. No COVID test proof was required, but we informed players that we reserved the right to turn them away if they showed symptoms. This was to protect the other players as well as ourselves, and the same policy applied to our researchers. We also took measures to make the playtest areas feel safe for players. Surfaces were wiped down before every session, and there were hand gels available for every player to use. Rather than bring in the maximum number of players possible, we aimed to bring in half of that, allowing for empty seats between players for additional safety, at the cost of needing more sessions to fulfil player sample requirements.

Planning

When conducting research, the first and most important action you can take is to understand the problem, the needs of your stakeholders, who are usually a sub-team owning the feature or section you are testing, but they can also be the entire development team, marketing, C-level execs, and so on. Any insights from your research will be considered and actioned by your stakeholders, so there needs to be alignment from both sides.

From your side, it is about understanding what insights they need. What part of the game (if not the whole) do they want feedback on? What concerns do they have? What questions can be asked to quell or validate their concerns? The best research isn't a "ask every question possible, catch-all study", but focused on what matters and what leads to impactful decisions for the game and for the stakeholders.

From the stakeholders' side, it's about setting expectations, and what is possible with research. If the game is extremely early in development, there are usually no visual effects or sound effects, and the player's journey is fragmented due to multiple non-functional sections. In this scenario, asking

about the user experience is going to yield fragmented results on a non-representative flow of the game due to its state of incompleteness. However, testing usability for parts of the game that are functional can lead to viable and actionable results.

At the heart of this alignment, stakeholders and researchers should be satisfied with how the research study will be conducted. The results of UX research studies can sometimes be surprising (pleasantly and unpleasantly), but if all parties are aligned on the methodology, then results are more likely to be considered, accepted, and actioned. If stakeholders don't agree on the methodology used, then any results that are not to their liking (potentially due to confirmation bias) will likely be rejected. This is not to say that researchers should carry out research studies with specific methodology just to please stakeholders, but rather that effort be made to inform stakeholders on why best-practice methods are being used, with the goal of reaching an understanding and compromise, from both sides.

Even when everyone is aligned, it's helpful to keep stakeholders updated throughout the planning process. Sharing details of the playtest structure and questions that will be asked helps to keep them informed, while also allowing them to provide feedback, enabling iteration. Both transparency on processes and constant collaboration helped build trust and ensure alignment between UX research and stakeholders. These updates also work both ways; having a constant line of communication open means that if anything changes with the slice of the game being tested, researchers will find out sooner and be able to tweak the research plan accordingly.

As with data analytics requests, not every research request needs to be fulfilled. Sometimes, stakeholders will come to you with a data request that UX research is ill-suited for. Perhaps it's more about player behaviour that is already tracked, and the analytics team can answer it. Perhaps it's more about market research, or community feedback, and those teams already have the answer. Perhaps it is for UX research, but this is the wrong phase of development for it. The ability to direct data gathering and analysis to where it can be carried out most efficiently is another benefit gained from constant discussions with your stakeholders.

Part of the planning process also involves familiarising oneself with the game or the slice of the game being tested, as much as possible. As a researcher, you'll have to observe players interacting with your game and ask them questions about their experience. The ideal scenario would be one

where you yourself have experienced as much of the game being tested, so that there are no surprises; whatever the player sees and hears should be something you have already experienced multiple times. This ensures that you're able to ask informed questions based on your thorough knowledge and experience with the game, and understand at a glance what a player is trying to do. As a bonus, familiarisation also builds trust with the stakeholders; if you don't know or understand the game you are conducting research for, how can you be expected to talk to players about it?

Finally, the planning period also involves tech checks, playtest setup, and rehearsals. Tech checks are needed to ensure the build you receive for testing works on your PCs or consoles or devices without any issues. If there are issues, they can be flagged to the development team, and the playtest date altered if needed (the earlier you discover this, the better, and ideally before players are recruited). I've found that an efficient use of time is to combine familiarisation with the game and tech checks together; rather than play the game on your work PC, play it on one of the playtest lab PCs, solving two tasks at once.

Playtest setup involves ensuring that all the equipment and software you need to run a successful playtest are working smoothly. OBS checks, graphical driver updates, hardware checks, and so on, all of these need to be signed off as working as expected. Doing this early on means that fixing any issues found becomes a relaxed and carefree task. Doing it late, on the day before the playtest, or just before the playtest begins, means that fixing any issues becomes stressful and potentially unfixable in the time left. Rehearsals are simply end-to-end checks; you're going through every step of how the playtest would be run on the day, making sure you're as prepared as you can be.

Recruitment

Recruiting players to come in and playtest your game can require quite a bit of effort, due to a variety of factors. These factors can include:

Company location

This comes down to whether your company office is in a major city, which will naturally have a larger population. Outside of major cities, and

particularly in small towns, it can become much more difficult to source all the participants you need, especially if the player recruitment criteria are quite complex. In this case, higher compensation (including transportation costs) can help incentivise players to travel from afar to your offices. Higher compensation does come with a drawback (aside from the cost) in that it will increase the number of "bad faith" applicants seeking to join the playtest for the money, with these applicants entering misleading or incorrect information on the recruitment criteria survey try and be selected. To that end, it's best to include some validation questions in the screener survey to help weed out these "bad faith" applicants; more on that in the next paragraph.

Player recruitment criteria

Recruitment criteria can range from being extremely broad, like "We need players who play games", to being extremely specific, "We need players who have played at least 200 hours in four specific competitor games (each game, not across all games), and they must also have participated in esports for at least one of those games", the latter being an actual request I received. The more restrictive the criteria, the more difficult it will be to find suitable players, to the point where running the playtest might not be possible. Rather than reject the "200+ hour" request above outright, I went to the requestor to understand what they wanted to achieve, so we could work together to modify the recruitment criteria to be more feasible while also meeting their needs. They intended for the game they were working on to appeal to extremely competitive players, and they wanted to get feedback from very competitive players of other games. We were able to modify the request to be more feasible, whilst still maintaining the "very competitive aspect", since esports is not the only criteria for players to be competitive. We did this by relaxing the criteria of players needing to have 200+ hours in four different games, since a player can be competitive but still only play one game excessively, and by asking about their preferences that could indicate a desire to compete, such as interest in topping leaderboards, researching best strategies to win, and interest in esports.

These recruitment criteria can be applied to players through a screener survey, which will cover aspects such as what genre of games they play, how long they have played, what they are currently playing, what consoles

or devices they own, and so on. As mentioned earlier, some applicants will fill in the survey in bad faith, usually for monetary reasons. To deal with this, several trick questions and/or answers can be inserted into the survey to catch out these bad-faith applicants, and those who are not paying attention. Such questions can be as simple as “This is an attention check question. Please select the answer ‘4’ from the list below” to catch out players who are not paying attention. Others can include trick answers, such as inventing an obviously fake game (be sure there is no actual game that a player might mistake for your fake one, invent something outlandishly fake if possible) and asking players if they have played that fake game.

Finally, some obvious patterns in the data can be used to rule out “bad faith” applicants. For example, we had multiple questions about playtime in various competitor games in a genre we wanted to test, with one of the answers being “200+ hours played”. Early on in my career as a UXR researcher, we had a player select this option for every possible competitor game. When this player showed up to the playtest, it was extremely clear from his behaviour that he had never played any game in this genre before; he did not understand basic mechanics common to games in that genre, and struggled to carry out basic actions. His data had to be discarded entirely, since the goal of the playtest was to obtain information from experienced players. Even using him as a single source of beginner experience would not have been useful; his answers were monosyllabic, and it was clear he wanted to leave as soon as he could, fee in hand. Since then, we have excluded any player who presents us with such extremes across all questions; the likelihood that they have that kind of gaming experience is low, and the chance of bad faith is high, especially since these players tend to answer the trick questions incorrectly as well.

Playtest length and timeslot flexibility

Players can be university students (we did not recruit under-18s, due to the combination of extra measures needed to do so and our games being mainly by adults), employed, unemployed, and so on. Depending on other factors described above, the length and timing of your playtest might affect who can join. Think about running a playtest on a weekday, lasting for over 6 hours. Those in employment have to take time off work, students need to have free time in their schedule (or cut classes), parents have to make sure

they are back in time to pick up their children from school, and so on, creating restrictions that might limit your ability to recruit. You could run the playtest on weekends, bypassing some of the problems above, but this comes with its own problems with regards to a physical lab, since the building you work in might not be open, or if it is, there may be legal issues with bringing in external players into the playtest lab, since there may be legal requirements around needing a fire marshal and staff trained in first aid on hand, in case of emergency.

When you need to fully observe and question players for as much detail as possible, playtests that only use one player per session are usually the best choice. But what if other factors are a higher priority? For example, we've run research studies on multiplayer games that required teams of three, and as such, the playtests needed three players per session. We've also run playtests aimed at getting subjective feedback, which requires a reasonable sample of players (usually around 30) to ensure the feedback is representative of your target audience.

Recruiting more than one player per playtest session introduces some interesting challenges. For example, if you have only one player per session, one moderator is acceptable. The singular moderator can greet the player, moderate the playtest, and ensure that the player is escorted at all times, for security and safety concerns. Once you have two players, what happens then? What if they arrive at different times? What if during the playtest, one needs to go to the lavatory and needs to be escorted; who keeps an eye on the other player? A single moderator can moderate and conduct a focus group, but what if you need to conduct 1:1 interviews? Do you keep one player waiting? Where? Do they get to listen to the first player's interview, which may influence their feedback?

Now, reconsider the questions above, but with ten players in a playtest session instead of two. The simple answer to all of the questions is: you'll need more moderators. Thankfully, the number of moderators needed doesn't scale with the number of players. I've found that two moderators can reasonably handle 2–15 players per session, although as detailed earlier, having an extra person as a third moderator or an assistant makes the entire process much more comfortable, especially at the upper end of that range of players.

Multiple moderators handle the same tasks a single moderator does; greeting players, briefing on the playtest, observing and dealing with the

occasional request, and handling feedback, whether that be via interviews or focus groups. One moderator can also cover for the other in case of temporary absence (escorting players to the lavatory, their own call of nature), ensuring that the players always have at least one moderator available to call upon.

In the previous chapter, I covered recruiting other team members and university students as players, which was useful in the first year of building up UX Research as a resource for the company. Over time, this became less and less viable for two reasons.

The first was that university students, while a good resource for dipping our toes into UX Research, were not a representative sample for any playerbase. University students, by and large, come from a narrow age group, are not in full-time employment, tend to be unmarried, and have less disposable income than someone who is employed. There's nothing wrong with any of that, but unless your game specifically targets university students as an audience, such a recruitment pool will not provide you with a full or representative picture of how your players behave and feel.

The second is that manually recruiting via universities was starting to take up a significant portion of our time. While that time was useful to enable UX Research, it was another hefty task on top of syncing and planning with stakeholders, moderating the playtests, analysing the data, writing up the report, and presenting the results. It also created a distraction or delay when we were in the middle of running a playtest; we could either handle recruitment for a future playtest during an existing 2–3-day playtest (distraction) or put it off until after the existing playtest was over (delay). Neither of these options were appealing, especially when considering the workload would likely increase once we expanded the audience beyond university students. I sought out advice from other UX researchers in other companies and learned that they either had a full-time staff member handling recruitment, or used an external agency. We didn't have enough budget for a full-time staff member just for this, so we looked into external recruitment agencies.

The recruitment agencies we've used in the past will advertise to recruit players, handle the screener survey (with questions we provide) and subsequent filtering, contact prospective players to arrange scheduling, and generally be a point of contact for players. That last task is very helpful, since players may be late, get lost, or need to make a general inquiry on the

day of the playtest, and it can be difficult for researchers to take on these ad-hoc calls when they are preparing for or running the playtest. Using recruitment agencies also meant we could switch away from using Amazon vouchers to compensate players and not run afoul of employment laws, since we would pay the agency, who would pay the players and make their own legal arrangements.

Using a recruitment agency still involved work from our side, as we had to provide them with recruitment criteria, work with the agency to ensure we were getting representative players, provide digital NDA and GDPR links (plus check that players had signed), and so on. I estimated that using a recruitment agency reduced the time we spent on recruiting players by about 80%, and more importantly reduced our costs, since it was cheaper to use a specialised recruitment agency (which is efficient at what it does) than to use staff.

While cheaper, this didn't come without a cost; we would still be paying player fees (via the agency) and also adding agency fees on top. We ran a cost-benefit analysis of the time saved using recruitment agencies against having a dedicated staff member conduct the work, and having the agency conduct the recruitment work was more cost-effective given our workload at the time. Had our workload increased to the point where we needed to run a research study more than once a week, it would have been more cost-effective to hire someone to work full-time on recruitment.

WHAT ABOUT USING AGENCIES FOR FULL UX RESEARCH WORK?

Speaking of cost-benefit analysis when using an agency, there are agencies that will conduct full UX research work on your behalf. These companies will plan, recruit, run the study, analyse the data, and provide you with a report and briefing, essentially handling the majority of the work, while working with you to ensure they understand your needs so they can plan and run the study accordingly.

These agencies can be useful, even if you already have a UX Research team; I've had to use them when the team was fully occupied, and we could not fulfil additional requests in the short-term. If you don't have UX researchers, or have no desire or time to run research

yourself, using agencies can be a good way to get feedback from your players.

Feedback

There are a variety of things to consider when obtaining feedback from players. This section will first detail generic tips common across all feedback gathering methods, then dive into specific methods, such as focus groups and think-aloud protocol.

What people do matters more than what they say. Actions speak louder than words. Deeds, not words. These sayings are as true in UX Research as they are in everyday life. Where possible, always try to accompany your attitudinal data with behavioural data. In an in-lab playtest, this means pairing their recorded actions in-game with their feedback. For a survey or a diary study, this would involve validating their feedback with analytics data, if possible. For example, if players in a diary study state that they play on a daily basis, and you have their user ids as part of the study, you can validate whether they do play every day. When you cannot pair their behavioural and attitudinal data, try to validate with aggregated player data or demographic data. For example, you can include a question in your survey to ask players about facts you already know, e.g. asking players if they engaged with your game's multiplayer mode, when you already know that 40% of your players do so. From here, you can compare the results of your survey respondents to see if they roughly match your players' behaviour. If they do, then you know you've at least recruited a representative sample with regards to multiplayer engagement, increasing your confidence in their answers.

Players, like people, are not good at predicting their future behaviour. It's why New Year's resolutions, where people set goals for positive change, such as exercising more, learning a new language, or writing a book, tend to fail more often than not. People get a positive boost from the act of setting the New Year's resolution, but when it comes to enacting those changes, factors like lack of time, hardship, and other unforeseen circumstances can get in the way. For this very reason, it's best to avoid starting questions with "Would you…", as this requires players to predict their future behaviour and provide their answer based on that unreliable prediction. From my

experience, asking “Would you…” has resulted in extremely optimistic or positive responses, particularly for questions like “Would you buy this game when it releases?” or “Would you play this game?”

Asking players if they understand a game mechanic isn’t ideal, because it can lead to simple “Yes/No” answers. In the case of “No”, you lose out on why they don’t understand. In the case of “Yes”, you may not be getting reliable data, as players might have an incorrect understanding of the game mechanic, while being unaware of their mistake. Instead of asking questions that lead to yes/no answers, it’s better to ask players to describe the mechanic or topic, allowing you to gauge their answer against the reality. From that context, you’re in a better position to ask follow-up questions that lead into the why.

When asking for feedback, be wary of social desirability bias, which occurs when players provide untruthful answers in order to be polite or present themselves favourably. This can occur in the form of pretending to like the game because they want to be polite and not offend you, over-reporting their income, or claiming to do positive things they would normally not do. The net effect of social desirability bias is that it can positively skew the feedback you get, whether through interviews, focus groups, surveys, or otherwise, unless you take measures to mitigate or dampen it.

Such measures include neutral wording in questions and answers to avoid the impression of “positive” or “correct” answers, or indirect questioning, which involves asking what friends or other people might think about a topic, rather than the player. In interviews or focus groups, this can include emphasising that there’s no need to hold back on providing negative feedback, as you are open to all feedback, positive or negative. If you don’t work on the game directly, adding that you won’t be offended due to having no personal stake in the game can help encourage players. For surveys, using anonymous surveys where players don’t have to provide any form of personal id such as name or email means they have less reason to paint themselves in a positive light.

Sharing your own views is also a bad idea, as it will likely skew the results since players may be influenced by you. In addition to neutral wording, a neutral stance is important. If players ask you for your opinion on a topic or game mechanic, it’s best to reflect the question back onto

them, stating that their opinion is what matters for helping to improve the game.

Focus groups

When moderating focus groups, ensuring that players feel like their voices are heard equally is important. Some players will be louder, more forceful, more outspoken, and others will be quieter, more shy, and more withdrawn. As a moderator, your job is to ensure an environment where everyone feels comfortable speaking. Setting the rules early on helps with this by stating that the goal is for everyone to be heard, so all participants are aware. As focus groups are by their nature made up of multiple players, someone will have to speak first. Rather than let it be organic (this usually leads to the outspoken players going first, repeatedly), it can be useful to have an order in which players speak.

Imagine five players seated in a semicircle, with you, the moderator, in the centre. When asking a question, you can direct it to the leftmost player, who will answer first, then transition from left to right, giving all players a chance to answer the question in order. With the next question, you reverse the order, starting with the rightmost player, then transition from right to left. By doing this, every player gets their chance to speak, and no player always speaks first. It's also good to state that you'll be doing this at the start of the focus group session, and why, so players understand the process.

One slight drawback of the above method is that some players will never go first (or last), and one player will always be in the middle in an odd-numbered group. I've run variants of this method, where instead of reversing the order, I cycle through the group. For the first question, the leftmost player answers first, followed by transitioning from left to right as with the above method. For the second question, the second leftmost player answers first, followed by the same transition, looping round and ending with the leftmost player answering last. For the third question, the third leftmost player answers first, and so on. This method has the benefit of giving an equal chance for all players to answer first, but has the drawback that it requires more cognitive load to keep track of which player should go first.

Think-aloud protocol

Depending on a variety of factors, you can also include a think-aloud protocol in your playtests, which is a fancy way of saying that you can ask players to narrate what they are doing or what they are thinking while they are playing. This method can be useful for understanding why players are doing or what they are feeling in the moment, without them later having to think back and remember during the survey, interview, or focus group period.

The factors that enable or prevent this method include player willingness to do so (players can be shy), the playtesting environment (our prototype playtest in an open office space did not allow for this, though our dedicated lab did), the number of players being tested at once (one player thinking aloud is fine, 15 players thinking aloud all at once would not be), and so on. And while it can be useful for the moderator to have the player narrate out loud and provide information in the moment, there are some drawbacks which have to be taken into consideration.

First, playing a game can involve heavy cognitive load, and while a player might be willing to narrate what they are doing, there is always the chance that doing so means they have to divide their attention between playing the game and narrating their inner thoughts, meaning a potentially worse experience, which in turn may affect the feedback they provide. Second, the act of narrating while playing a game is generally not how players play games (unless they are influencers), and players may play the game in a different way because of it. Third, some games aren't a good match with think-aloud protocol due to the additional cognitive load added. For example, fast-paced games like shooters, where the player may only have a split second to react to an enemy attacking them, do not allow time for players to consider what they should say. At the end of the day, the usage of the think-aloud protocol depends on the needs of your study and what you are aiming to get out of it.

Player suggestions

During interviews, focus groups, or any other qualitative feedback method, players may suggest solutions to problems they experience. You should always be receptive to these suggested solutions, as you always want to make sure players feel they are heard, especially when they are trying to be helpful. The tricky bit is that players rarely have a full understanding of

how all the interconnected parts of a game work together, and view the player experience solely through their own lens. Players focusing on their experience will tend to "solve" the symptom, rather than the problem. For example, if a tutorial is long and involved because the game is complicated, players who grasp the mechanics easily may grow bored and suggest the tutorial be made shorter, cutting out teaching sections for mechanics they understood easily. However, said mechanics may not be as easily understood by other players, who need those sections.

Feedback, but for you

Feedback on your findings can be as useful as player feedback to help improve your games. For any research studies you conduct and share, follow up with stakeholders. Find out what worked well, what didn't, what decisions were made, what weren't, and why. Perhaps your findings need more context or background on methodology used to reassure stakeholders. Perhaps they would like more recommendations on what to do, actions to take based on the insights. Perhaps they'd prefer the findings to be conveyed in a different way, through a presentation, a shorter report, an executive summary. Or perhaps the feedback is entirely positive, the findings and recommendations were effective – continue as you are!

Sometimes you'll have to do bespoke work as a result of feedback. I've had feedback that research reports were too long, with too many findings to action, from a team that was behind schedule and didn't have much spare time beyond essential work. We ended up distilling their reports down to micro-versions that only contained findings for the most important issues, based on the areas they were most worried about. While this meant many valid findings went unaddressed, it's better to get some player feedback actioned, rather than none at all.

10 UX research case studies

DOI: 10.1201/9781003735335-10

MORE UXR EXAMPLES

With the methodology from the previous chapter in mind, we'll go back into practical definitions and examples of UX research, but this time instead of going through them chronologically, we'll group them by what the goals were, starting with Usability.

USABILITY

Usability issues are issues that make it difficult or frustrating for users when engaging with a system, normally due to design flaws (intentional or otherwise) or lack of information presented. Usability issues are not based on user likes or dislikes, but rather on friction points that prevent or delay users from achieving their goals within a system, whether that goal is something as complex as completing a level in a video game or as simple as opening a door in real life.

Imagine walking up to a building with transparent glass doors. On either side of the door are two metal pull handles, with no hinges in sight. To open the door, do you pull or push? Without any clear indication, you have a 50/50 chance of getting wrong. The existence of handles on both sides is not going to prevent you from opening the door, but it can create a tiny friction point when you pull or push incorrectly and experience negative feedback.

Doors like this, where it isn't clear how to open them, are known as Norman doors, named after Don Norman, a huge advocate of user-centric

design. Norman doors are a fairly light usability issue, as they can cause some annoyance, but won't be a blocker. Does this mean we can leave them as they are?

Ideally, no. Users shouldn't have to guess how to open a door. Even if it's a minor annoyance, consider a Norman door that experiences heavy traffic, with say, a thousand people using it every day. At a 50/50 error rate, this means around five hundred people will experience momentary annoyance before opening the door the correct way, every day. Is that a good user experience? After all, people will get used to it, or put up with it, or ignore it. They might also wonder why the doors aren't designed in a more intuitive way. Why don't the doors have labels? Why do they have handles on both sides, if one side requires pushing to open and the other requires pulling?

Doors should easily convey how they are opened via good design principles or via signalling. Returning to the example, now imagine that the transparent glass doors have a pull handle on one side, and metal plate on the other. As a user can see both handle and plate at any given time, there is no ambiguity on how it should be opened, as long as the handle (pull) and plate (push) functionality is understood.

Labelling can also fix this. Returning to the original example of the glass doors with pull handles on both sides, now imagine that above each handle is a label. Your side reads: Push. Now you know exactly how to open the door from your side without having to make a 50/50 guess. On the other side, we can see that the label reads: Pull. No ambiguity whatsoever, at the cost of including some labels.

While each of these solutions does remove the friction, an even better solution would be to use multiple channels of information, as a design feature alone or labels alone may not be understood for a variety of reasons. Combining both solutions to provide two channels of information, a pull handle and a metal plate, both with appropriate labels, reduces the risk of friction even further. People with dyslexia or trouble reading can rely on the design feature, while those who aren't familiar with the design feature can read the labels.

Camera rotation

Those were examples of usability issues in the real world, what about within video games? Some time ago, I was working on a game franchise which was having trouble retaining new players who had never experienced those games before. Sentiment from playtests on previous entries in the franchise revealed that new players found the game overwhelming due to its complexity and difficult to get into.

Because of this finding, the development team wanted to significantly ramp up their efforts on onboarding new players for the sequel, and began work on the tutorial much earlier in development, with the goal of having it be an engaging experience that taught new players the basics of the game gradually, rather than dropping them in the deep end from the start.

The team wanted constant feedback from new players on the ongoing state of the tutorial. To make sure they were achieving their goals, the tutorial had to be informative, not overwhelming, and engage players through a strong narrative. To this end, we ran multiple playtests on the tutorial over several years, with the first being on an early development build that lacked polish and the narrative. The goal of this first playtest was to gauge any usability issues or problems with how mechanics were being taught to new players.

Players who went through the tutorial would start out in a small enclosed area, with only one direction to progress, north. The game's minimal narrative and instructions at the time would guide the player by telling them they had to go north, with occasional reminders if they failed to do so after some inactivity. As players progressed, parts of the user interface (UI) and game mechanics would be unveiled to them to lower the initial cognitive load required. During this first playtest, some players would progress north, up until a point in the tutorial where the ability to rotate the camera would unlock. Players viewed their avatar through a top-down view from an angle (an isometric view), and were able to move the camera, rotate it, pan it, and zoom in. By default, the camera was facing north, since that was the direction they needed to go.

Upon unlocking this capability, players would experiment with various camera movements, including rotation, the result being that some players unknowingly rotated the camera to point in the opposite direction from its original orientation. For these players, this meant that the camera was now facing south instead of north. Since the camera had been previously facing north, and there was neither a compass nor a map or mini-map to indicate

directions, players associated whatever lay ahead to be “north”. Players would then follow the original instructions to go north in order to progress, and in doing so, players who had inverted the camera’s orientation ended up going south instead, back the way they came. Upon reaching the initial area, which was enclosed apart from the path they had just backtracked, a few players realised their mistake and retraced their steps, while others were confused, as in their minds the camera was still pointing north, and the game, having detected that they were not progressing, would remind them to go north, which they could no longer do.

This small action, rotating the camera, in conjunction with the fixed nature of the early part of the tutorial, caused an inconvenience for some players who realised their error, and confusion for those who did not. Notably, at this point in the tutorial, there was no need for the camera controls to be unlocked, as players only needed to go north in order to progress, and there were no obstacles on the map that required camera rotation in order to view around them.

A simple solution here, which was implemented, was to not unlock camera rotation until it was needed much later in the tutorial, when the tutorial switched from being a linear path to a non-linear environment, or an open-world. In this new open-world environment, camera rotation was required to see around obstacles or terrain features, so unlocking the camera rotation would be meaningful to the player, as it would be associated with a problem that needed to be solved.

This is an example of how something as small as the timing of when to introduce camera rotation can cause usability issues for players, in conjunction with instructions that could not be followed, due to players being instructed to go north, with no way to validate which direction was north without relying on memory.

Unclear help messages

Games often use text to teach or guide players, and usability issues can occur when the text used isn’t as clear as possible. Players come from all walks of life, and can interpret words in different ways, based on their past experiences, particularly with other games. To compound this problem, developers tend to be very familiar with the features and functions of their game, so it is easy for them to read text describing game mechanics and

discern its meaning, regardless of whether it was ambiguous or not. As such, identifying a line of ambiguous text, especially if there are thousands of lines of text in your game, can be difficult for developers who work on the game for months or years on end.

An example of this was in a simulation game we tested, which involved constructing buildings and rooms. Players could then furnish a room as they wished, decorate it, and save each room once they had customised it to their hearts' content. Players could also press a button to cancel the room instead of saving it, if they weren't happy with the layout and wanted to start from the beginning. When cancelling the room, a text box would appear, informing players about this outcome, using the words "Are you sure you wish to undo?", with an accept and refuse button to continue or cancel the action.

During a playtest for this game, players would press the button to cancel the room, read the text that appeared, and then select accept, only to be visibly confused when the room was deleted. They would then redo the room, continue playing, and some time later repeat the same cancel action on another room, only to be visibly frustrated, with some muttering to themselves. During the interviews after, we asked about these moments, and players replied that they were surprised that the entire room had been deleted. Further questioning revealed that they had interpreted the "Are you sure you wish to undo" text to mean that it would undo the last action they had taken, rather than deleting the entire room, similar to using undo in a Word document. The room deletion was unintentional and frustrating on their part, especially when they had spent quite some time furnishing the room, and only wanted to undo their most recent action. And because the "undo" terminology was ingrained in players as being associated with undoing the last action, they would forget they had made this mistake after some time, and repeat the error, creating more frustration.

We detailed this usability issue by using video clips of players' behaviour and their feedback, along with analysis of the issue's impact on players, since it kept occurring, with players being frustrated at both the game and themselves. An initial recommendation was to change the word "undo" to "delete", to remove the ambiguous word association. The developers went one step further after reading the issue and watching the videos; they reasoned that even with the word changed to "delete", some players might interpret "delete" to refer to the last piece of furniture or

decoration within the room, rather than the room itself. They added a few more words to remove ambiguity altogether, in the form of the phrase “Are you sure you wish to delete this room and its contents?”

We ran another playtest on that game some time later, and followed up on this issue by checking if players experience the same issue when deleting a room (intentional or otherwise). The developers had done a good job fixing the issue, and we saw no occurrences of players mistakenly deleting their room when they wanted to undo their last action instead.

UI blocking

So far, I’ve listed simple solutions to simple usability issues in games. However, not all solutions can be as simple as the ones described above. Sometimes the solution to a usability issue may not be obvious, or a possible solution may cause more problems.

We playtested a puzzle game on phones, with touch controls to navigate and solve the puzzles. Because this game was not on PC, we had to make some adjustments to our setup. The phone screen and player taps could be streamed to a nearby PC and recorded, so we could keep gameplay footage. However, we felt it was important to have a recording of not just gameplay, but how players were interacting with the game.

Some smartphone games require the player to hold the phone with both hands, using one thumb to input directional commands and the other to tap on virtual buttons to jump, shoot, or dodge. This game, being a puzzle game, only required tapping and dragging puzzle elements, using either both hands or one.

Because of this less restrictive control scheme, players had a variety of ways to hold and interact with the screen, depending on their personal preference. Some would hold the phone in both hands and use their thumbs. Others would hold the phone with one hand and use the other for interactions. These different ways of holding and interacting meant that it was important to record how players interact with the phone as well as the game, as the next playtest example will show.

Mindful of how different players could interact with the phone, we made sure to recruit more players than we normally needed for a usability playtest, twelve instead of six.

HOW MANY PLAYERS SHOULD I RECRUIT FOR A USABILITY PLAYTEST?

There are studies that indicate five users in a playtest will experience around 80% of the common usability issues your game might have. And the more users you recruit for a usability playtest, the more diminishing returns start to kick in, with less and less unique issues found with each successive player. Over many years and many playtests, I've found both of these statements to be accurate.

For example, the first player you have in a usability test will surface multiple issues, some that are expected, and some that are not. The second will surface more new issues, but will also likely experience some of the same usability problems that the first user experienced. And so on with the third, the fourth, and the fifth. In my experience, on average, by the time you've reached the eighth player in a usability test, what you'll mostly observe are the same usability issues you've seen from those that came before. This isn't necessarily a bad thing, as it provides more data on how likely those issues are to be experienced, but it does have a fairly sizable drawback.

Because recruiting more players takes more time and money, there is a balance to be had between being thorough (more players means less risk of missing usability issues, more confidence in the data) and being expedient (less players means time needed to gather and analyse the data). Your ultimate goal is to gather data in order to make your game better for players, and the longer it takes to do that, the less effective this is, particularly if your stakeholders are blocked until they can get that data to make decisions.

In my experience, six players is a good amount to recruit for a usability test. This is because when recruiting, around 10% of your recruited players won't show up, for a variety of reasons. Some might fall ill, have a sudden change of plans, or simply change their mind. One of the more unique reasons I heard from a player was that they had spent the previous night searching for their dog, which had run off. Since the previous night was Guy Fawkes night in the UK, when fireworks are lit up for most of the evening, it was a believable story. As such, it is best to recruit with this factoid in mind; on average, 10% of your players won't show up. So if you want to recruit five players

for your playtest, increase that number by 10%, and round up, to cover for no-shows. If all players do show up, you'll have a bit more data for your playtest, even when accounting for diminishing returns.

We recruited double the number of players for this usability playtest, because we were aware that players might hold their phones in different ways, leading to varying experiences. To this end, we included an extra question in the screener survey, asking players how they preferred to hold their phones when interactions only required one hand. Based on these answers, we were able to select a mix of players who played games on their phone by holding it with both hands, and those who held it with one hand. While we could have skipped this question and left it to chance, that ran the risk of recruiting too many of one type of player, or in the worst case, only players of one type.

We had two camera setups to record how players interacted with their phones. The first involved a magnetic attachment that connected to the back of the phone, with a web camera protruding from it that could be pointed at the screen via an adjustable neck. Because the camera was connected to the phone, this setup had the benefit of enabling players to sit however they liked, on a chair or on a sofa, leaning back or forward, in whatever posture was comfortable for them. The attachment did weigh around 0.5 kilograms, meaning players could get fatigued over time due to the extra weight. The other setup involved a fixed camera on a tripod, angled downward. This added no extra weight, but had the drawback that players had to hold the phone so that the screen was in view of the camera, i.e. players could not lean back.

In the interest of player comfort, we explained the above to players at the start of the playtest and gave them the option of which setup they wanted to play with, with the option to switch partway through if they desired, or move away from the camera setup entirely, if both setups were too uncomfortable. While the latter was a good option to have, none of the players needed it.

Recording the phone screen and their hands turned out to be very useful, as we were able to use the footage to showcase an important usability issue with the game. The game had significant UI elements on the left side of the screen. Since these UI elements were used to convey important information

to players, being able to see them at all times was important. Most of the players who held the phone with one hand did so with their left (common for right-handed people), and in doing so the thumb of their left hand would partially obstruct some of the UI elements.

Players noticed this quickly and tried to adjust in various ways. Some would shift to a right-handed grip, but found this uncomfortable over time due to having to use their non-dominant hand to tap and swipe on the phone. Others would switch grips temporarily to look at the UI elements and get the information they needed, then switch back. The most successful solution was by players who switched to a two-handed grip. Notably, players who held the phone with both hands did not experience this issue, as the two-handed grip meant that they could hold on to the very edges of the phone comfortably, without obscuring UI elements.

This issue meant that players who preferred to hold the phone in a certain way would have important information partially obscured at all times, unless they adjusted in ways that were uncomfortable and inefficient. Upon showcasing this issue to the developers, the immediate feedback we received was that they were all used to playing the game by holding the phone with two hands, and so had never experienced this issue.

So, we knew what the problem was and why. However, there was no easy fix for several reasons. Simply moving the UI elements to the other side of the screen would invert the audience experiencing the problem; left-handed players would now experience the problem instead. The top and bottom of the screen already had existing UI elements that could not be compressed any further, else there would be new usability issues from the icon size being too small and unrecognisable. Likewise, the UI elements that were being blocked by players' hands could not be compressed. Compounding all of this was the timing of the playtest; we ran it less than a year before the game was due to launch, meaning that a full UI redesign would be difficult and might potentially create new issues, both technical and usability-related.

Because of this lack of time, the developers opted to take the least worst option and move the UI elements to the other side of the screen. The problem would still exist, but only for left-handed people, which account for roughly 10–12% of the world population. Not a great outcome, but when there is no good solution (given the constraints), it's better to have a usability issue affect a minority of players, rather than a majority. Had the

circumstances allowed for it, a better outcome would have been to run the usability playtest earlier, meaning we would have found the problem sooner, and there would have been more time for a better fix.

The ideal is to run usability tests as soon as possible. The earlier you can test, the better, since less budget and time will have been invested and potentially wasted if severe issues are found.

However, the reality is that even if you and the development team are fully aligned on testing as early as possible, more often than not, something will block or delay this from happening. The build of the game might not be ready, due to production delays or a desire to have it be as good as possible. Some other high-priority task may take precedence. The development team might have to cancel the usability test due to changes that lower the budget and time available for either UX testing or to make changes based on the UX test results. You and/or your team might not have the bandwidth to run the playtest.

Of these reasons, the one I've heard most in my career is, "We want to polish the build a bit more". This is completely understandable, especially if your game is going to be exposed to external players for the first time. There is a desire to make a good first impression, to get positive feedback on the project you've been working on for months or years. I'm guilty of this too, both as an analyst and a UX researcher; sometimes I want a bit more time to investigate or polish up my report or fine-tune some of the findings so they flow together better. When your work is going to be exposed to others, potentially facing criticism, you want your work to be as good as it can be. For usability testing, this can be counter-productive.

Usability tests don't always require polished, pretty builds for your game. It's nice to have, but what matters is that they are functional and have a working core loop for players to experience. You're not looking for sentiment, but whether players can fundamentally interact with the game. Some of the most effective usability tests I've run have been extremely early in a game's development cycle, before textures were implemented, sometimes with multiple placeholder art and audio assets in place. This environment is usually known as a greybox, and can be useful for playtesting.

Testing with greybox environments

We've been over paper prototypes, in Chapter 4, and greybox is a step up from that. In video games, a greybox refers to a simplified version of a level or an environment used to test and iterate on designs. The environment is usually built using blocks (grey boxes) in order to have a full layout quickly, hence the name.

Playtesting using a greybox can be useful to test the mechanics of your game's core loop early on in the development process, before significant time has been invested into detailed art assets or visual/audio effects. When playtesting using a greybox, there are some things to consider.

Since a greybox looks grey and bland, any investigation into player sentiment around the aesthetics of the game would be out of scope, and contextual clues from the environment that might inform players will not be available, which must be taken into consideration for related usability issues. For example, in survival-based games, players might understand implicitly that they need warm clothing when entering a snow biome, and water supplies if entering a desert biome. In a greybox environment, the lack of textures and effects make it unlikely the biome type would be recognisable, and thus players might lack the implicit understanding of what was needed to survive. The lack of visual or audio effects also mean that player sentiment towards the core loop may be muted.

Greyboxing is a useful tool for both playtesting and showcasing a concept of your game, particularly as part of a pitch process to receive budget approval for a new game. A greybox demo lets you show your stakeholders a very basic version of your game in action, usually with the core gameplay loop implemented, and serves as a blueprint for the final version of the game. Stakeholders such as publishers or company owners will usually be the ones playing the greybox demo, or observing a playthrough of it, to evaluate its potential. Since the greybox demo should demonstrate the core loop of your game effectively, ensuring that basic mechanics are understood with minimal friction even with low-fidelity assets can be helpful in ensuring budget approval.

We ran a playtest on a greybox demo for this very reason; a small development team was going to showcase the demo to the publisher in a month or so, and wanted to remove as many friction points as possible for the stakeholders who would be playing the game. As a small team, they were extremely familiar with the game they had designed, pitched, and developed a greybox demo for, and were concerned that their existing

knowledge smoothed over any existing usability flaws the game might have.

We (researchers and development team members) discussed what the goals were; mainly usability-focused, with some interest as to whether players enjoyed the game as a bonus. This was because the team wanted to prove their concept could work to the publisher, and players enjoying even a greybox demo would be a helpful point in their favour. Ultimately, the greybox demo had to be easily understandable and playable, with the controls, UI, and core gameplay loop being the main points of interest.

While recruiting players, we made sure to be upfront with them that the playtest they might choose to participate in would involve a game that was very early in development, to set expectations. We would not have wanted players to travel all the way to playtest a game, expecting a near-final polished product, only to be disappointed, potentially skewing their impressions, whether that be for usability or sentiment. This information was repeated to successful applicants upon their arrival, to set expectations for what was being tested.

We learned that of the main points of interest, the controls worked as expected. Players were able to navigate the world and control their units with very minimal onboarding, using only a sheet of paper with instructions. The UI was more of a mixed bag, with some aspects being familiar and clear, while others caused confusion for players repeatedly. Finally, the core loop itself was understood by players, who knew what they needed to do to progress (the core loop was clear), if not necessarily why they were doing it (the story and setting were not part of the demo).

The development team took on the feedback and were able to make the improvements needed to resolve the aspects of the UI that players had found confusing. Because of this, the pitch with the stakeholders went well, and the game was approved with some budget to move on to the next stage of development.

Usability benchmarking

So far, a lot of usability examples provided have been about qualitative data; players don’t understand a mechanic, players get stuck somewhere, players are confused about something, and the why behind all of these

problems they discover. But usability testing doesn't have to be only qualitative.

Your game might have a new mandatory task for players to complete, whether that be solving a puzzle, defeating an enemy, or constructing a room. You could quantify your players' efforts using measures like:

- Completion rates: What % of players were able to complete the task.
- Time to complete: How long did it take players to complete the task on average.
- Single-ease questionnaire (SEQ): How difficult was the task.
- System usability scale (SUS): More on this below.

The first two are straightforward, letting you monitor how many players succeeded in the tracked tasks, and how long they took to do so, which can be done manually or with the aid of game telemetry data. The SEQ is a simple question that can be asked at the end of the task, "Overall, how difficult or easy did you find this task?", with numerical options ranging from 1 to 7, with 1 being very difficult and 7 being very easy.

The SUS is more complicated, but easily used as it is a standardised questionnaire with ten questions and five responses for each question, ranging from Strongly Disagree to Strongly Agree (represented by integers ranging from 1 to 5):

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

Scoring the SUS is also a bit trickier than the previous measures, with the following steps:

- For the positive statements (odd number): subtract 1 from the user response.
- For the negative statements (even numbered): subtract the user responses from 5.
- This scales all values to values between 0 and 4, with 0 being the most negative response and 4 the most positive.
- Add up the scaled values for each player, and multiply that total by 2.5.

By following this formula, each player ends up with a possible SUS score between 0 and 100. By averaging the SUS scores of your players, you are able to arrive at an average SUS score that describes the usability of the game system being tested.

Obviously, this score on its own is meaningless, as we need to compare it to something. Studies have shown that an SUS score of 68 is considered average, and if your game system scores slightly above 80, it is better than 90% of devices and digital interfaces. You can also compare your game system's SUS score with previous iterations, learning if your efforts are improving the usability of your systems. A point to note is that while qualitative usability testing has diminishing returns with more players, quantitative usability testing using the above methods benefits from having more players, as that will lower the confidence interval and increase the reliability of your scores.

ACCESSIBILITY

In order for games to be truly usable, they also have to be playable by everyone, or at least as many people as possible. Which is where accessibility and accessibility testing comes in. Like any other person, players are just as likely to have some form of disability, where that be visual, auditory, motor, or cognitive. While any or a mix of these disabilities can impede a player's gaming experience, the blocker is not with the player or the disability, but with the game.

Consider a person in a wheelchair, trying to get into a building which requires ascending a few stairs to get to the front door, but has no ramp. Neither the person nor their disability is the blocker; the building lacking a ramp for accessibility is. So too is it with games; if a game requires the player to listen to dialogue or auditory cues in order to solve a puzzle, this can be a blocker for players who have auditory disabilities such as deafness. Adding an accessibility feature, such as subtitles, would remove this blocker.

Removing blockers isn't just a case of adding accessibility features; the design of the game itself plays a big part in helping with this too. Earlier in this chapter, Norman doors were introduced, along with the usage of two channels of information as an ideal solution, as only a single channel can create accessibility issues for some users. The same is true of games; presenting information to players using only one channel of information is not good design from an accessibility standpoint. For example, having an indicator flash green to signal that the player is in a safe area and red when in a dangerous area is a single channel of information and would cause problems for colour blind players who cannot distinguish between red and green. It would also affect players whose culture does not associate green with safety and red with danger; for example, in China, red is associated with good fortune. Rather than just having one channel of information (the colour), a second channel of information should be added, whether that be in the form of iconography (tick and cross), or text, whatever is suitable for your game.

Accessibility testing is a good way to get feedback from players with disabilities. It can be focused on understanding what blockers your game might have or testing if your accessible design or feature works to remove known blockers. When it comes to accessibility testing, having the capability to conduct remote testing is desirable, as it may be a much greater hardship for players with disabilities to come down to your UX research lab, which in turn limits the players you can recruit. In particular, non-remote testing is likely to cause self-selection bias, as some players with disabilities will be more capable of travelling to your playtest lab than others. Having remote testing options that allow any of these players to playtest from their home will greatly aid in recruitment and ensure player comfort is a priority.

Speaking of recruitment, the recruitment criteria for an accessibility test will require far more consideration than for a usability test. Simply recruiting "players with disabilities" is too broad. What are you interested in focusing on? Perhaps your game has a lot of text to read, so it would be useful to understand if players with visual disabilities like cataracts or cognitive disabilities like dyslexia can engage with your game, and if not, what you can do about it. Perhaps your game has a lot of fast-paced action, requiring quick reflexes and reactions, so recruiting players with cognitive impairments or motor disabilities would provide useful information on what you can do to enable an accessible and enjoyable gaming experience for them.

Actually finding these players can also be a challenge, and reaching out beyond your normal recruitment channels to charities and organisations that specialise in accessibility and disabilities can be helpful. For example, I've contacted the Royal National Institute for the Blind in the UK, as well as the Special Effect charity, who help gamers with physical disabilities. Both organisations have been very helpful in aiding with recruitment of players and are also happy to provide advice.

If you do conduct any accessibility testing in-person, do consider extra steps in addition to the ones outlined for usability testing; will players be able to access the venue? Is there sufficient signage available? Are there disabled toilets that are close enough to your lab? Does your playtesting setup take into account the extra space that may be needed for a wheelchair or someone on crutches? What about sensory overload for players who have cognitive disabilities? Are there any flashing lights anywhere that might trigger epilepsy?

For example, with regard to accessing the building, we checked in with our building support staff and discussed alternate ways to enter the building. If a person's wheelchair made it impossible to fit through either the revolving doors or side door, we had a larger side entrance available as a backup entrance. While we were only running one-person sessions, we used the large playtest room instead of the smaller room, as there was much more open space available. The large playtest room was also the only one with dimmable lights, which would be useful if any players were sensitive to bright lights.

Also consider any tools that might be needed, as players with disabilities may use screen readers, magnifying tools, or other devices that enable them

to interact with the game. It is likely that players may bring their own devices, but checking in with them to see what their needs are can help with the playtest setup. It can be very illuminating to see how players overcome blockers with external devices, as you can learn where your game helps or impedes them with this. One of our sessions included a player with motor disabilities, who could only use one hand to interact with a controller. He could still play games, using a tool that would translate voice commands into controller inputs, and was able to progress quite far through one of our games. Unfortunately, he reached a point in the game where he had to repeatedly press a controller button in order to complete a mandatory task and progress, and was unable to do that physically. While he could use voice commands to simulate the input, the tool did not allow him to do that rapidly enough to fulfil the success criteria, creating a blocker that meant he would never be able to progress beyond that point.

This was a useful finding that reminded the team that an accessibility option needed to be added for game mechanics like this. For example, adding a toggleable accessibility feature that allows players to complete the task by holding down on the button, or simply press it once to complete, or even let the task auto-complete, any of these would have removed the blocker entirely and allowed the player to continue enjoying the game. More and more game companies have added such accessibility features, removing blockers and opening up their games to a wider audience. This even has a side benefit of providing more customisation for players who don't have disabilities. Consider a player who is capable of mashing a button, but doesn't find that to be a compelling experience; that player can also take advantage of the accessibility setting to customise their experience as they see fit. It's a win for everyone, the company for enabling more players to play (and thus more sales), and for players to have enjoyable experiences, regardless of whether they are disabled or not.

LEGALLY BLIND

Legally blind does not mean a person is in total darkness, but indicates that they have severe vision loss that corrective lenses cannot help with, and likely needs adaptive tools for mobility (canes), or reading

(magnifiers or screen readers). This status enables benefits to aid people with such a disability.

Beyond these extra steps, accessibility testing itself is not too different from usability testing, so I will not go into too much more detail on the process and instead focus on some useful findings that impacted decisions.

One of the games I had worked on had a significant amount of text to read, some of it for informational and instructional purposes, the rest for immersing players in the world. We wanted to make sure we were doing everything we could to remove any possible blockers, so we ran an accessibility test with a focus on recruiting players with visual disabilities, such as glaucoma, cataracts, and legal blindness.

We were able to run the playtest with players who had a mix of visual disabilities, and through their actions and feedback, we gained a better understanding of how accessible our game was. Some results were expected, some surprising, one slightly embarrassing. In some areas, players found the text difficult or impossible to read because the font size was too small, calling out sections that could only be understood with the aid of a screen magnifier or text-to-speech tool.

Font size wasn't the only issue. The game used a mix of different panels to showcase text, with some panels being stylised to match the setting. Due to the styling, some panels had contrast ratio issues. Contrast ratio with regards to text refers to the brightness difference between the text and the background. For example, you might be reading this via black text on a white background, or, if you are reading this on a device in dark mode, white text on a black background, both of which have a high contrast ratio. A low contrast ratio minimises the brightness difference between text and background, for example black text on a dark grey background. Depending on how low the contrast ratio is, the text might be difficult to read or entirely unreadable. This came about in our game because the text used a consistent font colour, but as mentioned earlier, certain panels had different styling to fit the context, and subsequently had different background colours, some of which created low contrast ratios.

The embarrassing finding we had was that some players did not make use of the accessibility menu, despite navigating to it. When asked about this, these players mentioned that they could not read the options, as the text

was a bit too small for them to make out, so the feature we provided to aid with accessibility was itself not accessible.

Most games now have some form of an accessibility menu, where options related to improving accessibility are located, and in some cases, duplicated from other menus (subtitles sometimes being present in both audio and accessibility menus). Accessibility menus can be a huge boon for players with disabilities, providing them with a one-stop area for their needs. More and more games are also starting to display the accessibility menu as the first thing a player sees when they launch the game for the first time, reducing or removing potential blockers players with disabilities might have. For example, a player with a visual disability or disabilities may have trouble navigating the main menu screen to locate the accessibility menu. For this very reason, accessibility menus on startup tend to have the text-to-voice option at the very top, and enabled by default, so players with visual disabilities that need the audio as a second channel of information have it from the very beginning.

Since the accessibility menu is one of the first things players in our games experience, we wanted to make sure it was as player-friendly as possible. Once the developers made accessibility improvements based on the feedback detailed in the examples earlier, they also experimented with adding previews for each setting. Initially, the colour-blind preset settings were for Deuteranopia, Protanopia, and Tritanopia, which are the more common forms of colour blindness, but in order to actually see what effect they had, players would have to select one of the settings and then start playing in order to understand if the setting was the right fit for them. If it was not, they had to quit what they were doing, go back to the accessibility menu, and try again, a tedious process, especially if none of the preset settings helped them.

To improve the player experience, developers added a suitable preview image next to the colour blindness setting, so that players could see how each setting would affect the image, removing the back-and-forth process above entirely. The players with disabilities that we recruited to playtest this new and improved accessibility menu were very positive about the change, as it minimised their frustrations with having to experiment with what the settings actually did.

The previews were not limited to colour blindness settings; developers had also added previews for what larger text size and increased contrast

ratios would look like. Due to all these additions, players mentioned an unforeseen benefit; the previews encouraged them to experiment with all the accessibility options, even if they did not have a disability that necessitated said options. They wanted to see if any of the non-relevant accessibility options would still improve their experience.

Heuristic evaluation (also known as expert analysis)

A heuristic evaluation is a method where general design principles are used to evaluate game UI or systems to identify usability issues based on said principles.

A benefit of a heuristic evaluation is that the preparation for it is much simpler compared to other research methods involving players. First, define the scope of what is it that will be evaluated. Next, select the people who will be running the heuristic evaluation; it doesn't have to be only researchers. I prefer to run them with a mix of researchers, UI designers, UX designers, and game designers, to cover as many perspectives as possible. After that, discuss and select a suitable set of heuristics to be used; Jakob Nielsen's 10 usability heuristics are a good starting point. It is important to set boundaries here, focusing on the scope of what needs to be evaluated; if you try to evaluate every possible aspect, the evaluation will likely be too broad in its findings, as well as requiring much more time and effort (one of the benefits of this method is that it is cheap and quick).

If time allows, I find it useful to run a short practice session with the group, using whatever heuristics you have selected. You could run an evaluation on an app, a website, or even one of your previous games or a competitor title. The goal here is to ensure all participants understand what is expected of them, and what the goals of the evaluation are.

Once everything is prepared and everyone is on the same page, it's best to run the evaluations independently, at least initially. Each person will have their own biases, small or large, that will affect what they discover or focus on, which is why we have a varied mix of people running the evaluation. The time spent on this depends on how complex the evaluation target is, but I've found that somewhere between 30 minutes and a couple of hours is

sufficient for systems like creating an account and signing up for a service, or going through a basic first-time user experience section of your game.

The people running the evaluation may or may not be familiar with what's being evaluated. Either scenario is fine, but those who are not familiar with what's being evaluated should go through the flow at least once as a normal player would. When actually conducting the evaluation, it's best to go through the flow one step at a time, keeping in mind your pre-selected list of heuristics and noting down anything that conflicts.

Once everyone has conducted their evaluations, it's time to bring all individual findings together, and discuss. What findings are common and in agreement, and what findings clash? Which discovered issues are the biggest concern, based on the goals of the study? What are the next steps, what should be fixed, what can be left alone? Are there any issues which may need more data to act on; should we prioritise this in a future usability playtest?

CLASHES

Clashes will occur, not just between people's findings, but between the heuristics themselves, and between heuristics and design intent. This is perfectly fine! Heuristics aren't hard rules, but guidelines, and the design intent, if strong enough, can and should be used to overrule these guidelines where appropriate.

As an example of how heuristics can clash with each other, Jakob Nielsen's second and fourth heuristics are "Match Between the System and the Real World" and "Consistency and Standards" respectively. The second heuristic states that for whatever systems are in your game, they should ideally match the real world or what players expect, while the fourth states that they should be consistent. However, the traditional save icon, while consistent across most applications, looks like a floppy disk, a format which has been outdated since the 1990s. Here, heuristic 4 takes priority over heuristic 2, as the floppy disc icon is associated with saving, regardless of its now defunct existence in the real world.

A design intent clash example would be FromSoftware games. Most games will try to guide the player and provide a non-punishing experience. In comparison, FromSoftware games tend to not point

players in the exact direction they need to go, instead letting the player discover what to do on their own. Their games reward perseverance and skill, but punish players who fail, with a mechanic that can cause players to lose some resources if they die repeatedly. FromSoftware games are anything but comfortable, but that is the design intent, and one that resonates with their target audience.

Overall, heuristic evaluation is useful when budget or time is a constraint, or when your game build isn't fully ready for players, as it can be done cheaply and quickly with just company employees. It does have some drawbacks, in that it does require familiarity (hence the practice run), and may miss real player issues, since heuristics cannot cover every possible player scenario.

Accessibility of company buildings and offices

UX Research isn't limited to improving your games. A heuristic evaluation can be helpful not just for the player experience, but for the experience of company employees as well. Earlier, I showcased examples of how accessibility testing can help players, both those with disabilities and those without, and the same is true for members of your company.

While conducting accessibility testing on one of our games, we took the time to ensure players with disabilities would have no trouble accessing the UX research lab and nearby facilities, in order to provide them with an accessible and comfortable experience. Once the playtest was over, we considered how some of the potential obstructions might affect existing staff. For security reasons, the access doors were quite heavy, as they had to close automatically to ensure that no one could get in without a security card. This presented a potential blocker or inconvenience for attending players whose disability might cause them to struggle or be unable to open said doors. Examples of such disabilities would include being in a wheelchair, needing to use walking aids like a crutch, or having weak upper-body strength due to motor neurone disease. Members of the company could easily have any of these conditions, making the building less accessible to them. What other accessibility issues might we be missing?

To find out, we contacted an accessibility specialist to come in and conduct an expert analysis of our company's facilities, with the goal of understanding where we could make accessibility improvements that would benefit staff, those with disabilities and those without. Together, we – the accessibility specialist, facility staff, myself and another member of UX research – spent a day walking around our office buildings, both inside and out.

The accessibility specialist was in a wheelchair and could use crutches for short periods of time, so she was able to show practical examples of accessibility issues in addition to pointing out problems based on her expertise. Upon arrival, she was able to demonstrate that the visitor bays for one of our buildings were too narrow; they were suitable for drivers with no disabilities, but for those who needed to disembark onto a wheelchair, there was insufficient space to do so. This was further exacerbated by this particular building not having any accessible parking. Such parking spaces would have to be added, or the visitor spaces would have to be widened to allow a wheelchair to fit easily between parked vehicles.

The building also had no ramp at the front entrance for wheelchair users or people using crutches. There was a ramp on the side entrance, but no signage at the front indicating this. The intercom to speak to reception was at the top of the stairs, so people who were unable to use the stairs would have no means to bypass them, nor receive information on how to do so. Since then, a vertical platform lift has been added to the front of the building to improve accessibility.

The specialist toured our buildings, alternating between her wheelchair and crutches for navigation. Through practical examples and accessibility guidelines, she was able to surface a variety of accessibility issues as outlined below:

- Guest lanyards and staff lanyards were not extendable; some card readers to open doors required bending over to reach them if the lanyard was hanging from a neck, an option not always available to people with mobility issues. She demonstrated this while on crutches, showing the awkwardness of having to do so.
- Handicapped toilets were available on every floor and had multiple helpful features implemented, apart from one. Each toilet had a long red string hanging from the ceiling, to be used in case of emergency.

However, it did not reach the floor, its end hovering about half a meter above. This was a violation, as it needed to reach the floor, in case an occupant collapsed and was not able to get up. The string needed to be pullable even if someone was unable to reach beyond floor-level.

- The fire exit access was cramped due to furniture being stored there, which is dangerous in an emergency (such as a fire!) when people are trying to get out. The door also had bolts at the top and the bottom, meaning someone in a wheelchair would not be able to unfasten the door.
- Kitchen cupboards were mostly at shoulder or waist height, with cutlery and utensils stored at those levels. Ideally, there should be a cupboard low down with a mix of all cutlery and utensils, so that someone in a wheelchair would have easy access to all of them, without having to ask for help. The same was true of stationary cupboards with office supplies.

It wasn't all negative, as she complimented the office layout as being very accessible. There were wide pathways throughout, and anyone in a wheelchair would have easy access to all desks. Multiple elevators were available as alternatives to stairs, and there were no tricky stair/ramp issues within the building; a person in a wheelchair could navigate the entire interior without any issues. The number of handicapped toilets was good, with at least one on each floor, and most of the issues discovered above were not structurally difficult to fix.

Having these notes & solutions from the accessibility specialist and footage of her showcasing problems with entering the building, accessing the fire escape route, and other issues above, was very informative for both facility staff who were with us for the entire tour, and for management, to see what issues our buildings had, and how to fix them.

Retention and churn

So far, most of the studies discussed have delved more into usability, rather than whether players like the game. When a game's usability is good, it goes unnoticed, allowing players to focus on the experience of playing, rather than struggling to play. Conversely, when a game's usability is bad, these struggles can impede the player from having a good experience.

We worked on a game that was in closed alpha, and the game was in a polished enough state that players could sign up to play it online, resulting in thousands of players providing behavioural data through their actions. The game's retention during this period was far below expectations, and the match data indicated that most players would play for a few matches and then never play again.

We had a lot of theories for what the problem could be, but the possible solutions varied, and we could not implement them all at once, as some solutions contradicted others. So, we ran a playtest to try and understand what was going on. During preparation, the behavioural data indicated that most players quit the game within 10 matches, informing how long we should run our playtest sessions. Because the game was played entirely online with the majority of players playing with, and against strangers rather than friends, we also decided to structure the sessions such that recruited players who played in the UX research lab would play online against strangers, rather than playing against other players in the lab. This was primarily to avoid players experiencing feelings of camaraderie that might occur from being in the same physical location, altering the experience that online players were having.

While this would prove to be the correct choice, it also applied some restrictions. We knew from behavioural data that players were mostly playing in the evenings, less in the afternoon, and barely at all in the mornings. This meant that we could not run any sessions in the mornings or even early afternoons, since it would have a negative effect on matchmaking.

MATCHMAKING

Online competitive games tend to have matchmaking services, which is the process of matching players with others in order to create an online match. Generally (but not always!), matchmaking will attempt to use some formula to ensure that players of relative equal skill levels are matched together, to ensure both fairness and a better player experience. It isn't fun to play a match where you or your team are entirely outclassed with no chance of victory. Even the winners of lopsided matches will eventually lose interest, since winning

effortlessly a few times might be fun, but doing so repeatedly starts to induce boredom.

Skill-based matchmaking works well when there are many players online, queueing for a match. However, when there are only a few players online, the matchmaking formula may be a drawback; players might have to wait minutes to play a match rather than seconds, due to having to wait for similarly skilled players to queue. Some formulas may have a time component, whereby the longer the queue time, the more lenient the matchmaking criteria is, exchanging shorter queues for players having less balanced matches.

Ideally, the more players your competitive online game has, the better, since this will equate to faster matchmaking, shorter queues, and more balanced matches, for an overall better player experience.

If we ran the sessions later in the evening, there would be more online players, leading to faster queues and a better player experience. But evening sessions would mean staff would have to work past normal working hours, and recruiting players for the playtest would also be more difficult, since the playtest would run past dinnertime. Support staff, like receptionists to receive players, and first aiders or fire marshals (both legally required) would also have gone home for the day, restricting what we could do. We settled on running mid-afternoon sessions, striking a balance between the best player experience and feasibility of running the playtest.

The goal was for players to play up to 10 matches, experiencing the core loop as naturally as possible, so we could identify what factors were causing them to end up quitting. Playing 10 matches would be quite lengthy and tiring, so we scheduled a break in the middle of the session, which also allowed us to have a short interview with players to get their initial impressions after a few matches, and let them relax after if needed. They would then resume the remaining matches, and have another interview about their experience at the end.

The results were interesting; during matches almost all players showed signs of frustration – muttering to themselves, wondering aloud what to do – and gave feedback that they would likely not play any more matches if they were playing at home, or would not have kept playing for up to 10 matches. The reason for this was that they felt the game wasn’t for them, or

that it was not their kind of game. This reason was interesting, because we had recruited players who specifically enjoyed and played competitive games in this genre. When we dived deeper into their answers, asking questions around why they felt the game wasn't for them, the answer was simple; they felt they were doing everything correctly, everything the game had taught them during the tutorial, but felt like they weren't in control of the outcome. Even when they won matches, they did not feel like they had done anything different from when they lost.

From observing their behaviour during matches, we had noted a consistent pattern between all these players; they were not engaging in a portion of the core gameplay loop. The core gameplay loop involved steps A -> B -> C -> D, and they were ignoring B entirely. We asked players about step B, and they were ignorant of it, saying they had no idea the step existed, or only being slightly aware that it did, but as a tertiary priority.

Because step B was actually an important part of the core experience, ignoring it due to lack of awareness or deprioritising it meant these players had a much lower chance of winning their matches, compared to other players who did. And when they did win (through luck, or when matched against players who also were not aware of step B), it was not a satisfying experience, since they felt they had not won through mastery of the game, but through random chance, having carried out the same actions.

So, the lack of knowledge of step B skewed the entire player experience, and players lacked this knowledge because the game's tutorial had glossed over it. The concept of step B was in the tutorial, but was neither explicitly stated nor emphasised, unlike the other steps of A, C, and D. Players naturally ignored or deprioritised step B, to the detriment of their experience.

The developers took on this feedback and made a sweeping change to the tutorial, integrating step B much more deeply and explicitly. Subsequently, based on this change, retention increased, indicating that players were sticking with the game longer, likely meaning a better player experience.

Narrative

Over the decades, games have branched out from focusing on gameplay to become cinematic experiences in their own right. Games can tell a compelling story in a way that other mediums like films and books cannot, since they can give the player control to shape the outcome, or allow the player to be the protagonist of the story. As with gameplay, these stories can be fine-tuned to ensure players understand and enjoy them.

Fine-tuning stories and testing them has existed long before games. Movies are shown to audiences at test screenings, and negative responses can result in changes ranging to alternate endings or shorter movie lengths. Authors provide beta readers with a manuscript of their book, using the feedback to tweak the story or further develop underwritten characters prior to publication.

Earlier in this chapter, I mentioned a development team wanting to improve the onboarding for one of our games, with one of the tools for doing so being an immersive narrative experience. As this was a major goal for them, they wanted to know if players could understand the story beats (usability), and whether the intended emotional beats were working as intended (sentiment).

Players can fail to interact with important UI elements, get stuck trying to carry out a task as part of gameplay, or simply ignore game mechanics, all observable events that inform what questions to ask. The same is not true of story beats; players who do not understand the story can still experience the game's flow without any blockers unless an understanding of the story is explicitly required to progress, and our game did not require it.

To conduct research on the narrative, we relied on player feedback and asked players to describe the role of important characters, why they had to carry out certain actions (plot-related ones, rather than gameplay), and their understanding of the overall story. We avoided asking if they understood the story, since this can lead to yes/no answers that may not be reliable; a player might have an incorrect understanding of the story but still say yes. Instead, we asked them to summarise the story, which gave us a better picture of what they did and did not understand.

Some players care about the story and the background lore of a game world, and some don't. As the latter tend to ignore story in favour of playing the game, we made sure to differentiate between these two groups of players through their gaming backgrounds (sourced during recruitment),

their behaviour, if possible (observations of them actively choosing to skip dialogue or cutscenes), and through direct questioning. If players who care about the story do not understand the story beats, that is a concern to be resolved; players who willingly ignore the story are not a problem, as long as they are having fun with the gameplay, as each player should be free to enjoy the game however they choose.

Questions about understanding the story beats and characters could segue naturally into questions on how players felt about them. These questions were open-ended, giving players a chance to elaborate not just on what they felt, but why. Depending on their answers, we could ask follow-up questions. For example, if they felt the protagonist did not resonate with them, why was that? If they felt the ending gave them chills, would they be able to elaborate on this? Their reasons helped the writers understand which story beats were working, which weren't, and more importantly, why.

Concept Art

Concept artists start their work on a game long before most others do, usually in ideation, as they will visually define how the game will look and feel. They might explore drastically different themes (fantasy vs sci-fi, steampunk vs cyberpunk), or variations of a theme (realistic vs stylised), with dozens of different concept arts to choose from.

Before deciding on the theme, stakeholders (concept artists, art directors, game directors, etc.) may want to know which themes resonate best with players, adding that factor to a list of pros and cons for why they should decide on one theme over others. This can usually be done with a survey, since preference towards a theme is subjective, and surveying thousands of players helps with reducing individual subjectivity in favour of what your target audience wants as a whole. If more thorough insights are needed, using focus groups can be a good way to dive into the quantitative data from the survey, adding context and a deeper understanding of why certain themes or variations resonate more than others.

Concept artists from one of our development teams were working on a new game in pre-production for an established franchise, and had multiple different themes and art style variants they wanted to validate with existing players. For this, we had procedures in place that enabled us to send surveys to our existing players directly, simplifying recruitment. We discussed the

research topics with stakeholders; were the themes and variants recognisable at a glance, whether players felt these themes and variants would fit the mechanics of our game well, which themes and variants resonated the best with players, which were a good fit for our franchise overall, and for all their answers, follow-up branching questions to ask about the why behind their selections.

Some of the findings validated stakeholder concerns; a few of the variants were inspired by popular mobile games, and existing players felt they were derivative and cheap-looking, unsuitable for our existing franchise. Artists also wanted to experiment with an anime aesthetic, which players felt looked good but was a bad fit for our franchise, preferring that we use that kind of aesthetic for a brand new franchise instead. (Note: People don't like change, and it's possible that if we had queried players new to our franchise, we might have received different responses. However, the main concern was ensuring that our existing players would be satisfied with whatever theme and variant we selected.)

Other findings, however, gave us surprising and useful insights. One of the themes was centred around building up an industrial empire, with rows upon rows of factories on display. While the intention was to instil a sense of industrial revolution and technology progress (and players did feel this was the case), quite a few players were also concerned about the glorification of environmental damage, due to huge plumes of smoke prominently displayed emanating from the factories. Given the current state of the real world and concerns about climate change, players were uncomfortable with glorification of pollution in our game.

We discussed the findings with stakeholders, and after excluding the non-contenders (themes and variants that did not resonate at all with players), the themes and variants were narrowed down to a handful. Stakeholders eventually went forward with one of the themes, based on a combination of our research findings and other factors (budget, creative desire, avoiding over-saturated themes).

UXR EPILOGUE

At the beginning of Chapter 8, I talked about starting my UX research journey with a pitch, whereby we would allocate some time over a period of a year to developing UX research. The goal was to have the capability to run at least one professional-level study by the end of the year, at which point said study's results would be evaluated to see if this UX research initiative would be progressed further. This section will briefly go into what happened.

Towards the end of the year, we were able to convince the game development team of our biggest franchise to let us run a research study on their game. Though they were initially reluctant due to lack of exposure to, and understanding of, UX research, they eventually agreed, based on the results of the studies we had run throughout the year.

We ran the study using the procedures already detailed (the "eye tracking in effect" example is taken from this study) and came away with a lot of useful data, not just for this game, but for the franchise as a whole.

Because it involved our biggest (and most costly) game in development at the time, there was a lot of interest in how players would react to it early in development via UX Research playtests, something we had not done for games in this franchise before. We showcased the results to the development team and company leadership in the same room, taking them through each issue, playing footage where necessary, and answering their questions.

The results were received well by all parties; the findings on player behaviour and their feedback made sense, even when it highlighted flaws and friction points in the project that developers had been working on for years. The development team were happy because some findings validated their hopes, while others justified their concerns, with the research ultimately helping them to make the game better for players. Leadership were happy because they could see the development team taking on the feedback, followed by making and prioritising design decisions swiftly, lowering production costs.

In the weeks following the presentation, the development team requested multiple research studies, so many that we had to apologetically decline half of them, as we did not have the resources to carry them all out. Our lack of capacity was a blessing in disguise, as it and the study's impact

resulted in leadership approval for a full UX research team and budget to hire multiple dedicated UX researchers.

REFERENCE

NN Group (Jakob Nielsen) heuristics: https://www.nngroup.com/articles/ten-usability-heuristics/

11 Case studies of using analytics and UX research data together

DOI: 10.1201/9781003735335-11

Analytics and UX research are very complementary to each other, helping to reinforce findings or fill missing data holes. Some of the richest insights have come from combining the two to tell a compelling story of player behaviour & feedback and cause & effect, leading to impactful decisions and results.

While I used what I learned both as an analyst and as a UX researcher in these examples, having both skillsets isn't necessary; you can just as easily team up with someone with skills complementing yours.

VALIDATING FEEDBACK WITH FPS PERFORMANCE

During one of our playtests for an in-development multiplayer game, a couple of players had lost most of their matches repeatedly. In the focus group, they mentioned that they felt the game wasn't enjoyable due to occasional stuttering that impacted their ability to win (it was a fast-paced PvP game).

After the focus group was over and the players departed, I discussed this with the other researchers. While we had not seen any evidence of stuttering during the sessions, the chances of observing either player's screen during a framerate drop are infeasible. We could have gone through their recorded

videos to validate what they had said, but there was a much simpler and more accurate way to check.

Our game was collecting and sending an assortment of data events, and one of those data events was related to PC performance, including loading times and frames per second. We had a look at the data specific to those players during the playtest, and it did in fact show that there were occasional drops in their frame rate. Because some players are better than others at noticing technical issues like this (and are willing to call it out), we checked the frame rate data for the other players, but found no similar anomalies.

This was strange, as we were using our dedicated playtest lab at the time, and all computers had similar specifications, which is ideal as you want your players to all be using similar hardware to ensure that their experiences are comparable. However, similar doesn't mean identical. We checked the hardware for each computer, and the two that were affected by the problem had identical graphics cards which were of comparable quality to the rest, but were from a different manufacturer. After speaking with the development team and investigating further, we collectively learned that the in-development game build had problems with that specific graphics card, which would be fixed in the future. This helped validate the two players' feedback, and we made sure to add that context into the report, as their negative feedback was influenced by a situation not related to the game itself, and would no longer occur once the issue was fixed.

SHOP PRICES

Free-to-play (F2P) games are capable of generating billions in revenue, with some of the more successful ones doing so annually. Unlike premium games, it is not enough to just provide players with a great experience so they'll return to play again. F2P games have to provide enticing reasons for players to want to buy something, whether that be a power booster or a cosmetic, a one-off purchase, or a recurring season pass.

This enticement can come through the design of the game. Some F2P games will start out easy and provide a steady stream of free power boosters to give players a good experience via quick progression. Then, the difficulty is ramped up over time, and the amount of free power boosters gifted to

overcome said difficulty is decreased. Eventually, the players' stock of free power boosters runs out, and players are given the option to purchase more to continue the positive experience or struggle without.

Other F2P games sell characters or units through lootbox mechanics that permanently empower players' in-game teams. This permanent power upgrade is offset by escalating the difficulty of in-game challenges over time, requiring newer and more powerful characters to be purchased to keep up with the increasing challenge. Cosmetics also play a part, relying on player desire to purchase characters or skins due to appealing aesthetics or nostalgia. Some F2P games make use of time-gated mechanics, such as requiring in-game energy or stamina to carry out actions. Said energy replenishes over time, necessitating players to stop playing and wait when the energy runs out, or alternatively, purchasing more in order to continue playing. Other time-gated mechanics include requiring players to wait for their actions to be completed, such as constructing a building. Players can wait hours for the building to complete, or they can pay to skip the timer and continue.

As with anything that needs to be sold, how you sell it matters just as much as what you are selling. Supermarkets, for example, will:

- Place high-margin items at average eye level height, as they are more likely to be seen and purchased. Children's goods are placed at a lower level to appeal to them and influence parents to purchase.
- Position essential everyday items like milk and eggs as far away from the entrance as possible, so customers must pass aisles of other items – and potential purchases – along the way.
- Group complementary goods together (such as pasta and pasta sauces), as customers are likely to purchase both, boosting sales.

We were working on a live (released) F2P game that players found enjoyable, but was falling slightly short of monetisation targets. One of many avenues to improve the monetisation was to try and optimise the in-game shop. While virtual shop screens in a F2P game don't have the luxury of using physical distances in the same way supermarkets do, the layouts, order of items, and customisation of the shop experience can still be used to encourage players to purchase.

To understand what we could improve in the in-game shop, we first had to understand how the shop worked, what was selling well, and what wasn't. At any given time, the shop would have standard packages containing set amounts of in-game currency, power boosters, and cosmetics, and more, with certain items being discounted from time to time to increase visibility and sales. The shop would display all discounted items at the top of the screen for increased visibility, followed by the standard packages sorted by ascending cost, and then the power boosters and tokens sorted by ascending cost.

By querying the data related to purchasing, we identified two distinct segments of spenders, low spenders and high spenders, with a clear gulf in overall spending between the two. We dug deeper into what they purchased and discovered that the low spenders didn't just spend less overall but also only purchased items below a certain value, whereas the high spenders purchased items with a variety of prices. We wanted to find out why these two segments of spenders were so different, and also understand non-spenders' lack of motivation to purchase.

To do this, we needed to gather detailed data and dive into players' answers to understand their motivations, so a survey would not have been a good method. Bringing players into the playtest lab was also not needed, since the game was live and players could experience it from their homes. While we considered online focus groups for a mix of detail and speed, we deemed detailed feedback unbiased by others to be more important than gathering the data swiftly, so we used online one-to-one interviews instead. We recruited by targeting spenders, low spenders, and non-spenders via an in-game survey, allowing players to opt-into being contacted if they so desired.

The interviews themselves were focused around players' purchasing habits and motivations; how did they engage with the game, what did they purchase and why, how did purchasing affect their game experience, and so on.

The insights that came from these one-to-one interviews were as follows:

- The low-paying segment had a very strict criteria for purchasing; they spent strategically on items that could help them progress, but only if the price was below a certain threshold. Multiple players in this segment referenced the price of a cup of coffee as the limit. Anything

under that felt OK to them, but anything over felt like they were spending too much on the game. When diving deeper into why they referenced a cup of coffee, the universal answer was that it was a comfy daily purchase that felt worthwhile and not too expensive, and they wanted their investment into the game to be on a similar level.

- The high-paying segment purchased items sometimes for strategic reasons and sometimes on a whim, depending if they were going to play for a long period and wanted to make the most of it. An interesting finding for why their purchases varied was that they felt that they didn't always have to buy the biggest and best items even though they could afford them; the game's challenge wasn't always sufficient enough for that. Sometimes buying the most expensive item was overkill, and cheaper items were good enough to enable them to breeze through the game's content.
- The non-paying segment was not interested in purchasing anything, as they enjoyed playing the game as long as it was free, and were willing to put up with frustration rather than pay. If the frustration ever became too much, they would simply leave and play some other game.

The game was coded such that it could change things subtly (such as the order of items in a store) based on a player's previous behaviour. For example, if a player had a preference for cosmetic items, the game could ensure that the items in the top row (and thus the most visible) of the shop were the latest cosmetic items. Based on our findings, additional logic was added to the code to surface all items under a certain value to the top of the shop for low spenders, increasing visibility of items they might like to purchase (and increasing revenue).

Changes based on player feedback were made beyond the shop as well; new optional settings of higher difficulty were added to existing parts of the game, with the goal of providing players with more challenging content (mainly for the high spenders). As these new difficulty settings were optional and did not block progress, this gave the high spenders something to strive for (and pay for, if needed), without negatively impacting the experience for other players.

The insights from this combined behavioural and attitudinal data report led to positive results overall; we saw a slight positive increase in how much low spenders spent, and while high spenders did not spend

significantly more, they ended up playing more, with this increased time spent on the more challenging content.

ADVANCING ACCESSIBILITY PROPOSAL

I don't like to mash buttons when I play games, because I don't find it fun, and I worry about developing Carpal Tunnel Syndrome. When I encountered Quick-Time Events (QTE) that required mashing a button rapidly to complete a task in order to progress, I'd use different hands or different fingers to try and spread out the repetitive motions. Ten years ago, I played Uncharted 4 for the first time and discovered the joy of accessibility settings. The game's QTE completion criteria could be changed from mashing a button to holding it down, via an accessibility setting. Since then, this has been one of the first things I've looked for in any new game I play.

This is one of the many ways improving accessibility can improve the player experience in games, for those with and without disabilities. But that was just my personal experience. Perhaps I was one of the few, rather than the many. I wanted to advocate for improving accessibility in our games, but I needed to know if there was enough data to justify it. I investigated disabilities, accessibility in games, and came up with the following data points.

The prevalence of disabilities themselves means there is a sizable audience of players with disabilities. As of 2023, the World Health Organization (WHO) has estimated that there are 1.3 billion people with some form of disability, or 1 in 6 people. Disabilities occur due to a variety of factors; they can be inherited through birth due to genetics, or caused as a result of injury, illness, or environmental factors. Disabilities also become more common as one ages, with vision, hearing, cognitive, and mobility issues all more likely to occur in the old than the young. This age factor was much less of a concern for gamers in the late 20th century, when the majority of them were young and video games were viewed mostly as a pastime for children. Over time, video games become more mainstream, increasingly played by those of all ages, and those young players from the 70s, 80s, and 90s have aged by several decades. And with that age comes the increasing likelihood of a disability or disabilities that might impede

those players from playing or enjoying games. Ageing players who have declining eyesight will have trouble reading text in games. Those with arthritis will have trouble gripping a controller. Fading hearing will mean sound cues may go unnoticed. And so on, as there are multitudes of disabilities and multitudes of ways they can hinder the player experience. These players with disabilities, if unable to play your game or games due to blockers, will either stop playing them or move on to other games that remove these blockers. Not designing with accessibility in mind means less potential players can buy and play your game.

Not all disabilities are permanent or long-term. Consider someone who has injured their arm and has to have it in a sling for some time to recover. Or a gamer who has recently become a parent, and needs to hold a baby in one arm for periods of time. Neither of these players is permanently disabled, but they now have difficulties playing games that require both hands (short periods of time for the parent). Accessible design or settings that help players with long-term disabilities (such as the loss of an arm or digits) will also help these people with temporary disabilities or temporary circumstances.

Some members of the development team were also advocates for accessibility, and for a sequel to one of our games, they had put in effort to enhance the existing accessibility features, namely colour blindness settings. They had added more colour customisation options and allowed players to change colours for specific parts of the UI in addition to the default profile settings. They had also improved how these accessibility options were exposed to first-time players. Through analytics data, we tracked if this had an effect on the sequel, comparing it to the previous entry. After adjusting for players who tried the improved settings and reverted the change – we didn't count those players – the improvement to colour blindness settings resulted in a ten-fold increase in the number of players using the feature. Since these were players who tweaked the colour profiles of UI elements and kept those tweaks for the rest of their recorded playtime, the assumption was that the new colour settings were satisfactory for their experience. It was a small change for the development team in terms of effort, but one with a large impact affecting hundreds of thousands of players. In addition, colour blindness affects around 8% of men and 0.5% of women, but the ten-fold increase we were seeing in the data was far more than 8% of our players. This implied that the new colour-blind settings were

being used by players who were not colour blind, presumably improving their experience with the game.

We also had run UX research playtests focused on accessibility with disabled players, and were able to showcase the blockers these players were experiencing with our games, such as a person with a motor disability being unable to mash on a button fast enough to progress past a mandatory task. Videos of these struggles, along with their feedback on other games that removed these blockers, provided useful qualitative data on the low-hanging fruit available with regards to cheap improvements resulting in impactful changes.

There are also legal requirements in the United States that mandate accessibility for communications mechanics within video games, known as the 21st Century Communications and Video Accessibility Act (CVAA). CVAA, which had come into law at the time, meant that any games we released in the United States (which was all of them, since the United States is one of the biggest markets for video games) would be subject to CVAA, so increasing accessibility in our games to match their requirements was not just good for our players, but for us from a legal standpoint. More recently, the EU has its own directive, the European Accessibility Act (EAA), which also mandates accessibility standards for games.

In summary, the data points I had gathered in favour of accessibility were as follows:

- 1 in 6 people have some form of disability (World Health Organisation), and gamers aren't exempt from this, especially as the generation that grew up with video games has now aged by several decades, with age being a correlating factor with disabilities. As such, games that have blockers due to disabilities will be subject to fewer players with disabilities buying them, and more existing players moving on to games without such blockers.
- Not all disabilities are permanent; accessibility helps people who are temporarily disabled as well, further increasing the size of the audience that accessibility can enable to enjoy games.
- Investing in accessibility, especially when your game doesn't have much or any, can yield benefits far in excess of the costs, since it reaches not just players with disabilities or temporary disabilities, but

also players who want more customisation options for their experience.
- Legal requirements in the United States of America and Europe have also been implemented, so improving accessibility in games avoids running afoul of those and incurring penalties.
- Finally, although this wasn't a data point and more of an appeal to emotion, investing in accessibility to enable players with disabilities is the right thing to do.

I collated all of this data (with more charts, sources, and competitor examples) into a story of how the company and our games could benefit from improved accessibility.

For the people concerned about resulting budget increases; the data showed that due to the likelihood of an increasing gaming population with disabilities, investing in accessibility could increase our sales and avoid players with disabilities leaving our game due to blockers. It also avoided any possibility that we would run afoul of breaching CVAA or EAA and incurring penalty fees.

For those who wanted to make the player experience better; we had behavioural data showing how a small improvement resulted in a ten-fold increase in players engaging with the accessibility feature, presumably providing them with a better gaming experience. We also had both behavioural data and attitudinal data from players with disabilities from playtests of our games, to help demonstrate how simple non-accessible design could create impassable blockers.

For the marketing team; it would be good PR to showcase that we were doing more for players with disabilities, and potentially aid in selling the game. For example, The Last of Us Part II by Naughty Dog was featured in an article on CNN detailing how a blind gamer was able to experience the game because of the effort Naughty Dog had put into accessibility.

This story of accessibility and its benefits, aimed at multiple segments of the company and powered by data from multiple aspects, was well received, first by the company's leadership team, and then by company staff. There was a shift in accessibility efforts afterwards, with more effort being put into our games, and into our company.

DLC INVESTIGATION PROJECT

A project I was working on had a large amount of paid downloadable content (DLC) available, with varying types, content, budgets, and prices. With each DLC came varying degrees of critical and commercial success. We also had financial data on unit sales and revenue, and data hooks to understand how players were interacting with the DLC content. What we didn't have were concrete insights into why each DLC performed differently and what players valued out of having ongoing additional content.

We started with the data we already had, player engagement, which could be measured in different ways depending on the type of DLC. Some DLC types would add content such as new units for players to use in combat, or cosmetics, or new factions to replay the game as, or new campaigns to experience. We avoided comparing measures such as usage or time spent on the different DLC content, as they would not have been like-for-like comparisons. For example, players would be able to play dozens of battles with the new units from one DLC in a span of 20 hours, but would be unlikely to complete a new campaign from another DLC in that timeframe. Similarly, players might enable a cosmetic feature and use it indefinitely from that point onward, so measuring time spent using the cosmetic against, say, the new factions would be a non-useful comparison.

Instead, we used retention, focusing on whether there was any correlation between players who bought a particular type of DLC, or a mix of DLC, with their continued desire to play the game. We did not assume this to be causation, as we could not tell if players continued playing the game because they had purchased DLC, or if they purchased DLC because they intended to continue playing the game. Essentially, we would measure the success of DLC based on whether they had a high correlation with keeping players engaged with the game.

We also analysed publicly available player feedback in the form of Steam reviews, of which there were thousands. Our game was primarily sold on Steam, which allows players to rate games or their DLC as positive or negative, and provide associated qualitative feedback. We read through the reviews for each DLC we were analysing, tagging each review with keywords that summarised shared points players were bringing up about each DLC. This process, known as coding, allowed us to transform the raw

qualitative data from Steam reviews into groups of codes, along with the frequency in which those codes are mentioned, and whether those codes were associated with positive or negative reviews. We then analysed those codes and their frequencies to identify common themes of how players felt, within each DLC and across the DLCs, positive or negative.

Combining these two sources of data gave us a picture of which DLC had the best player engagement, whether that engagement correlated with Steam review scores (they did), what themes were responsible for high performance, and whether there were any common themes across high-performing DLC.

There was a DLC type that performed the best overall, in player engagement and positive reviews. Coding for this DLC indicated that players were overwhelmingly in favour of being able to replay the game in different ways due to this type of DLC enabling them, with correspondingly lukewarm feedback for the other types, which were fun, but did not enable this same replayability. Several other factors included players feeling that the top-performing DLC type had the best playtime to price ratio, while other DLC had less longevity due to feeling stale after a single playthrough.

With the behavioural and attitudinal data telling a detailed story of DLC performance, we wanted to add financial data to complete the picture. We compared revenue for each of the DLC, limiting the timeframe to match the DLC that had been released most recently, in order to have a like-for-like comparison. We also factored in the budget needed to produce each DLC to generate profit figures using those same timeframes, since a DLC that generates $10M in revenue but cost $9M to make would not be as profitable as one that generates $5M revenue for a $1M cost. The financial data also correlated well with our existing findings, as the high-performing DLC had the best profit ratios of all the DLC.

We packaged the findings above (with more details and a lot more charts) and presented it to leadership, which led to a strategy shift for this game and its sequels, emphasising the design and production of more of this type of DLC, while still producing the second-best-performing DLC type from time to time to add variety and avoid monotony.

12 Conclusion

On AI and how to prepare for using data in game development

DOI: 10.1201/9781003735335-12

THE AI SECTION

This section is going to be intentionally short, because it is very likely that by the time this book is published, everything written in this section will be completely out of date due to the rate at which Large Language Models (LLMs) and Artificial Intelligence (AI) tools are advancing.

First things first, AI tools can be used to help with data analysis. However, at the current time of writing this in early 2026, AI tools like ChatGPT do not understand what it is they are analysing. They are built on LLMs that are able to use an extremely complex and sophisticated pattern-matching model to select the correct words and outcomes based on statistical probability. AI tools are very fast, and unfortunately, very confident in their output with no understanding of whether that output is right or wrong.

I have used AI tools to speed up data analysis, one example being coding a large amount of qualitative player feedback. A survey with over 20,000 responses to an open-ended question is a lot of text to read through, and AI can help significantly cut down on this time. But since AI can hallucinate (essentially a confident but incorrect output), it is best to do some of the analysis work yourself and validate the AI's output against your own.

Using the above example, I've randomly sampled 10% of the 20,000 responses (after cleaning the data), manually reading and coding them

myself, and using those codes to create a summary of findings. Independent of my own analysis, the remaining 90% was submitted to an AI tool, with a suitable prompt written to perform similar coding and summarising. I then compared the two outputs to check if I've missed anything, or to see if the AI has hallucinated any findings. The ideal would be that our summaries roughly match, and that I'm able to validate any findings the AI has that I have missed, producing a combined output that takes much less time than doing it without an AI tool, without risking the inclusion of bad data.

You could use AI tools for all of the data analysis without any checks (it will be much, much faster), but the reality as of the current time of writing of this book is that AI can hallucinate its findings, hallucinations equate to bad data, and bad data leads to bad decisions, so I would strongly caution against this.

Perhaps by the time you're reading this book, AI technology will have advanced so much that its drawbacks (hallucinations leading to bad decisions) are entirely gone, and the role of analysts and researchers will have transformed dramatically. Or perhaps it will go the way of Non-Fungible Tokens (NFTs). Or perhaps it'll end up somewhere in the middle, a useful tool like any other, but one that requires double-checking and experience with prompting to avoid ending up with bad data.

HOW TO PREPARE FOR USING DATA FOR GAME DEVELOPMENT

Perhaps you're a designer, or a programmer, or a QA tester, someone already working in the video game industry, and you want to use data more to aid with decision-making.

Or perhaps you're a student, or a professional outside of the video game industry, someone who wants to use their love of data to help make video games.

To do this, you'll need some initiative and skills.

Initiative

As someone who didn't start out as an analyst or a UX researcher in video games, but as a QA tester, initiative driven by passion and a willingness to learn landed me both my roles as an analyst and as a UX researcher. I applied for a data analyst role when I was in QA, and while I had experience with statistics and Excel, I had no experience with SQL, which was a hard requirement. So I spent 3 months doing overtime, learning to code in SQL when my work duties were over, then going home and continuing my studies. It paid off, and I was able to use SQL at a basic-intermediate level, which I then honed once I was hired and use it on a daily basis.

To transition into UX research, I had permission to spend a year experimenting on it in addition to my data analyst duties, but it also required extra effort beyond my normal work hours. I worked overtime, reading UX research books, watching videos, and coding a HTML-based UX research reporting system, all to expand my knowledge and ensure that by the end of the year, we would have the beginnings of a viable UX research team.

I did this while within a company, but that's not a requirement. You can work on your own projects and build up a portfolio of analytics or UX research work within or without company resources; hopefully, some of the lighter examples in this book can help!

If you need access to large amounts of game data, previous examples detailed the usage of publicly available data, such Steam Reviews for quantitative and qualitative data, and SteamDB for looking at historical data of concurrent players per Steam game. If you need more detailed research, playtest games, you can use a friend, a family member as your players. Even yourself, via heuristic analysis. All you need is a device to play with, your PC, your console, your phone, and a notepad for your observations and findings.

Detail your preparations, your methodology, and your findings in a portfolio demonstrating how you did the work. It doesn't have to be perfect. You'll make mistakes (I did, many times). And those mistakes should be part of the portfolio, what you learned, and what you would do differently next time.

If you happen to be part of a company already, you can try to experiment with analytics or research in a similar manner, just for your own game. If there's no analytics department, start with your sales data, or SteamDB

concurrent players. If there's no scope or budget for UX research, run a heuristic analysis on your game or look through reviews. Start small and build up over time. My initial UX research playtests resulted in no decisions made for the game; they were focused on learning the craft, making the mistakes then to avoid mistakes now. Read and learn from books, videos, and social media and convert theoretical knowledge into practical skills.

Skills

As someone who has interviewed and hired analysts and researchers, these are the skills I'm looking for when I go through CVs (assume junior analyst/researcher).

Any data analyst would be expected to have experience (work-related or self-taught) in the following:

- SQL (any variant)
- Statistics (descriptive, inferential, data sampling)
- Excel (pivot tables and advanced functions like INDEX & MATCH)
- Data visualisation tools like Power BI, Tableau

And any researcher would be expected to have (also work-related or self-taught):

- Knowledge of qualitative and quantitative methods (usability, surveys, interviews, focus groups, diary studies)
- Good communication skills, both verbal and written
- Statistics (descriptive, inferential, data sampling)
- Familiarity with common research tools like SurveyMonkey, Alchemer, UserTesting, PlaytestCloud

Making an effort to build up skills and knowledge for either (or both) of these lists will go a long way; even if you have no relevant video game experience, it will show hiring managers that you have put in some work and have the right mindset, showing that you want to work with data.

Why I work with data

I love working with data, whether it's quantitative analytics data, measuring and seeing how players behave in games, or a mix of qualitative and quantitative data, combining what players do in playtests with what they say about their experience, or using both UX research and analytics data together to paint a clearer picture of what and why behind our players. Finding answers to problems, even partial ones, gives me energy.

As with any problem, understanding it is the most important part of any solution; if we don't understand what players like and dislike about our games, and why, our decisions are less likely to have an impact. Not having data doesn't mean decisions in video game development can't be impactful; it just lessens the likelihood they will be. Having good data is like being pointed roughly in the direction of where you need to go; it's not a guarantee, but it's better than wandering in a randomly chosen direction.

So, after reading through this book, after reading through all the examples, the question you have to ask yourself is: How do you feel about working with data?

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z