TÉCNICO LISBOA

Active vs Passive Flow Adjustment in Games

Alexandre Nunes Zambujal Chícharo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Carlos António Roque Martinho

Examination Committee

Chairperson: Prof. Pedro Miguel dos Santos Alves Madeira Adão
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Maria Beatriz Carmo

December 2021

Acknowledgments

First off all, I want to thank my supervisor Prof. Carlos Martinho for all the support. This Thesis wouldn't have happened without his knowledge and immense patience.

I also want to thank my good friends and colleagues, Diogo Caria Ferreira and Nuno Martins, for being there and helping me develop ClusterTechRush. Together we learned a lot and made something worth showing off. Without them the game wouldn't be what it is now.

I want to thank my parents for all their love and support, they were always there for me, and they made me the person I am today. I also want to thank my sister for for always being there through thick and thin. And I want to thank my grandparents for their support and affection.

Last but not least, I want to thank all my friends and colleagues that were their in the best and in the worst moments. We went trough some pretty cool moments over the years, we grew so much, and it would definitely not be the same without any of you.

To each and every one of you – Thank you.

Abstract

Flow is the mental state in which a person is completely immersed in the activity at hand. Flow is arguably the holy grail of game design, yet it isn't trivial to design into our games. The main challenge comes down to matching the game's challenge with the player's ability. Because each player will have a distinct skill level, which will also improve at different rates, for a game to be able to keep a wider audience in the Flow state a highly adaptive system is required.

In this work, we explore two approaches for implementing flow in games, Passive and Active Flow Adjustment. The first refers to the use of dynamic difficulty adjustment systems and the latter to allowing the player control over the game's challenges through the mechanics of the game. We want to compare Passive and Active Flow adjustment, applied to an action game. For this reason, we developed ClusterTechRush, a top-down shoot'em up game, we asked participants to play two versions, one where the player controlled the game's difficulty and another where the game's systems had the control.

We didn't find any advantage when it came to experiencing flow between the two versions but we did find that more casual players felt more competent, less tense, and experienced more positive emotion while playing the version that used Passive Flow adjustment, which seems to suggest Passive Flow adjustment might be better suited for designing games aimed at more casual players.

Keywords

Flow; Passive Flow Adjustment; Active Flow Adjustment; Dynamic Difficulty Adjustment; Experience; Player Control;

Resumo

Flow é o estado mental em que a pessoa está totalmente imersa na atividade em mão. Flow é muito importante para jogos, mas conceber flow nos nossos jogos não é uma tarefa simples. O principal problema é ajustar o nível de desafio à habilidade do jogador. Cada jogador tem o seu próprio nível de habilidade e ao jogar um jogo esse nível vai melhorar a velocidades diferentes. Para que um jogo consiga manter audiências mais abrangentes no estado de Flow é necessário um sistema adaptativo.

Exploramos duas estratégias para implementar flow em jogos, Ajustes de flow passivos e activos. Na primeira são utilizados sistemas de ajuste de dificuldade dinâmicos, na segunda, através das mecânicas do jogo, o jogador tem controlo sobre os desafios do jogo. Querendo comparar estas duas estratégias aplicadas a um jogo de acção desenvolvemos ClusterTechRush, um *top-down shoot'em up*, e pedimos a participantes para jogarem duas versões. Uma onde o jogador tinha controlo sobre a dificuldade do jogo e outra onde os sistemas do jogo tinham o controlo. Não observamos vantagens em trazer jogadores para o estado de Flow em nenhuma das versões, mas observamos que jogadores mais casuais sentiram-se mais competentes, menos tensos e experienciaram mais emoções positivas, com a versão com ajustes passivos. Isto parece indicar que ajustes de flow ativos serão mais indicados para desenhar jogos para jogadores mais experientes.

Palavras Chave

Flow; Ajuste de dificuldade dinâmico; Experiência; Controlo do Jogador; Experiencia; Adjuste Passivo de Flow; Ajuste Activo de Flow;

Contents

List of Figures

List of Tables

Acronyms

AFA	Active Flow adjustment
EQS	Environment Query System
GEQ	Game Experience Questionnaire
HP	Hit Points
LOC	Locus of Control
PFA	Passive Flow Adjustment

1

Introduction

Contents

1.1 Motivation

The games industry is growing rapidly, having reached a value of over 150 billion US$ Worldwide [17]. With more and more people playing games, we as game designers dream of being able to create great experiences that everyone, no matter where they're from or how skilled they are can truly enjoy. What if we could make games that that would create an optimal experience no matter the player? That's where Flow comes in.

Flow is a term coined by Mihaly Csikszentmihaly [1] that refers to a state of mind where an individual is completely involved in the task at hand, there is a balance between how challenging the task is, and the individual's ability. It's linked with an altered sense of duration of time, and loss of concern over everyday problems, which is a very desirable property in video games. In his research, Mihaly observed that Flow could be experienced by anyone, independently, where they're from, as long as a few criteria were met. The problem is, that these criteria are not always easy to ensure.

Implementing Flow into games is not a simple task. With this work, we hope to help answer some of the questions about Flow so that we can be closer to a future where it is easier to implement. And, as game designers, we can bring optimal experiences to as many people as possible.

1.2 Problem

The ability for a game to invoke Flow on its players is a very desired quality. However Flow is not trivial to implement, there are a lot of variables that will influence an individual's Flow state. Jenova Chen [2] applied Mihaly's Flow to video games. And according to him, the biggest challenge is in keeping the player in the Flow zone, the space where the challenge of the game is in balance with the player's skill. This is hard because every player will not only have a different skill level at the start of the game but will also improve at different rates, as they progress through the game. To compensate for this the game must adapt its challenges to the player's ability as the game progresses.

Chen distinguishes between two ways of achieving this, either through Passive Flow Adjustment or Active Flow Adjustment. Passive Flow Adjustment (PFA) refers to Dynamic Difficulty Adjustment(DDA) systems, which through software will keep track of player Flow, and dynamically adjust the difficulty to try and maintain the player in the Flow zone. While Active Flow Adjustment (AFA) refers to player choice-driven difficulty adjustment giving control to the player, over the game's challenges. Considering control over the task at hand is one of the important elements necessary for Flow, Chen goes on to define a methodology for designing Flow into games based on active Flow adjustment. To test his methodology he developed the game Fl0w [2] which became a commercial success.

This doesn't make AFA immediately a better option. PFA has a lot of potential that we don't yet have the technology to harness. We still don't have the technology to accurately measure the Flow state of

a player. Ultimately, a generic PFA tool could be created, that could be used in different games. This would allow developers to save development time and money. Because Active Flow adjustment is more of a design problem, the same couldn't be done for AFA. There are also examples of successful games that use PFA, like Resident Evil 4 or Left 4 Dead, which proves it can also be a viable option.

The main difference between the approaches is the amount of player control, which raises the question of how important is player control when it comes to Flow?.

1.3 Research Questions

By having two very similar versions of the same game, the only difference being, how the difficulty is managed. we want to compare player experiences to see which version they enjoy more. In the DDA version, the challenge will be adapted to the player, by the game, while in the other version the player will be responsible for controlling the amount of challenge the game is presenting at any time. Locus of control defines a personal belief about whether outcomes of behavior are determined by one's actions or by forces outside one's control. Because the main difference between the two versions will be how much control the player has, it will be interesting to see how the player experience relates to Locus of Control.

With this work we hope to get closer to answering these questions: What option will produce a better game experience? Active or Passive Flow adjustment? which option will work best with our game? How important will player control be for maintaining the Flow state? How will Locus of control influence the player experience?

1.4 Contribution

This work is intended to further explore Flow, comparing different approaches to the problem of trying to maintain players in the Flow Zone. We'll compare Passive Flow Adjustment, in the form of a DDA system, and Active Flow Adjustment, by applying Jenova Chen's Flow in games methodology [2]. We hope to better understand how important player control is when trying to invoke Flow. And how much of an effect locus of control or other player traces influence.

This work will also contribute with a summary of the relevant areas, Flow, PCG difficulty in games, and Game experience evaluation, where we'll explore concepts that will be important for this work as well as some solutions to problems related to these areas.

As well as a testbed game developed in unreal engine. The game will be a Survival, Bullet Heel, Shoot'em Up, and will have two versions. One uses Active Flow Adjustment and another that will use employ Passive Flow Adjustment.

1.5 Document Outline

This document will be structured as follows:

- In Chapter 1 a presentation of the motivation for this work is given, as well as the problem it tries to solve, how the work will try to solve the problem presented, and its expected contribution;
- In Chapter 2, the related work, we'll explore Flow, Difficulty in games, PCG, Game Experience Evaluation;
- In Chapter 3 The Case study is presented. We'll describe the game that was created as a testbed game for this work, all its components, how they interact. We'll explore development challenges, and they were overcome;
- In Chapter 4, we'll discuss the preliminary evaluation, how it influenced the experience. And the final evaluation, procedure, and results;
- In Chapter 5, we'll summarize the work done, the conclusion we arrived at, and describe possible future work.

2

Related Work

Contents

In order to develop this work, certain concepts will need to first be introduced. We'll discuss Flow, concepts pertaining to Procedural Content Generation, explore different approaches for dealing with difficulty in games, as well as, exploring different methods for evaluating game experience.

This chapter will be divided into five sections, Flow, Procedural Content Generation, Game Difficulty, and Evaluation, and at the end of the chapter, there will be a discussion section that will link the previously discussed work to the work that we'll be doing.

2.1 Flow

2.1.1 What is Flow?

Flow is a term coined by Psychologist Mihaly Csikszentmihaly [1]. He described it as
"*being completely involved in an activity for its own sake. The ego falls away. Time flies. Every action, movement, and thought follows inevitably from the previous one, like playing jazz. Your whole being is involved, and you're using your skills to the utmost.*"
In his research, he tried to find out what brought happiness by interviewing people of different cultures from all around the world. He found that, regardless of culture, they all described the most enjoyable experiences they had in a similar way. He identified eight components that were almost always present when a person was in the state of Flow:

1. We confront tasks we have a chance of completing;
2. We must be able to concentrate on what we are doing;
3. The task has clear goals;
4. The task provides immediate feedback;
5. One acts with deep, but effortless involvement, that removes from awareness the worries and frustrations of everyday life;
6. One exercises a sense of control over their actions;
7. Concern for the self disappears, yet, paradoxically the sense of self emerges stronger after the Flow experience is over;
8. The sense of duration of time is altered.

These eight components, together, can invoke in a person a sense of deep enjoyment, that makes the task at hand worth it, for the sake of itself, regardless of the effort necessary to complete it.

When trying to design tasks that invoke Flow, a common challenge in game design, it is important to maintain a balance between the challenge provided by a task and the skill of the player performing the task. If a player's skill level is higher than the challenge presented, the player will get bored, and if the skill level of the player is too low for the presented challenge it will result in anxiety. The space in between is called the Flow Zone (Figura 2.1).

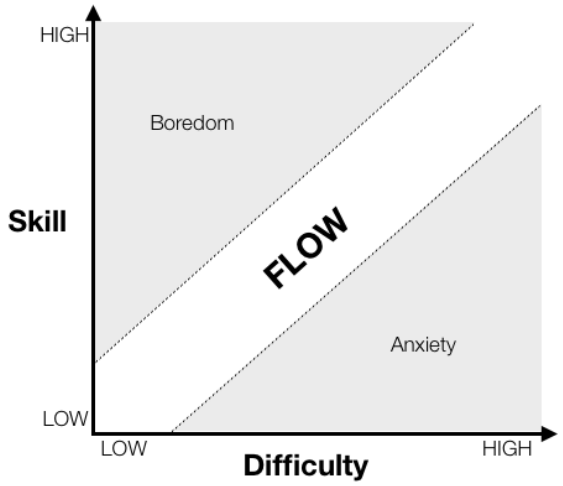


Figure 2.1: Flow Graph [15]

2.1.2 Flow in Games

In *Flow in Games* [2], Jenova Chen utilizes Mihaly Csikszentmihaly's Flow theory to define a game design methodology to create experiences that are adapted to different types of players. He breaks down Mihaly's eight Flow components and defines three core elements necessary for a game to invoke Flow experience.

1. As a premise, the game is intrinsically rewarding, and the player is up to play the game;
2. The game offers the right amount of challenge to match with the player's ability, which allows him/her to delve deeply into the game;
3. The player needs to feel a sense of personal control over the game activity.

These three core elements can be linked to the original eight defined by Mihaly has followed:

- Mihaly's Flow components 2, 3, 5, 7, and 8, the ability of the player to concentrate on the task at hand, clear goals, deep effortless involvement, the disappearance of the sense of self, and the altered sense of time duration, all contribute to a game that is intrinsically rewarding to play, and that the player will want to play.

- Mihaly's Flow component 1, "We confront tasks we have a chance of completing" directly relates to the J.C's second core element, the matching of the player ability and the game's challenge. Which when met will result in the merging of action and awareness, the disappearance of the sense of self and the altered sense of time duration, or Mihaly's components 5, 7, and 8.

- The third core element, the need for the player to feel personal control over the game, can be linked to Mihaly's third, fourth, and sixth components or the game has to have clear goals, the game must provide immediate feedback and the player exercises a sense of control over the game.

Assuming the audience is interested in playing the game, the biggest challenge in game design is how to maintain the player in the Flow Zone. Since different players will not only have different skill levels at the start of the game but their skill, as the game progresses, will also increase at different rates. The problem becomes how to continuously adjust the challenge in order to maintain the player in the Flow Zone until he finishes the game.

In order to realize optimal experiences for a much wider audience, not only do we need to offer a wide Flow Zone coverage, but we also need a highly adaptive system to weave the rich gameplay experiences together, adjusting Flow experiences based on the players. [2]

In Chen's work, he makes a distinction between Static Flow, Passive Flow adjustment, and Active Flow adjustment.

2.1.2.1 Static Flow

Static Flow refers to the approach that is most often used in commercial games, where the game is tuned for players of a certain skill level. And the game will not be adapted so that any player regardless of skill level can reach an optimal experience.

2.1.2.2 Passive Flow Adjustment

Passive Flow adjustment refers to the game systems that will adjust the difficulty of the game based on player performance. These systems often work under an iterative loop, that will monitor player performance, decide what needs to be changed in order to maintain the player in Flow, apply those changes, and repeat this loop (Figura 2.2).

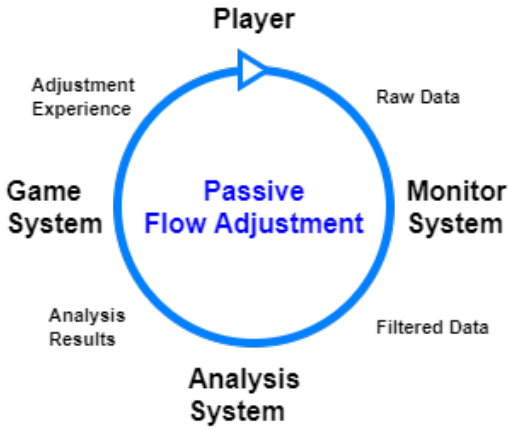


Figure 2.2: Passive Flow Adjustment

This approach in theory could solve the Flow problem, however, there are a few problems to which we still don't have a solution:

- No direct data - We still don't have the technology, to read what the player is thinking, it is really hard to know if a player is actually in a Flow state or not. And if not, if he is bored or anxious, so that an adjustment can be made.
- Performance doesn't not mirror Flow - Designer and researchers have figured out how to estimate player's performance, using data like, "Total Kills", "Accuracy", "Times Hit", "Headshots", etc. However, even if performance can be used as an approximation of Flow, it doesn't translate directly to it.
- Analysis based on assumptions - Assumptions never work for a mass audience. When a player enjoys performing a suicidal stunt in Grand Theft Auto, it would be ridiculous for a DDA system to assume that the player's skill is too poor because of the death count.
- Changes are based on rigid design – The way a system adjusts its difficulty is pre-determined by the designer. Different designers use their own preferences when deciding how many changes should be applied; however, the individual preferences of a designer will never represent the preferences of a mass audience.

2.1.2.3 Active Flow Adjustment

Through the lens of the three core elements of Flow, as defined by Chen [2], we will notice that most System-oriented DDA designs focus a lot on the second element, keeping the balance between the challenge and player skill. However, they ignore another important core element, to make the player feel a sense of control over the game activity.

Using Active Flow Adjustment, players can be given choices that give them control over their Flow experience. This can be achieved by giving players a wide spectrum of activities with differing difficulties. Depending on the player's skill and tastes, each individual will make different choices and progress through the game at their own pace(Figura 2.3).

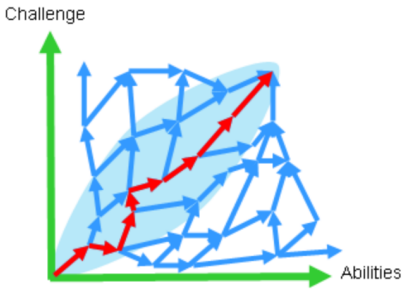


Figure 2.3: Active Flow Adjustment [2]

In order to adjust Flow experiences dynamically and to reduce Flow noises, the choices have to appear in a relatively high frequency [2]. A potential problem of this approach is that, depending on how these choices are presented to the player, they can be responsible for interrupting the player's Flow State. One solution to this problem would be to have a monitoring system to detect the best moments to present the player with these choices. Unfortunately, monitoring systems are not yet advanced enough to detect player Flow. The only solution is then, to embed the choices into the gameplay, letting the player deal with choices as part of playing the game, and, eventually ignoring them. Eventually, the choices will become intuitive and start reflecting their actual desires.

2.1.2.4 Flow in Games Methodology

According to Jenova Chen [2], designing a game system that will be able to invoke Flow in a wide range of players is not difficult. He claims that by applying the methodology shown below, any game can be become more dynamic and flexible, meaning the game will be able to invoke Flow in more players.

1. Expand your game's Flow coverage by including a wide spectrum of gameplay with different difficulties and flavors.

2. Create a Player-oriented Active DDA system to allow different players to play in their own paces.

3. Embed DDA choices into the core gameplay mechanics and let the player make their choices through play.

2.1.3 Measuring Flow

Sweetser and Wyeth [3] have developed a mapping of Game Flow that can be applied to evaluate a game's ability to invoke Flow. They tested the mapping by evaluating two RTS[1] games, one, Warcraft 3[2] considered to have a high rating and another, Lords of EverQuest[3] considered to have a low rating. Not only, were they able to clearly differentiate between the two games, as Warcraft 3[2] got an overall score of 4.8/5 and Lords Of EverQuest got 2.4/5, but they could also tell in what Game Flow elements the second one fell short.

In their mapping, the evaluation is divided into eight components. Concentration, Challenge, Player Skill, Control, Clear Goals, Feedback, Immersion, Social Interaction. Each evaluated according several criteria with a value from 0 to 5 where 0 is "N/A", 1 "not at all", 2 "below average", 3 "average", 4 "above average", 5 "well done".

Element	Criteria
Concentration Games should require concentration and the player should be able to concentrate on the game	– games should provide a lot of stimuli from different sources – games must provide stimuli that are worth attending to – games should quickly grab the players' attention and maintain their focus throughout the game – players shouldn't be burdened with tasks that don't feel important – games should have a high workload, while still being appropriate for the players' perceptual, cognitive, and memory limits – players should not be distracted from tasks that they want or need to concentrate on
Challenge Games should be sufficiently challenging and match the player's skill level	– challenges in games must match the players' skill levels – games should provide different levels of challenge for different players – the level of challenge should increase as the player progresses through the game and increases their skill level – games should provide new challenges at an appropriate pace

[1] Real-Time Strategy
[2] "Warcraft III: Reign of Chaos", Blizzard Entertainment, Microsoft Windows, Classic Mac OS, 2002
[3] "Lords of EverQuest", Rapid Eye Entertainment, Verant Interactive, Microsoft Windows, 2003

Player Skills Games must support player skill development and mastery	– players should be able to start playing the game without reading the manual – learning the game should not be boring, but be part of the fun – games should include online help so players don't need to exit the game – players should be taught to play the game through tutorials or initial levels that feel like playing the game – games should increase the players' skills at an appropriate pace as they progress through the game – players should be rewarded appropriately for their effort and skill development – game interfaces and mechanics should be easy to learn and use
Control Players should feel a sense of control over their actions in the game	– players should feel a sense of control over their characters or units and their movements and interactions in the game world – players should feel a sense of control over the game interface and input devices – players should feel a sense of control over the game shell (starting, stopping, saving, etc.) – players should not be able to make errors that are detrimental to the game and should be supported in recovering from errors – players should feel a sense of control and impact onto the game world (like their actions matter and they are shaping the game world) – players should feel a sense of control over the actions that they take and the strategies that they use and that they are free to play the game the way that they want (not simply discovering actions and strategies planned by the game developers)
Clear Goals Games should provide the player with clear goals at appropriate times	– overriding goals should be clear and presented early – intermediate goals should be clear and presented at appropriate times
Feedback Players must receive appropriate feedback at appropriate times	– players should receive feedback on progress toward their goals – players should receive immediate feedback on their actions – players should always know their status or score
Immersion Players should experience deep but effortless involvement in the game	– players should become less aware of their surroundings – players should become less self-aware and less worried about everyday life or self – players should experience an altered sense of time – players should feel emotionally involved in the game – players should feel viscerally involved in the game

Social Interaction Games should support and create opportunities for social interaction	– games should support competition and cooperation between players – games should support social interaction between players (chat, etc.) – games should support social communities inside and outside the game

Table 2.1: GameFlow Elements and Criteria

This mapping is designed to evaluate games, and not the player's experience with the game, so, it is not the best tool for comparing the two versions of our game. However, it should prove useful as a guideline for designing Flow into both versions of the game.

2.2 Procedural Content Generation

In this section we'll define Procedural Content Generation, as well as explore a few concepts related to it, that will be important for this work.

Procedural Content Generation (PCG), as defined by Togelius et al [5], refers to the algorithmic creation of game content. PCG refers to computer software that can create game content on its own, or together with one or many human players or designers. The implementation of PCG methods can be seen as solutions to content generation problems. A content generation problem can be the generation of grass textures in under 50 milliseconds, it can be the generation of an entire level, a sound effect, a song, an animation, or even content that might not be considered an asset by game engines, such as game rules, challenges, quests, etc.

PCG is used in games for a number of different reasons, including savings to development time and costs, increase in replayability, making adaptive games possible, assisting designers, and in academia, for studying game design. Depending on the application, the solution will require different proprieties. The common desirable properties of PCG solutions are speed, reliability, controllability, diversity, and believability. In this work, the main use of PCG will be, in the testbed game, in the form of the enemy spawner and Level Layout selection. That will be responsible for choosing where to spawn enemies. In the case of the Passive Flow Adjustment version of the game, the difficulty manager will also be responsible for increasing and decreasing the difficulty.

2.2.1 Taxonomy of PCG

With such a range of problems that can be solved using PCG, it is important to define a structure that highlights the differences and similarities between approaches. In this subsection, we'll explore the Taxonomy of PCG as originally presented by Togelius et al [6]

• **Online vs Offline**

PCG techniques can be used either online or offline. Online PCG means that it's being done as the game is being played, it's usually used to increase replayability and, since it's being done the player is playing, it can be used to generate content that is adapted to the player. Offline PCG is done even before the game is shipped, it can be used to generate large quantities of levels that can then be tweaked by a designer during development.

An example of Online Procedural content generation can be found in Hades[4]

where the player goes through procedurally generated rooms with different rewards and enemies, that are generated online, making every run different from the last.

• **Necessary vs Optimal**

PCG can also be used to generate necessary content, which is crucial for the completion of a level. Or Optimal content that is not necessary and can be discarded or swapped for other content.

This can also be seen in Hades[4] where certain rooms like boss rooms, will always appear but the regular rooms along the way, can appear or not depending on the player's choices. In the case of our testbed game, when it comes to the enemy spawner, or the layout selection it can be seen as Necessary PCG since it is an important part of the game.

• **Degree and Dimensions of Control**

How much control do the game designers have over the content that is generated? Some games, like Minecraft[5], use a random seed to generate content, then, with the same seed, the same content can be generated. Other games require more control from the designers and a set of parameters is used to control the generation over several dimensions.

• **Generic vs Adaptive**

PCG for Generic or Adaptive content generation. In Generic content generation, the content that is generated doesn't take into account the player's behavior. This is opposed to Adaptive content generation where the player interaction is analyzed and the content generated depends on previous player behavior. Most commercial games use generic content generation, while, lately, Adaptive PCG has been receiving a lot of attention in academia.

• **Stochastic vs Deterministic**

Deterministic PCG refers to the content generation that, from the same starting point and parameters, the same content will be generated every time. As opposed to Stochastic PCG, where, generating the same content is usually not doable.

The World generation in Minecraft[5] is a good example of deterministic content generation, with the same seed, the same world will always be generated.

• **Constructive vs Generate-and-Test**

[4]"Hades", Supergiant Games, Nintendo Switch, macOS, Microsoft Windows, 2020.

[5]"Minecraft", Mojang, Microsoft Windows, PS4, PS3, Xbox 360, Xbox One, Nintendo Switch, Android, IOS, 2009

Constructive PCG generates content in a single pass, while, in generate-and-test, the content is automatically generated and tested, in a loop, until a good solution is found.

An example of a game using the generate and test paradigm can be seen in Yavalath[6]which was completely generated with the help of PCG.

• **Automatic Generation vs Mixed Authorship**

With automatic generation, the content is generated automatically, by the software, while the designer's control over the generated content is exerted by tweaking generation parameters. On the other hand, in Mixed Authorship PCG the designer cooperates with the generation algorithm to create the desired content.

One example of mixed authorship can be seen in Tanagra [7], a system where a designer will define part of a level, and the algorithm will generate the missing part, while maintaining playability.

2.3 Difficulty in Games

As we have discussed previously, for a game to be enjoyable it needs to present its players with challenges suited to their skill level. In this section, we will present different approaches for dealing with game difficulty, used by different games, both commercial and in academia.

2.3.1 Difficulty Selection

The most used approach, in most AAA games, is the difficulty selection, where, before even starting to play the game, the player is prompted to select a difficulty setting from a small list of options, usually some variation of easy, normal, and hard. This approach is used frequently, due to it being simpler to implement compared to other options. The designers need to define minor variations to the different difficulty settings, like enemies having less health in an easier difficulty, or doing more damage in a harder one. This means the designers will end up having a finite number of versions of the game that they have to balance. And ultimately more control over the presented game experience.

The main problem with this approach is that the player needs to choose a difficulty before playing, or knowing anything about the game, and this choice will have a great impact on the player experience. If a player chooses a difficulty setting lower than the one that matches his skill level, the player will think the game is too easy and get bored, and if the player chooses a difficulty setting higher than his skill level then the player will be too hard, and the player will be frustrated.

[6]https://boardgamegeek.com/boardgame/33767/yavalath

Figure 2.4: Difficulty selection Menu from the game Remnant From the Ashes[7]

It is common for games, nowadays, to allow the player to change the difficulty at any point, but, usually, at the cost of losing access to certain achievements. However, players are not likely to adjust the game's difficulty after the start. Especial especially to lower it since most players will see that as quitting. In a time where most people will have access to more games, if a game experience is being boring or frustrating, players will just go play another game.

Most games will have a description for each difficulty settings (see Figura 2.4 and Figura 2.5), but it's not always enough information for the player to make a decision.

Figure 2.5: Difficulty selection Menu from the game Shadow of the Tomb Raider[8]

This approach is also used in 4X[9]games, and it can be greatly beneficial for experienced players, as they can tweak the experience to their liking. However, a great number of options can be overwhelming for new players.

In Diablo 3[10]'s approach, difficulty selection is used, but in a way that encourages the player to tweak the difficulty as he progresses through the game. As the difficulty increases, enemy health, and damage, loot drop chance, and experience points will increase significantly. If the player's gear, and

[7]"Remnant From the Ashes", Gunfire Games, Microsoft Windows, PlayStation 4, Xbox One, 2019
[8]"Shadow of the Tomb Raider", Crystal Dynamics, Eidos-Montréal, Nixxes Software BV, PlayStation 4, Xbox One, Microsoft Windows, Classic Mac OS, Linux, 2018
[9]Stands for Explore, Expand, Exploit, Exterminate, it's a sub-genre of Strategy games

level, are weaker than expected for a certain difficulty, with enough skill the player can still progress, but enemies will take significantly longer to kill, rewarding the player with less loot and experience over time. Making it so the player would progress faster if he was at a lower difficulty level. This way the game incentivizes the player to find the difficulty sweet spot and maximize loot and experience gain. To accommodate this, the game has twenty difficulty settings, Normal, Hard, Expert, Master, and then Torment 1 to 16.

2.3.2 Mechanics Driven

Another approach is to allow the player to select the difficulty through the use of game mechanics. This can be done by having challenges that can be solved in different ways, of varying difficulties, and then giving the player freedom to choose how he wants to tackle them and in what order.

For example, this can be seen in a game like *The Legend of Zelda: Breath of the Wild*[11] where a player has the option, since the start of the game, to go straight for the final boss. Or he can explore and get stronger, tackling the boss only when he feels strong enough.

Or in a game like Dark Souls[12] (One cannot talk about game difficulty without mentioning a souls game) where the game will vary in difficulty depending on what weapon the player decides to use. Ranged spells tend to provide an easier experience than a sword and a shield, and those tend to provide an easier experience than wielding a single sword. This approach is also flexible, maybe for a certain player, a mechanic like timing a dodge, is easier than the mechanic of managing stamina, so he finds using a single sword easier than using a sword and shield.

The problem with this approach is that it is hard to implement. It is easier to design a challenge with one solution than it is to design one that can be tackled in a number of different ways, that accommodate different skill levels. Not only that the game needs to have clear rules that need to be consistent over the course of the entire game.

2.3.3 Intrinsic Difficulty

In some games, the difficulty doesn't come from the challenges provided by the game, but from the desire of the player to complete it in a certain way. For example in a game like Assassin's Creed II[13] where if the player's goal was to just beat it, it wouldn't be a challenging game at all, and the players would get bored. However, the goal of the player is often to play the Assassin's Creed fantasy and to complete the game's challenges how an assassin would, by being stealthy. The game doesn't need to explicitly reward the player for playing in that certain way, the player has an intrinsic motivation to want to play in

[10] "Diablo 3", Blizzard Entertainment, Nintendo Switch, PlayStation 3, Xbox 360, Microsoft Windows, Classic Mac OS, 2012
[11] "Legend of Zelda: Breath of the Wild", Nintendo Entertainment Planning & Development, Nintendo Switch, Wii U, 2017
[12] Dark Souls, FromSoftware Inc., PS3, Xbox 360, Microsoft Windows, 2011

the desired way, and the challenge stops being to beat the level, but how to beat the level in this way.

2.3.4 Git Gud[14]

I want to mention difficult games in how they relate to Flow. If a player's skill level exceeds the one required by a game, the player will be bored and out of the Flow zone, and unless the player stops playing for a long time, his skill level won't go down so that he can go back to the Flow zone. If a game's difficulty is higher than the player's skill the player will feel frustrated and, again, out of the Flow state, however, if the player doesn't give up he can increase his skill until he can get in the Flow Zone. This can be seen in a game like Dark Souls [12] where even though the difficulty can be choice-driven even with all player choices into consideration it can still be a difficult game. A lot of players can't get over the difficulty barrier but those who do, almost always have a great experience.

2.3.5 Dynamic Difficulty Adjustment

Another approach with a lot of potential is the use of Dynamic Difficulty Adjustment (DDA). This can also be seen as Passive Flow Adjustment, as discussed before. Games that use this approach will have systems that monitor how the player is doing and will adjust the difficulty in order to try and keep the player in the Flow Zone.

First, we'll explore a few examples of how DDA has been used in commercial games and then explore what is being done in academia.

2.3.5.1 Commercial uses of DDA

- **Left 4 Dead**[15]

  Left 4 Dead[15], the first-person survival horror shooter by Valve is known for its seamless approach to dynamic difficulty. For the game, Valve developed an AI they called "Director" that was responsible for adapting the game depending on the player's actions. It would control things like Zombie spawns, weapon locations, and even segments of music [14], to maintain a fresh and unpredictable experience, that would add to the Zombie horror experience.

  More concretely, The Director would handle Zombies by deciding when to spawn hoards, what special Zombies[16]to mix in, and always making sure the zombies would spawn outside of the player's view. Weapon and health spawn had set locations, where they may or may not spawn, depending on

[13] "Assassin's Creed II", Ubisoft Montreal, Ubisoft Ukraine, PlayStation 4, Xbox One, PlayStation 3, Xbox 360, Microsoft Windows, Classic Mac OS, 2009

[14] Git Gud is an intentional misspelling of the phrase "Get Good", it usually used to poke fun at players when they complain a game is too hard.

[15] "Left 4 Dead", Valve Corporation, Turtle Rock Studios, Certain Affinity, Android, Xbox 360, Microsoft Windows, iOS, Classic Mac OS, 2008

how the players were performing. Sometimes the Director would even change level layouts, offering new paths or blocking others, depending on how much resources the player would have used until that point [14].

• **Mario Kart**[17]

Mario Kart[17] uses a rubber banding approach to difficulty. In the game, the AI racers will behave differently depending on how well the players are doing. If the player is struggling the rubber band AI, will slow down and hit more obstacles, giving the player a chance for a comeback. On the other hand, if the player is doing well, the AI racers will speed up, so they become a threat even to good players.

This approach helps every race feel exciting, giving less-skilled players a chance to compete, and highly skilled players still feel challenged. However, at higher difficulties, it can give the impression the AI is ”cheating” [14].

• **Metal Gear Solid V**[18]

In Metal Gear Solid V[18] the player has several ways of completing objectives, to keep things fresh, so that the player doesn't always use the strategies he gets comfortable with, the game AI adapts to counter the player's favorite methods. For example, if the player got comfortable snipping, and sniped all the enemies from afar before moving into the objective, the AI will adapt and more enemies will start using bulletproof helmets, forcing the player to adapt different strategies.

• **Resident Evil 4**[19]

Resident Evil 4[19] is an especially interesting example, because the developer, Capcom, has never stated that the game adapts its difficulty on the fly, and the game never explicitly tells the player that it's getting harder or easier. We only know it because of players comparing their experiences and players that noticed it after playing the game multiple times. In the game, if the players were doing particularly well, enemies would do more damage, be more aggressive, and resources, like bullets, would be scarcer. At the same time, if the player is struggling, the game will make enemies deal less damage, wait longer in between attacks, resources are more plentiful, and, if the player is really struggling the game would go as far as removing certain enemies, so that the player doesn't get stuck.

Because of Capcom's choice to not disclaim the use o DDA, there is probably a lot of players out there, that never even noticed the game was being adapted to their skill. This is a good thing because player's don't like being patronized, and the sense of achievement they get from beating the game

[16] In Left4Dead there are special zombies with different abilities, like the Boomer, a fat zombie that explodes and attracts more regular zombies, the Hunter, a stealthy zombie that will pounce and pin down survivors, The Tank, the strongest zombie that will pummel players and throw rocks at them, and more.

[17] ”MarioKart”, Video Game Series of Racing Games by Nintendo, the first one was released in 1992

[18] ”Metal Gear Solid V: The Phantom Pain”, Kojima Productions, Konami Digital Entertainment Co., Ltd., PlayStation 4, Xbox One, PlayStation 3, Xbox 360, Microsoft Windows, 2015.

[19] ”Resident Evil 4”, Capcom, Capcom Production Studio 4, PlayStation 4, Android, PlayStation 2, PlayStation 3, Nintendo Switch, Xbox 360, Xbox One, Microsoft Windows, IOS, GameCube, Wii, 2005.

can be lessened if they feel like the game gave them the win. Also, if the players know about it they will always try and exploit it. This can be seen when watching how Resident Evil 4[19] speedrunners will often take damage on purpose so that future areas will be easier to get through.

2.3.5.2 DDA in Academia

• Holiday Knight

Holiday Knight is a game developed by João Filipe Lopes Pardal [9] has a testbed game for his work in Holiday Knight: a Videogame with Skill-based Challenge Generation [9]. The game is a Bullet Hell[19]/ Shoot'em Up[19] where the player must defeat all the enemies in a room to move to the next one. The game would keep track of player performance in real-time and use that information to select the enemies that would be spawned in the next room. The selection of the enemies would also take into consideration a novelty factor, so that game would feel fresh.

In this work, enemies were seen as obstacles and had tags denoting their proprieties, for example, if an enemy uses ranged or melee attacks, or, if an enemy is fast or slow. The game will then keep track of how the player has been performing against different tags. And how long had it been since a tag was selected. For every room a challenge would be selected, a challenge being a combination of obstacles, depending on how it is predicted for the player to perform against that challenge, and depending on a novelty value, that is based on the novelty of the tags present in the challenge.

The hypothesis presented was that using a system like this, that adapts to the player would have players play for longer and\or replay the game more times. To test this hypothesis two versions of the game were used, one that used the dynamic adjustment discussed, and another where the challenges were static and selected by the designer. The final ended up not revealing any significant difference in the two versions when it came to the duration or the number of sessions. Even so, Pardal claimed this approach could still be useful since it allowed developers to save time on designing individual challenges for every room, meaning it wouldn't take much more work to create 100 rooms compared to 20.

• Fl0w

Fl0w[19] is an interesting example since it was originally created for academia, to accompany, Jenova Chen's [2] master thesis. But it was later reworked and released on PS3 where it became a commercial success.

Chen designed Fl0w using his Flow in Games Methodology. Since the game was intended to be used in research it was intentionally designed to be simple, facilitating the evaluation of the player-oriented

[19] Shoot'em Up is a sub-genre of video games within the shooter sub-genre in the action genre. game in which the protagonist combats a large number of enemies by shooting at them while dodging their fire. [16]

[19] Bullet Hell refers to a game genre where the player has to dodge a high number of projectiles.

[19] Fl0w, Thatgamecompany, Flash, PlayStation 3, PlayStation 4, PlayStation Portable, PlayStation Vita, 2006

DDA System.

Fl0w expands the Flow coverage by having simple controls so that players of any skill level can enjoy it while also having space for hardcore players to master it. From simply swimming around, eating, and fighting, the game manages to offer a wide range of gameplay.

There are 20 levels in the game, each level introduces new creatures and different challenges. With Flow in mind, the game allows the player, control over what level he wants to be at. By eating certain foods the player will go up or down a level. The player can choose to go straight to more advanced levels or to go back and get stronger before advancing. In order not to punish the player for dying, as that would take him out of Flow. So, if the player dies he just goes back to the previous level.

In the end, this game proved that Chen's methodology can work. In the first two weeks of being only the game got more than 350,000 downloads. And "Addicting" was the most common word to describe it.

2.3.6 How To Present Game Difficulty Choices?

In "How to Present Game Difficulty Choices? Exploring the Impact on Player Experience" Smeddinck et al [8] tried to explore the impact of automatic and player-choice driven difficulty changes, on player experience. For this, they conducted two studies:

For the first one, they developed a simple testbed game, where the player played as a bird and tried to fly as high as possible while collecting points and avoiding obstacles. The game was designed to have rounds of about 60 seconds. And to deal with difficulty two versions were implemented. One where the player was prompted, between rounds, if he would like to lower, maintain, or raise the difficulty. These difficulty changes manifested as a boost, where the lower the difficulty, the higher the player would be launched upwards at the start of the round. And another using DDA was based on the performance of the player in the last 5 rounds, the game would automatically tweak a few parameters in order to adjust the difficulty to the player.

For the second one, they recreated Fl0w and implemented three different approaches to deal with the difficulty. The first was the original Fl0w with its embedded choices to change the difficulty. The second gave the player the option to at any time pause the game and lower or raise the difficulty. And lastly, a version using DDA that would raise the difficulty of the game over time, and then adjust it back down if the player was hit.

In each study they had players play the different versions of the game. And found that there was not a significant difference in player enjoyment in the different versions. Even though Smeddinck et al found players said they preferred to have control over the difficulty, they couldn't find significant changes in presence and immersion of the different versions.

2.4 Game Experience Evaluation

In this section, we'll explore ways to evaluate the player's game experience. This will be important so we can compare both versions of the game.

2.4.1 The Game Experience Questionnaire (GEQ)

The Game Experience Questionnaire [10] was designed with a modular structure in mind, it has a core questionnaire, and can then be extended with a Social Presence module and a Post-Game module. It should be administered immediately after playing the game.

The Core part of the GEQ scores the player experience in seven components, Competence, Sensory and Imaginative Immersion, Flow, Tension/Annoyance, Challenge, Negative affect, and Positive affect. The Social presence module tests the Psychological and Behavioural involvement of the player with other social entities, they can be in-game characters, other players online, or, other local players. This module consists of three components, Psychological Involvement - Empathy, Psychological Involvement - Negative feelings, and Behavioural Involvement.

The Post-game module scores the player in 4 components, Positive Experience, Negative Experience, Tiredness, Returning to Reality.

Each item of each component is scored from 0 to 4, and each component is then scored by averaging the scores of its items.

2.4.2 Intrinsic Motivation Inventory

The Intrinsic Motivation Inventory [11] (IMI) is a multidimensional measurement instrument used to assess the participant's subjective experience related to a certain activity in laboratory experiments. The device is used to assess participants interest/enjoyment, perceived competence, effort, value/usefulness, felt pressure and tension, and perceived choice while performing a given activity, thus yielding six subscale scores. Each of the subscales will be represented by several items that the participants will be asked to evaluate with a score from 1 to 7, where 1 is ”not at all true”, 4 is ”somewhat true” and 7 is ”very true”.

When scoring, first, some of the items will have their score reversed (subtracted from 8), and then each subscale score is calculated by averaging across all the items of that subscale. The subscale scores can then be analyzed to answer relevant questions.

2.4.3 The Player Experience of Need Satisfaction (PENS)

The Player Experience of Need Satisfaction [12] (PENS) is designed to measure game players' experience across five dimensions: Competence, Autonomy, Relatedness, Presence/Immersion, and Intuitive Controls. Items in the PENS are presented to participants as statements about their game experience, which are rated from 1 to 7 where 1 is "do not agree", and 7 is "strongly agree".

2.4.4 Locus Of Control

Locus of control defines a personal belief about whether outcomes of behavior are determined by one's actions or by forces outside one's control [13]. Most Locus of control scales used twenty to thirty items, so, Kovaleva et al [13] constructed a four-item scale for the assessment of Locus of control(IE-4). Using the IE-4 scale player are asked how much they agree with 4 items:

- If I work hard, I will succeed.
- I'm my own boss.
- Whether at work or in my private life: What I do is mainly determined by others.
- Fate often gets in the way of my plans.

The first two items look for internal locus of control, thinking you have control over your actions, and the last two items for external locus of control, believing outcomes are determined by external forces. Each item is scored from 0 to 4. The final score is calculated by adding the two internal scores, subtracting the two external scores and then dividing by the number of questions, four.

2.5 Discussion

In this section, we defined concepts and explored literature related to our work. With this, we hoped to get an understanding of how our work can fit in the current state of the Game Flow related research.

Firstly, in section2.1, we introduced the concept of Flow as defined by Mihaly Csikszentmihaly [1]. We detailed its components and explained why Flow is desirable for video games. We continued by exploring Jenova Chen's [2] approach to applying Flow to video games, where the author differentiates between two approaches for tackling the problem of matching challenges to player ability, active and Passive Flow adjustment, which are concepts particularly relevant, since comparing the two approaches is what motivates this work. We then detailed Chen's game design methodology for widening the range of players that will experience Flow. We'll be applying it to the version of our game with Active Flow Adjustment, as it focuses on this approach, and lastly, we explored the Game Flow Questionnaire [3],

a useful tool for evaluating games in how they deal with Flow. This tool's not used for evaluating player experience, so it won't be used for comparing our two versions of the game. however, it will prove more useful, as a guideline for designing Flow into the testbed game.

Then, in section 2.2, we explored a PCG work by Togelius [4] where the author defines a taxonomy for structuring and highlighting the differences between different PCG approaches. According to this taxonomy, our work can be defined as Online, since it will generate will occur in real-time, Necessary, since the content being generated(spawning enemies) is essential for the game, and Adaptive since the generated content will be adapted depending on player behavior.

In section 2.3, we looked at several approaches for dealing with game difficulty, both in commercial games and in academia. Some approaches that are important for this work include mechanic-driven difficulty and dynamic difficulty adjustment.

The player-choice-driven difficulty will be used for the Active Flow Adjustment version of our game. Our approach for this version will also be similar to the approach used in Diablo 3[10], with the difference that, in the Diablo 3[10] the player needed to pause the game, and couldn't adjust it in the middle of a level. And in our version the player will be able to make the adjustments, at any time, has he plays, with no need for interruptions that can take him out of the Flow state.

Dynamic Difficulty Adjustment will be used for Passive Flow Adjustment, we'll be using a similar approach to the dynamic difficulty adjustment used by Smeddinck et al [8], in their DDA version of Flow. Where the difficulty will go up over time, and then be adjusted back down if the player is struggling. Smeddinck et al [8]'s work is interesting in the context of our work since they were trying to answer a similar question to ours. We'll want to try answering this question with a different game, and see if we get the same results.

Lastly, in section 2.4, we discuss different approaches for evaluating player experience, so that we can compare player experiences in the version of our game. Since Flow is an important component for us we decided to use GEQ [10] for our evaluation.

3

Case Study

Contents

In this chapter we'll discuss the Implementation of the testbed game, ClusterTechRush. We'll explain the game's mechanics, components and how they differ in the two versions of the testbed game, as well as explaining how logging player data was implemented. The game was developed in Unreal engine[1]. Unreal engine was selected because it's a powerful game engine that widely used in game's industry.

3.1 ClusterTechRush

ClusterTechRush is a 3D top down shoot'em up game, where the player's goal is to get to the last level and beat the final boss. In Every level the enemy's health and damage increases considerably, so, to survive the player must get stronger by picking up power ups dropped from enemies.

The player's power is defined by four main variables, maximum hit points(HP), damage per shot, fire rate and movement speed. Every one of these could be increased by picking the corresponding power up dropped from enemies.

Figure 3.1: Snapshot of ClusterTechRush

If the player gets hit they loose a certain amount of HP, to recover their lost health the player needs pick up health nuggets or maximum HP power ups, both dropped by enemies. Health nuggets heal the player by 5% of the players max HP, meaning has the players max HP increases the relative value of health nuggets wont. Enemies have a 30% chance to drop between one and six health nuggets, meaning that killing an enemy can restore between 5% and 30 % of the players health. If the player is on a level too strong for him at that moment, they have a harder killing an enemy and a harder time recovering their health. However if a player is at a lower level, they can easily kill enemies and quickly recover their health.

Another element that was added to create some interesting situations was a loot crate that when destroyed had a chance to reward the player with power ups and health nuggets.

In order to teach players the basic mechanics of the game, before their first run of the game, players

were be asked to play a tutorial that would teach them about the basic mechanics that are common in both version of the game. The tutorial has floating tool tips that tell player about the different mechanics. First the player is instructed on how to run and dash. Then after making their way through a corridor player is taught how to shoot, and asked to shoot some loot crates that are blocking the path. next they find a shooting gallery with stationary enemies that need to be killed in order to open a door. This also introduces the player to the multiple HP bars since those enemies have different amounts of HP. Next the player is taught about each power up, before reaching a corridor with a turret at the end of it. In order to defeat it the player has to dash trough it's projectiles. After defeating it a door opens revealing a teleporter and the player has to press E to activate it and leave the tutorial.

Two versions of the game were developed one where the player has control over their own progress through the game (Version A) and another here that progress is controlled by the game (Version B). The main differences between the two versions is how the player goes up and down the levels, and, the way death is handled.

In version A active Flow adjustment is implemented. In this version it's up to the player to control their experience, and decide how fast they progress through the levels. In version A, in every level, there is a portal, somewhere on the map, and at any moment the player can enter it in order to move to the next level. At the same time, if at any point, the player wants to move down a level they can press a key to go back to the previous level. Because the player has control over going back, we wanted to increase the punishment of death, so in this versions if you die you loose the power ups that were picked up in the current level. In this version the player can also choose to remain in a level after killing all the enemies, so after a few seconds a new wave of enemies is spawned. In this version it is important for the player to find the sweet spot of difficulty so they can grow in power faster.

Figure 3.2: Snapshot of Version A of ClusterTechRush

Version B is the version with passive Flow adjustment. This version takes the decision of moving

up or down the levels from the hands of the player. In version B, if the player kills all the enemies in a level they automatically moves to the next one, and, if the player dies they go to the previous level. Now, because of the harsh difficulty scaling the player is expected to go back a lot, but we didn't want the player's feel frustrated, so we implemented the safety teleport mechanic to try and minimize that frustration. If the player's health reaches below twenty percent a "Safety Teleport" will start charging, and the player has ten seconds to recover his health points, or he will be teleported to the previous level. The safety teleport will also protect the player activating a shield if they're about to die, never letting the player's health reach below one health point. This mechanic is meant to take death and the negative emotion associated with it out of the equation, and give the player a second chance if they are in fact strong enough to be at his current level. The only way for the player to recover health is by picking up health nuggets dropped by enemies when killed. So, if the player is strong enough to kill enemies in the level they are at, in less than 10 seconds, they can continue in that level, but if they're not strong enough they'll be moved to the previous level. Another difference between the two versions is the death penalty, because the player in this version has no control over is progress we do not punish "death" has much, so the player may loose progress by going down a level but they will not loose progress in power up levels, meaning when they go down a level they'll be stronger than they were the last time they were there. In this version by going up a level when killing all enemies and by going down on death, brought the player closer to the sweet spot for picking up power ups.

Figure 3.3: Snapshot of Version B of ClusterTechRush

Another way the two versions differed was in the UI. Both version showed the player, how much hit points (HP) they had, how much power ups they had picked up so far, what level the player was currently on, how much time had passed since the beginning of the run and the dash's cooldown see fig 3.2 and 3.3. In both version the HP bar expands as the player picks up max HP power ups. We tried implementing a subdivision system but ultimately it wasn't very readable once the bar reached it's

maximum size, so we implemented a simple text displaying the current and max HP of the player.

In version A we displayed additional information about the cooldown for going back, this also served has a reminder of which key to press. And the game displays not only the number of pickups of each type but also how many were picked up during the current level, this shows the player how many pick ups they loose if they die. This can been in figure 3.2

In version B the only addition was a progress bar showing the player how many enemies were left in the current level. Since the player had no control and had to kill all enemies to progress we felt it was important for the player to know how close they were from finish a level. One last detail was that in version B there is a marker on the HP bar marking the 20% mark below which the safety teleport would be activated.

Additionally, in order to explain the different version specific mechanics, the menu has a page dedicated to explaining the versions before the player started playing them (see figures in 6).

3.2 Player Actions

The actions that can be performed by the player are:

- **Running**

  The basic movement in the game, using the WASD keys the player can run around, dodge projectile, or pick up power ups and health nuggets.

- **Shooting**

  Aiming and Shooting by clicking with the mouse is the only way for the player to kill enemies, and destroy loot crates.

- **Dashing**

  The player has the option to dash in the direction he is running by pressing shift or the right mouse button. This is useful, because it gives the player increased mobility allowing the player to quickly move around the map, and, while performing the dash the character can pass through enemies and projectiles allowing it to be used defensively.

- **Version A - Entering Teleporter**

  In version A the player can press E next to a portal to move to the next level.

- **Version A - Going back**

  In version A, has long has it isn't inside the 10 second cooldown after going up a level, the player can press Q go back to the previous level.

[1] "Unreal Engine" Game Engine developed by Epic Games,written in C++ initially developed for 3d FPS Games //todomaybe complete

3.3 Enemies

In ClusterTechRush four types of enemies were implemented, a melee enemy, a ranged enemy, stationary turrets and a final boss. The enemies AI was implemented using unreal engine's behaviour trees. All the enemies have initial stats for health and damage. The way the difficulty scaling was preformed is by having a multiplier for each level that is applied to the enemies health and damage.

- **Melee Enemies**

  Melee enemies run at the player and try to punch them. They start with 50 health and do 10 damage per punch on the first level. After spawning they wander around the map, And only engage combat after seeing the player (fig.3.4).

Figure 3.4: Snapshot of Version A of ClusterTechRush

- **Ranged Enemy**

  Ranged enemies are equipped with an assault rifle and shoot at the player from a distance. Their default health is 100 and they do 20 damage per shot. Their default behaviour is to wander around the map, and engage once the they see the player. After seeing the player they run towards him until they are within firing range, they shoot, and then dodge in a random direction, then they repeat that behaviour (fig.3.5).

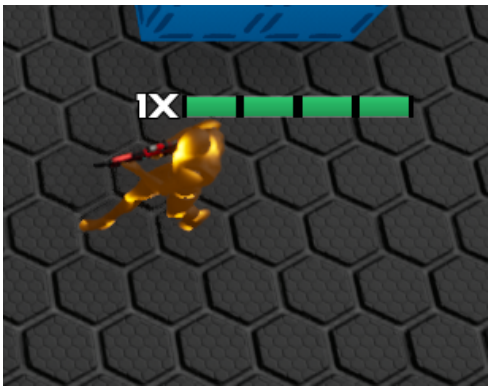


Figure 3.5: Ranged enemy in game

• Turret

Stationary turrets are used to keep the player moving, they shoot the player with homing projectiles that follow the player, forcing the player to either shoot the projectiles or run. Turrets start with 100 health and do 20 damage. After increasing their fire rate player could just stand still and shoot at the turrets without moving. For this reason we implemented splitting projectiles, that split into three projectiles when hit. To maintain the total damage each fragment of the projectile only does a third of the initial projectile's damage(fig.3.6).

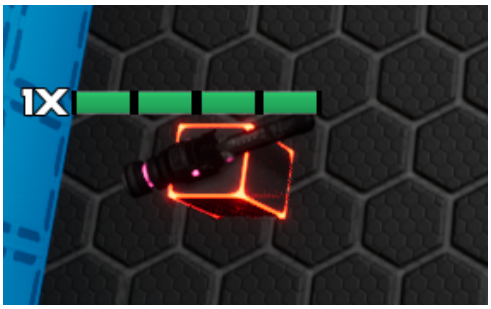


Figure 3.6: Turret enemy in game

• Final Boss

The final boss a big stationary turret that alternates between holding four shotguns(fig.3.7), eight assault rifles(fig.3.8) or activating a shield and spawning regular enemies(fig.3.9). When firing the boss also alternates between two different types projectiles, one always goes straight and can't be destroyed by the player, and another that can slightly curve in the direction of the player, and split when hit like turret projectiles. It has a total of 24000 health which gives it a total of 120 health bars.

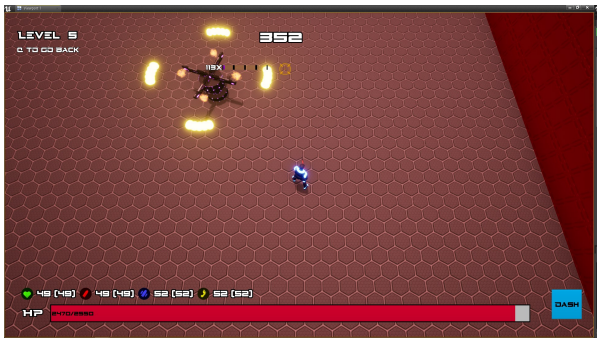


Figure 3.7: Boss in four shotgun mode

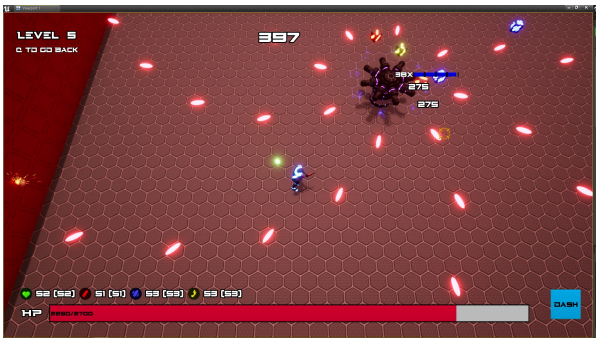


Figure 3.8: Boss in eight assault rifles mode

Figure 3.9: Boss with shield activated

It was important to show players how much stronger enemies got from level to level. We couldn't show the enemies damage just by looking at them, so we did in the form of health bars. The HP multiplier goes from 1, in the first level, to 16, in the fifth level. We needed a flexible way to show how much HP enemies had, so, we implemented a multi bar system that stacks HP bars with different colors on top of each other(similar to what is seen in fighting games). Every bar holds 200 HP and to show how many bars enemies had left a multiplier is shown next to the bar(see figure 3.10).

Figure 3.10: Multi HP bars colors and and numbers

3.4 Power Ups

The power ups are an important part of the game, as the player needs to became stronger if he wants to reach higher levels. Their are four power up types which increase a different variable each with their

own icon and color (see figure 3.11), these icons and color are then matched on the counters in the UI. To create a sense of emergency power ups disappear after 10 seconds, creating tense moments where the player may have to risk walking into heavy fire to pick up power ups before they disappear.

- **Max Health Power Up**

  Increase the player's Max HP by 50.

- **Damage Power Up**

  Increase damage by 5%.

- **Fire Rate Power Up**

  Increase Fire Rate by 5%.

- **Movement Speed Power Up**

  Increase Movement speed by 2%.

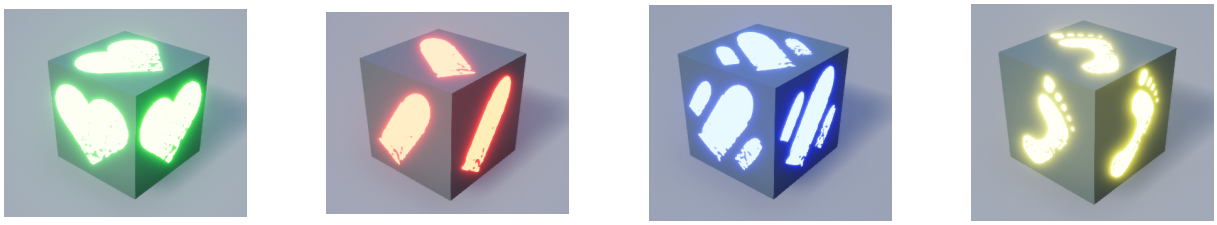


Figure 3.11: All Power Up models, Max Health, Damage, Fire Rate and movement speed respectively.

To reward player for surviving and killing enemies on higher levels the higher the player is the better drop rates are. The drop rates are defined with a float flooring this value gives us the number of pick ups always dropped by any enemy killed on that level. And the remaining value is the chance to drop an extra power up. Level 1 has a 0.5 drop rate, so there was a 50% chance an enemy drops a power up. Level 2 has a drop rate of 1 so enemies always drop one power up. Level 3 has a 1.5 drop rate, so player always drop at least one power up but has 50% chance to drop another one. This pattern was repeated on the remaining levels that have drop rates of 2 and 2.5 respectively. This drop rates did not affect health nugget drop rate, that is always 30%.

3.5 Gameplay loop

The game has a total of 5 levels and the goal of the player is to reach the last one and defeat the final boss. As the player goes up the levels the enemies grow increasingly stronger, but, if the player is able to survive the amount of power ups dropped, by enemies, also increases significantly. This means that at every moment, depending on the players skill and number of power ups, there is a level that maximizes the amount of power ups gained. If the player is in a lower level, for their power and skill, they could probably kill enemies fast but overall he won't pick up as much power ups. At the same time if the player

is at a higher level, they need to kill enemies fast enough to make the higher drop rates worth it. This control over the current level of difficulty was the main difference between the two versions of the game:

- **Version A Game Loop**

  In Version A the game loop consists of killing enemies to pick up power ups and go up the levels while at the same time managing the challenge by choosing in which level to be at. The player enters a level, and at any point can choose to keep fighting, run to the teleporter to move to the next level or press the go back key to move to the previous level. This decision is affected by factors such has their current health, how much damage they are doing to enemies in this level, how many power ups they have picked up so far. If deciding to keep fighting the player will be both shooting and dodging projectiles while trying to pick up power ups and health. After killing all enemies on a level the player has choice stay and wait 10 seconds for the next wave of enemies or move to the next level. With this choices the player is able to manage their challenge, find their own sweet spot, and progress through the game at their own pace.

- **Version B Game Loop**

  In version B, the player don't have control over going up and down the levels, so, the game loop consisted in killing all enemies if they are strong enough to do it and moving to the next level, or dying and moving the previous level. While fighting the player has to shoot at enemies while dodging their projectiles and trying to pick up power ups. If the player's HP gets low enough to trigger the safety teleport they have 10 seconds to recover their health back up to 20% or more. This creates intense moments where the player need to be strong enough to kill enemies, but also be lucky to get enemies drop enough health nugget or max HP power ups. If the player fails to recover their HP they are moved to the previous level but keep all their power ups. Making it easier to climb up the levels again to where they were previously.

  A media folder was created containing video footage of gameplay and other media [2] .

3.6 Architecture

The game was developed in unreal engine, using the unreal engine game framework. In this framework the Game Mode class is responsible for controlling the game rules and getting the game ready to play, this includes spawning the player, enemies, setting up the level, transitioning between levels, and more. We extended from this class and created our own game mode class were we defined all the logic that was common in both versions of the game. Then, we extended from this class and created two one game mode class for each of the versions.

[2] https://drive.google.com/drive/folders/1gqkheS7Bd4uv5EkCLY2YnqwptnWnQDEw?usp=sharing

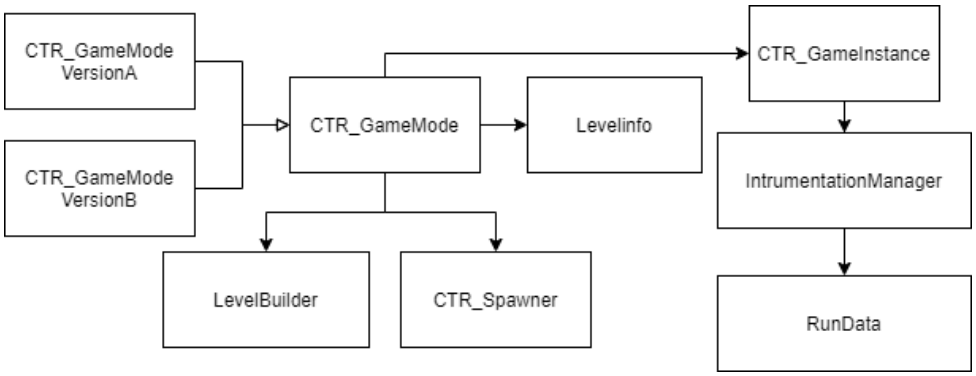


Figure 3.12: ClusterTechRush UML Diagram

Another important class in the unreal engine framework is the game instance. The game instance class is the only class that persists after switching or reloading levels. For this reason it used it to store information about the current state of the player when changing levels. All levels, apart from the last one, consisted of the same square arena, with a different randomized layout. So we decided to just use the same map for the first 4 levels. When starting a new level the game mode class would load from the game instance all necessary information about the current level, and then, spawn the player enemies, loot crates and level layouts.

Spawning was done through the a spawner class that uses unreal engine's built in Environment Query System(EQS) to select spawn location. The environment query system allows us to query possible spawn locations an select from a random subset that fits rules set by us. For example, for spawning an enemy all possible location would be ranked based on distance to the player and then one of the furthest location would be selected. This ensured that enemy would not spawn next to the player. The same thing was done for teleporters in version A, because we didn't want the player to see the teleporter as soon as they spawned.

Initially the game was to have different kinds of weapons, so we also developed a flexible weapon class that could be parameterized to create any kind of weapon. With it we created the assault rifle that is used currently in the game, along with a pistol, a burst firing sub machine gun and a shotgun. The player started with a pistol and then could pick up new weapon from loot crates. Later we decided to simplify the experience and that the player would only have access to the one weapon so that everyone had the same experience.

During development we used git[3]has our version control software, and github[4]to host the project. The source code for the game is still available at `https://github.com/Drannnor/ClusterTechRush`.

[4]Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

[4]GitHub, Inc. is a provider of Internet hosting for software development and version control using Git. It offers the distributed version control and source code management functionality of Git, plus its own features.

3.7 Levels

To help us during development we built a flexible tool to help parameterize level construction. Each level is defined by a LevelInfo structure containing all the information needed to start the level. This information is then utilized by the game mode class when initiating a level. The structure holds information about, what enemies to spawn, the multipliers for health, damage and drop rate, for that level, the level color and in case we wanted to implement levels in different maps, the name of the map to spawn instead of the default. This last point is used to spawn the last level. Initially we had a fourth layout, but through player testing we observed that in some rare occasions player were spawning inside the layout so we decided to remove it.

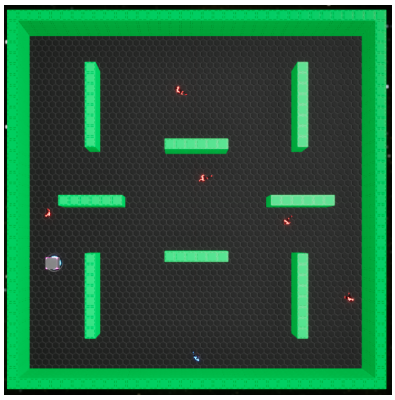


Figure 3.13: Level Layout 1 With level 1 colour

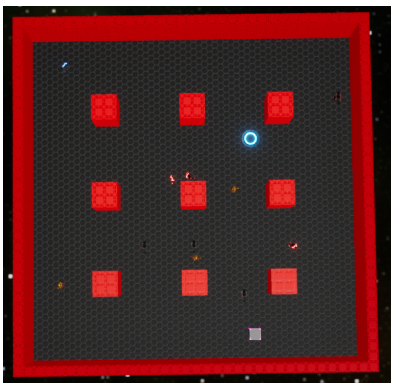


Figure 3.14: Level layout 2 with level 4 colour

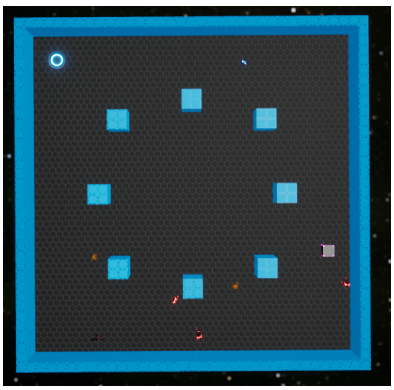


Figure 3.15: Level layout 3 with level 2 colour

Level colors were implemented after feedback from player about it being difficult to know on which level they were currently. The colors followed the same order has the HP bar colors, so the first 4 levels were colored green, blue, purple and red. And finally the last level didn't use the automatic color changes because it was a different map but we decided on a darker shade of red.

Players had to play the game twice and we didn't want it to feel repetitive. So, for each level a random layout of walls and pillars was selected. Since players would be going up and down the levels it was important to keep the layout consistent, so, when starting a run a random layout would be selected from a pool for each of the levels, and that would be stored on the LevelInfo structure. Then, when starting a level the correspondent layout would be spawned.

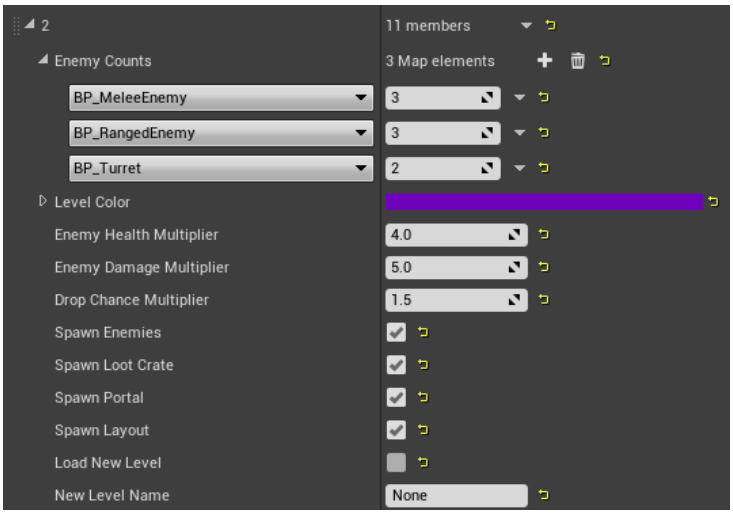


Figure 3.16: Level info in unreal engine, level 3 example

We wanted to have a different level in the end, so the player knows that they are close to the finish. So a final level was designed. Instead of a square arena with a random layout of obstacles, like the other levels, in the final level the player starts at the start of large corridor with pillars along side it. After defeating the enemies in their way, and getting to the end of the corridor, the player finds the final boss arena. And, after entering it two big wall shield activate, locking the player inside the arena with the boss. After defeating the boss the shields deactivate, and give the player access to the final teleport.

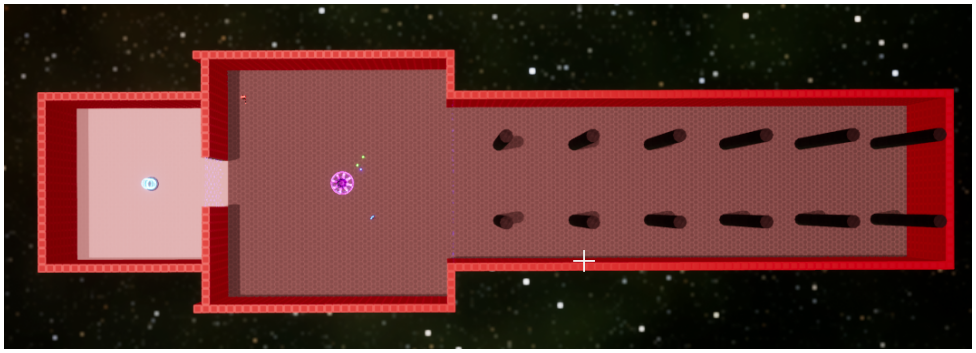


Figure 3.17: Final level top view

3.8 Balancing

A lot of variables needed to be decided in order to balance the game. Namely, player damage and health, enemy damage and health, enemy health and damage multipliers for each of the levels, drop chances for pickups for each of the levels, and finally the effect pick ups would have on player stats. We decided to simplify the problem and started by focusing on health and damage since we could look at the

problem trough the metric of the number of hits it would take to kill an enemy, or the player. After having a general idea of how many hits to kill we wanted for each of the levels we then looked at balancing the power ups. Following the same logic we started by focusing the damage, and maximum HP power ups. This helped us decide how many power ups players were expected to get per level, and much of an impact they would have. After having damage and max health we then balanced the movement speed power up and the fire rate power up so they were on par with the other two power ups.

From here we had a general idea of the values we wanted and then after some player testing we decided on the values that are currently in the game. Three tests were done with 3 to 4 players each, where players were asked to play one of the versions of the game, while we observed, and took notes on feedback.

3.9 Instrumentation

We implemented instrumentation into the game that was used to gather more data from players runs, this data was used for balancing or for the evaluation. Has discussed before the only class that has a persistent instance trough out the game is the game instance, and that is where we stored the instrumentation data during play. The game instance stored a pointer to an instance of the InstrumentationManager. A class responsible for storing all the data and then export it to JSON once the player finished the run. All inputs into the instrumentation manager were done through the game instance class.

The data collect consisted of:

- Final Score;
- Final totals of each power up;
- Number of power ups (total and per level);
- Time (total and per level);
- Damage dealt (total and per level);
- Damage received (total and per level);
- Health recovered (total and per level);
- Kill count (total and per level);
- Death Count (total and per level);
- Version A only - Number of times ”Go back” was pressed (total and per level);
- Version A only - Enemy wave respawns (total and per level);

- Version B only - Number of times Safety teleport was activated (total and per level);
- Version B only - Number of times Safety teleport recovered (total and per level).

To facilitate working with the data collected, a tool a simple script was developed that would receive has input both the log files from participants and the .csv exports from the forms, parse both and format the data in a way that was easy to import into SPSS.

3.10 Summary

In this chapter we described the game developed to server our research ClusterTechRush, it's components, and how they interacted with each other, as well as the development challenges they we faced. We go into detail about the players action, enemies and power ups. We explore our approach for implementing both Active and Passive Flow adjustment in each version. We explore the architecture, the tools and technologies that were used during the development of ClusterTechRush, including flexible tool that helped us easily tweak level parameters.

Lastly, we described how the steps taken to ensure the game was balanced and the instrumentation approach that we implemented in order to store extra data about players experiences.

4

Evaluation

Contents

In this chapter we will describe the evaluation process, describing the two evaluation phases, the experimental process and analyzing the results obtained.

4.1 Preliminary Evaluation

4.1.1 Objectives

A preliminary evaluation was run to make sure that every aspect of the experimental procedure was working as intended. This included making sure the instructions in the forms were clear and the participants knew what to do, and that the game was working properly. As for the game we wanted to make sure it was running smoothly with no game-breaking bugs or crashes, and make sure participants were understanding what they had to do. At this point, a small tutorial had been developed but it hadn't been tested yet.

4.1.2 Procedure

In order to test which difficulty adjustment would create a better experience, we asked participants to play the two versions of ClusterTechRush. The players were asked to fill in a questionnaire, before playing any of the versions, where we collected demographic information, and then after playing each of the versions the players were presented with the in-game QEG questionnaire. Two forms were created one in which participants were asked to start with version A and another where players were asked to start with version B.

The form (see chapter 6) started with a quick description of what was going to be asked of the participants and how long it would take. Followed by a link to where to download the game.

Next, we asked the participants to fill in the demographic section. In the demographic component of the questionnaire, we inquired participants about age, gender, how frequently they play video games, how familiar they were with the genre of game played in the experience, how comfortable they were playing this type of game with a keyboard, and mouse, and lastly we add a section measuring Locus of Control.

After this section players were instructed on how to extract and get the game ready to play. With the game ready players were asked to play the tutorial, so they got used to the game's mechanics, and then play one of the versions. After finishing they had to answer the in-game GEQ, and then do the same for the second version. After finishing both versions the players were asked to evaluate each of the versions from 1 to 7, and if one had a higher score than the other they were asked why. Finally, the players were asked to upload the log file created by the game, containing the logged data about both runs.

4.1.3 Results and changes

From the preliminary evaluation, we didn't find any major problems with the forms, only minor text fixes were needed. However, we did find a few problems with the game.

Apart from a few small bugs that were easy to fix, we noticed that players weren't leaving the tutorial with a full grasp of the game mechanics. Namely, some players were playing without ever realizing they had a dash mechanic. For this reason, we extended the tutorial with a section where a turret would fire at the player from the end of a narrow corridor, forcing the player to dash through the projectiles to kill it.

We also noticed that the more casual players were finding the game too challenging, for this reason, we decided to make health drops move slowly towards the player. Meaning players still might have to put themselves in high-risk situations to grab power-ups but health drops would be safer to pick up.

Finally, in version A, we implemented a cooldown for going back after going up a level, to solve the problem of players exploiting the fact that moving up a level would make the power-ups you picked up during the previous level persistent. This way the players at least needed to survive 10 seconds in the harder level to save their progress.

4.2 Final Evaluation

4.2.1 Objectives

With this experiment, we wanted to help answer the research question presented in this document, what produces a better game experience? Active or Passive Flow adjustment? How important is player control over a game's difficulty for maintaining the Flow state?

4.2.2 Procedure

From the preliminary evaluation, no major problems were identified with the experimental procedure, and only minor text fixes were implemented. Because of this, the experimental procedure for this phase of evaluation was identical to the one described in chapter 4.1.2.

4.2.3 Sample

In total, we collect data from 23 participants. Of these 23, 9 started with versions A, and 14 started with version B. The majority of participants were male and mainly 24 and 25 years old, as can be seen in figure 4.1 and figure4.2. We had a total of 20(89.96%) male participants and only 3(86.96%) female participants. Participants' ages ranged from 21 to 32 but around 65% had either 24 or 25 years of age.

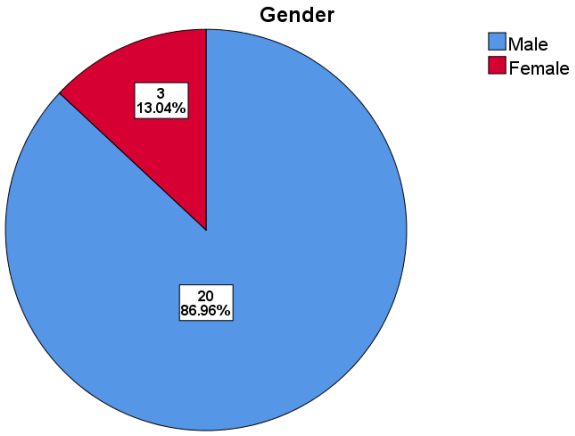


Figure 4.1: Demographics: Gender

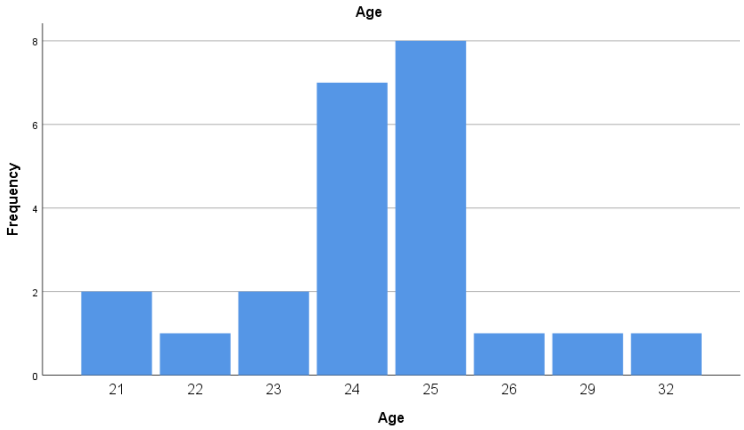


Figure 4.2: Demographics: Age

In order to differentiate casual players from more experienced players, we asked participants how frequently they played video games. They could answer with one of 3 options:

- I make some time in my schedule to play video games.
- I play video games occasionally when the opportunity presents itself.
- I very rarely play video games.

The results can be seen in figure 4.3. 12(52.17%) participants said they made time in their schedule to play video games, 10(43.48%) participants said they played video games occasionally, and only 1 (4.35%) said they rarely played video games.

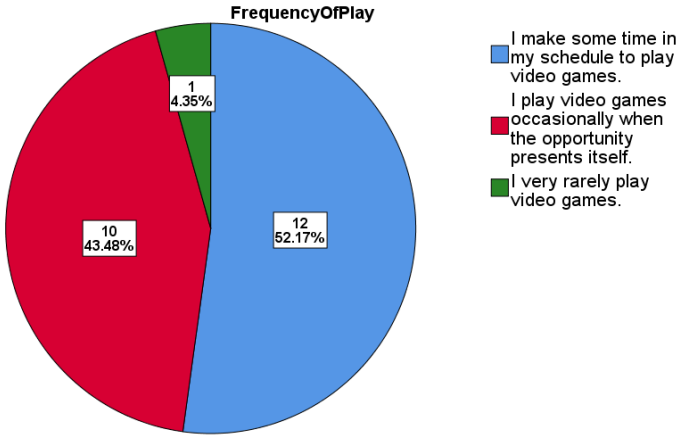


Figure 4.3: Demographics: Frequency of Play

Next, we wanted to know how familiar the participants were with similar games to ClusterTechRush. We asked if they enjoyed Top Down Shoot'em Up Games, and gave Enter the Gungeon[1] and Nuclear Throne[2]has examples. Participants could choose one of 3 options:

- I played/watched others play them enough to understand I do not appreciate them.
- I enjoy them and have played/watched others play them multiple times.
- I am not familiar with these games and/or have no formed opinion on them.

13 (56.52%)participants said they knew and enjoyed playing or watching the games, 8 (34.78%) said they didn't know the games, and 2 (8.70%) said they had played or watched but didn't enjoy the games.

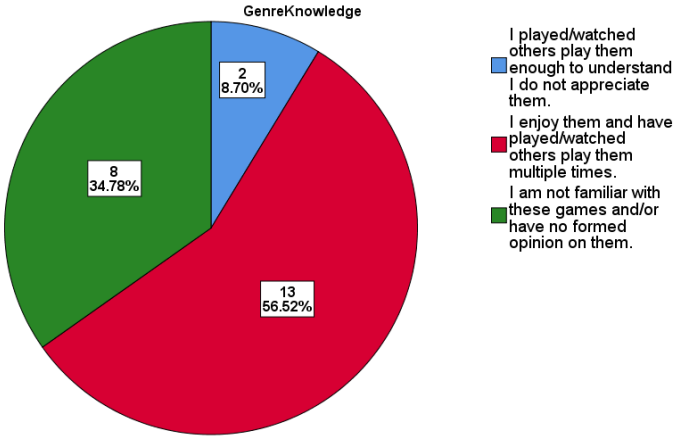


Figure 4.4: Demographics: Genre Knowledge

Next, we asked participants to choose between one and five, how comfortable they were with playing a Top-down shooter with a mouse and keyboard. Has seen in figure 4.5 most participants felt comfortable

[1]Enter the Gungeon, Dodge Roll, PlayStation 4, Xbox One, Nintendo Switch, macOS, 2016
[2] Nuclear Throne, Vlambeer, PlayStation 4, Microsoft Windows, Linux, macOS, 2015

using a keyboard. Of the 23 participants, 10 (43.48%) answer 5, 4 (17.39%) participants answered 4, 8 (34.78%) answered 3, and only one participant answered 2.

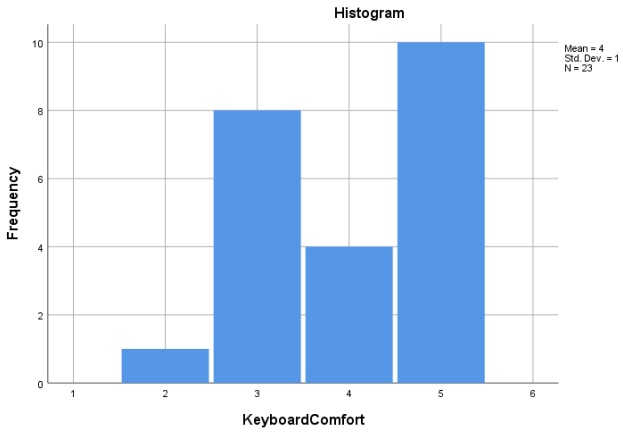


Figure 4.5: Demographics: Comfortable using a keyboard from 1-5

Lastly, participants were asked to answer a short locus of control section, participants had to answer 4 questions and with the results, we would calculate a total score that would have a value between -2 and 2. The following figure shows the distribution of locus of control scores of all participants.

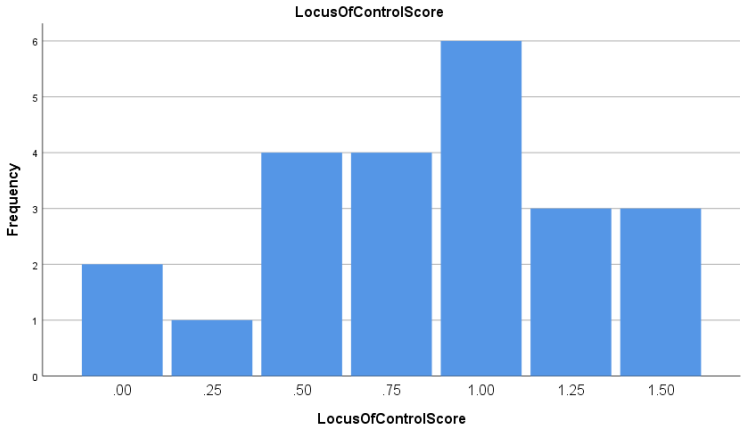


Figure 4.6: Demographics: locus of control scores

Overall we were please with the scope of the demographic sample. Despite participants being mostly male and between the ages of 24 and 25, we ended up with a good distribution in terms of frequency of play, genre knowledge, keyboard comfort, and locus of control.

4.2.4 Results

Since we only had 23 participants we used a non-parametric Wilcoxon test between two samples, to detect significant changes between the two versions of the game. We started by comparing each of the scores from the seven GEQ components between versions A and B, and the final scores attributed by the participants for each version. For the Wilcoxon test, two samples are statistically different if the row Asymp. Sig. (2-tailed) has a value of 0.05 or lower. We then repeated the test for different subsets according to demographic data. We tested players that play games frequently vs those who do not, we tested the group of participants who were familiar and liked top-down shooters against the group participants who either didn't like the genre or didn't know anything about it and, we tested participants based on how comfortable they felt playing with a keyboard and mouse, one subset with player how had selected 3 or less on the keyboard comfort question and another subset of players who selected more than 3.

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositiveScore - AGEQPositiveScore	VersionBScore - VersionAScore
Z	-1.961[b]	-.506[c]	-.461[b]	-2.481[c]	.000[d]	-1.149[c]	-2.442[b]	-.076[c]
Asymp. Sig. (2-tailed)	.050	.613	.645	.013	1.000	.250	.015	.939

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

d. The sum of negative ranks equals the sum of positive ranks.

Figure 4.7: Results from Wilcoxon test for all GEQ scores and versions final score

As can be seen in figure 4.7 no significant differences were found between QEG Flow scores (Z = -0.461, p = 0.645) or finals scores (Z = -0.076, p = 0.939) between the A and B. But we did find statistically significant differences in Competence(Z = -1.961, p = 0.050), Tension(Z = -2.418, p = 0.013) and Positive Affect(Z = -2.442, p = 0.015).

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositiveScore - AGEQPositiveScore	VersionBScore - VersionAScore
Z	-2.388[b]	-.718[c]	-1.058[b]	-1.955[c]	-.574[c]	-.862[c]	-2.549[b]	.000[d]
Asymp. Sig. (2-tailed)	.017	.473	.290	.051	.566	.389	.011	1.000

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

d. The sum of negative ranks equals the sum of positive ranks.

Figure 4.8: Results from Wilcoxon test for players that don't play games frequently

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositiveScore - AGEQPositiveScore	VersionBScore - VersionAScore
Z	-1.380[b]	-1.318[c]	-.061[b]	-2.095[c]	-1.279[c]	-.595[c]	-1.208[b]	-.540[c]
Asymp. Sig. (2-tailed)	.168	.187	.952	.036	.201	.552	.227	.589

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.9: Results from the Wilcoxon test for the subset of players that either didn't enjoy the game genre or didn't know about it

The competence score mean for version A was 2.022 and for version B it was 2.413. Showing that, overall, players felt more competent while playing version B. We then retried the same Wilcoxon test but with different subsets of players. We observed that for more experienced players the difference in competence score was not felt. The difference in competence score was only present for more casual players(Z = 0.017, p = -2.388) fig. 4.8 with whom we observed an average competence score in version A of 1.773 vs a mean of 2.455 in version B. Meaning casual players felt more competent in version B.

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositiveScore - AGEQPositiveScore	VersionBScore - VersionAScore
Z	-1.292[b]	-.586[b]	-.540[b]	-1.429[c]	-1.140[b]	-.997[c]	-2.326[b]	-.511[b]
Asymp. Sig. (2-tailed)	.196	.558	.589	.153	.254	.319	.020	.609

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.10: Results from the Wilcoxon test for the subset of players that were more familiar with the game genre

For the tension score in version A, we found a mean of 1.587 and a mean of 0.935 in version B, showing, overall, players felt tenser while playing version A. We didn't find a significant difference in tension score between casual and experienced players but we did see a difference when testing the genre familiarity and keyboard comfort. The test didn't show a statistical difference for participants who were familiarized with this game genre, but for participants who either didn't know the genre, or didn't like it we observed (Z= 0.036, p =-0.933) fig.4.9 a mean tension in version A of 1.759, and of 0.900 in version B. For keyboard comfort the more comfortable data set didn't show significant differences in the tension score, but, in the less comfortable subset, we observed a significant difference in tension(Z = 0.28, p = -2.200) fig.4.11. The mean in version A was 1.722 and 0.778 for version B.

In the positive effect component, the means for versions A and B were 2.196 and 2.609 respectively. Meaning, overall, version B elicited more positive emotions. Looking at the subsets we observed significant differences in the positive effect component of GEQ, in the frequency of play, genre familiarity, and keyboard comfort. In frequency of play we only observed differences for the more casual players (Z =

0.011, p = -2.549) fig.4.8 with positive affect score means of 1.864 and 2.591 for versions A and B respectively. When separating participants based on genre familiarity we observed no difference between participants less familiar with the genre but we did observe it with the more familiarized participants(Z = 0.020, p = -2.326) fig.4.10 with mean of 2.500 and 2.962 respectfully. And in Keyboard comfort, we observed a difference for participants that were less comfortable with a keyboard and mouse (Z = 0.31, p = -2.157) fig.4.11 with means of 1.667 and 2.278 for version A and B respectfully. This leads us to believe, that more casual players, who are also less comfortable with a mouse and keyboard, had a better experience in version B.

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositiveScore - AGEQPositiveScore	VersionBScore - VersionAScore
Z	-1.913[b]	-.791[c]	-.933[b]	-2.200[c]	-1.802[c]	-.954[c]	-2.157[b]	-.342[c]
Asymp. Sig. (2-tailed)	.056	.429	.351	.028	.072	.340	.031	.733

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.11: Results from Wilcoxon test for the subset of players that felt less comfortable playing a top-down shooter with a keyboard

Since players had to play two versions, one after the other, we wanted to know how much of an impact having already completed one run of the game had on a second run with a different version so we separated the GEQ scores and final score of the 2 versions into the first run and second run, independently of version. We ran the Wilcoxon test again with these variables and observed that overall the order in each of the players who played the 2 versions had no effect see fig.4.12. However, after checking the subsets of participants who started with version A or with version B we can see that participants who started with version A report significantly higher positive emotion when playing version B(Z = 0.31, p = -2.157) see fig.4.13. On average participants who started with version A reported a positive effect score of 2.100 and then an average score of 2.650 in version B.

	SecondRunScore - FirstRunScore	SecondGEQCompetenceScore - FirstGEQCompetenceScore	SecondGEQSensoryScore - FirstGEQSensoryScore	SecondGEQFlowScore - FirstGEQFlowScore	SecondGEQTensionScore - FirstGEQTensionScore	SecondGEQChallengeScore - FirstGEQChallengeScore	SecondGEQNegativeScore - FirstGEQNegativeScore	SecondGEQPositiveScore - FirstGEQPositiveScore
Z	-1.356[b]	-1.096[c]	-1.637[b]	-.210[b]	-.018[b]	-.628[b]	-.718[c]	-.426[b]
Asymp. Sig. (2-tailed)	.175	.273	.102	.834	.986	.530	.473	.670

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.12: Results from Wilcoxon test comparing first to second run results

	SecondRunScore - FirstRunScore	SecondGEQCompetenceScore - FirstGEQCompetenceScore	SecondGEQSensoryScore - FirstGEQSensoryScore	SecondGEQFlowScore - FirstGEQFlowScore	SecondGEQTensionScore - FirstGEQTensionScore	SecondGEQChallengeScore - FirstGEQChallengeScore	SecondGEQNegativeScore - FirstGEQNegativeScore	SecondGEQPositiveScore - FirstGEQPositiveScore
Z	-1.098[b]	-.351[b]	-1.000[b]	-.577[b]	-1.862[c]	-.425[b]	-1.382[c]	-2.157[b]
Asymp. Sig. (2-tailed)	.272	.726	.317	.564	.063	.671	.167	.031

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.13: Results from the Wilcoxon test comparing first to second run results applied only to participants who started with version A

The locus of control score had a median of 1.0 so we split the data set into one with participants with locus of control score of less than 1 and another with participants with a score greater or equal to 1. Running the Wilcoxon test again against the GEQ components and final score showed no differences for the subset with a high locus of control, but it did show significant statistical changes for the competence (Z = 0.039, p = -2.066) and positive affect scores (Z = 0.009, p = -2.623) see fig4.14, in the low locus subset. For the competence score, we observed a mean of 2.059 for version A and of 2.500 for version B. And for positive effects, we observed 2.147 and 2.647, for versions A and B respectively. This shows that players with lower locus of control felt more competent and had more positive emotions while playing version B.

	BGEQCompetenceScore - AGEQCompetenceScore	BGEQSensoryScore - AGEQSensoryScore	BGEQFlowScore - AGEQFlowScore	BGEQTensionScore - AGEQTensionScore	BGEQChallengeScore - AGEQChallengeScore	BGEQNegativeScore - AGEQNegativeScore	BGEQPositive Score - AGEQPositive Score
Z	-2.066[b]	-.212[b]	-.827[b]	-1.900[c]	-1.063[b]	-.930[c]	-2.623[b]
Asymp. Sig. (2-tailed)	.039	.832	.408	.057	.288	.352	.009

a. Wilcoxon Signed Ranks Test

b. Based on negative ranks.

c. Based on positive ranks.

Figure 4.14: Results from the Wilcoxon test applied to participants with less than 1 for the locus of control score

Finally, we exploring the in game logged data we used the spearman rank-order correlation coefficient to look for correlations between the different logged variables and the GEQ scores for each of the versions. We repeated the test for, total time, total kills, total deaths, power up total, total damage dealt and total damage taken.

When preforming the tests with version B, we found a negative correlation between the kill count and the final score given by the participant($r_s(20) = -.494$, $p = .027$), and another between total death count and final score ($r_s(20) = -0.527$, $p = 0.017$). The more a participant died in version B the more kills he would get since they had to repeat levels, and the more a participants died the lower the final score appears to be.

In Version A we found a few more interesting results. When running the spearman test with total time

we observed a positive correlation total time and tension ($rs(18) = 0.751$, $p = 0.00$), another between total time and challenge ($rs(18) = 0.517$, $p = 0.28$), and another between total time and negative emotions ($rs(18) = 0.667$, $p = 0.002$). Additionally we found a negative correlation between total time and positive effects ($rs(18) = -0.631$, $p = 0.005$), and between total time and the final score ($rs(18) = -0.561$, $p = 0.028$). This seems to show that in version A the longer the run would be, the participants would feel tenser, more challenge, experience more negative emotions and less positive emotions which resulted in the participant giving this game mode a lower score.

4.3 Discussion

In this chapter, we described the evaluation process of this work. We started by describing the preliminary evaluation, the feedback we got, and how it affected the final evaluation. In the final evaluation section, we described the experimental procedure, including what was expected from the participants and what questions were asked.

We had a total of 23 participants, and, even though most of them were male and between 24 and 25 years of age, we had a varied sample when it came to frequency of play, genre knowledge, and keyboard comfort, and locus of control.

After analyzing the data, we did not observe a significant difference, in the GEQ Flow score, nor did we find a difference between the final scores attributed by participants to both versions. However we did find, differences in the competence, tension, and positive affect QEG scores. Overall, players felt more competent while playing version B, felt tenser while playing version A, and experienced more positive emotion while playing version B.

When exploring the subsets of players we observed that more casual participants tended to display a preference for the version where they had less control over the game experience. This could be seen in the group of players that didn't play video games frequently, showing more competence and more positive effects on version B. The subset of players that were not very familiar with this genre, showing more tension in version A. And In the subset of players that were less comfortable with a keyboard, showing more positive emotions in version B and more tension in version A. This seems to show that more casual players might prefer a passive Flow adjustment approach to game difficulty.

As expected, while taking into consideration participant locus of control scores we found that participants with a lower score, meaning they were more prone to believe that they didn't have control over their lives, seemed to have a better experience while playing the version that didn't give them control over their progress.

By analyzing the game logged data we were able to find a few correlations showing that in version B player who died more scored the version less. And, we found several correlation in version A between

tension, total time, challenge, negative effect and positive effect. We found that the longer a player would take to finish version A the likelier it would be for the player to report higher, tension, challenge and negative emotions while a the same time reporting lower positive effect score and lower version A score.

By comparing scores between the first version played and the second version played we found that the order of play didn't seem to make a difference in scores. However, after testing separately participants who played the A version first and then participants that had played version B first. We found having played version B first didn't show any differences, but participants who started with version A tended to report significantly less positive effect scores.

The fact that players seem to have more positive emotions while playing version B after starting with version A, in conjunction with the fact that more casual players seemed to prefer version B, suggests that more casual players might have had trouble understanding the mechanics of the game, namely, the importance of going back, since in most games it is natural to always push forward. We didn't see such a difference with more experienced participants because they quickly get the importance of going back. And we didn't see such a difference in players who started with version B because they were already familiar with the game when playing version A.

5

Conclusion

Contents

5.1 Conclusions

In this document, we explored active and passive Flow adjustment, one approach that gives the player control over the game's difficulty and another where the game has control. We compared the two approaches and tried to figure out which one produced the best experience, and facilitates invoking Flow in games. Using unreal engine 4, we developed, from scratch, two versions of the same game, ClusterTechRush, to support our research.

ClusterTechRush is a 3D top-down shoot'em up game where the player has to beat 5 levels and kill the final boss. From one level to another the enemies get significantly stronger. To survive higher levels the player needs to kill enemies and pick up the power-ups they drop. Two versions of the game were created, version A using active Flow adjustment and giving the player the option to go up and down the levels, and version B, using passive Flow adjustment, where if the player kills all enemies in a level he will be moved to the next one and if the player dies he will go back to the previous one.

To compare both versions of the game, after answering a demographic section of the questionnaire, containing questions about frequency of play, genre knowledge, ease with keyboard control, and locus of control, participants were asked to play both versions and fill in an in-game Game experience questionnaire [10] after each of the version. Players were then asked to give a score from 1 to 7 to each of the versions

The final evaluation showed no significant difference in GEQ Flow score, nor did we find a difference between the final scores attributed by participants to both versions. However we did find, through the Game experience questionnaire, that player felt more competent in version B, tenser in version A and experienced more positive emotions in version B.

After further analyses, we observed that less experienced participants tended to display a preference for the version where they had less control over the game experience. This could be seen in the group of players that didn't play video games frequently, showing more competence and more positive effects in version B. The subset of players that were not very familiar with this genre, showing more tension in version A. And in the subset of players that were less comfortable with a keyboard, showing more positive emotions in version B and more tension in version A. This seems to indicate that Passive Flow adjustment might be a better approach for creating games for more casual players. Implementing an Active Flow adjustment solution requires precise design and balancing especially when trying to accommodate more casual players.

Finally, analyzing locus of control scores we found that participants with a lower score, meaning they were more prone to believe that they didn't have control over their lives, seemed to have a better experience while playing the version that didn't give them control over their progress.

In a game, making a decision to go back is not always natural since for most games players are asked to always push forward in order to overcome challenges. One possible explanation, for the fact that more

casual players seem to be having a better experience with version B, might be that more casual players found it harder to understand that going back is important. And more experienced players picked it up fairly quickly. This hypothesis seems to be further confirmed when comparing participants' first and second runs. Overall, the order in which the version were played didn't seem to have an effect on player experience. But, after analyzing the subset of participants that played version A first we observed that they reported higher positive emotion when playing version B. Further research is needed to understand what makes an active Flow adjustment approach less appealing to more casual players.

Both Active Flow adjustment and Passive Flow adjustment are viable options. We couldn't prove that one is objectively better than the other at creating a good experience. But we did find that Active Flow adjustment might be more appropriate when designing games with more experienced players in mind. Active Flow adjustment is hard to develop and is very much a design problem. The sensible nature of its implementation makes it harder to use when creating experiences for more casual players. Having a greater understanding of both approaches to implement Flow in games would be great for future game developers, so further research is needed on this topic.

5.2 Future Work

To finish, what follows are some interesting ideas on how to expand on our work.

- In our work it looks like some participants might have had trouble understanding the importance of the ”go back” mechanic. Recreating the experience with a better experience or with the player having already played the game before, might help understand if passive Flow adjustment is better for more casual players.
- The experiment showed some interesting results but 23 participants is not a very significant sample size. Recreating the experiment but with a bigger and more diverse group of participants might help get more concrete information about the topic.
- In our approach version B had a very simplistic, hands-off, implementation of passive Flow adjustment. Recreating the experiment with a more advanced passive Flow adjustment implementation might prove important to validate our results.
- We observed more casual players had a worst time with active flow adjustment. Further research could be done to find out what other Active Flow adjustment strategies could work better with more casual players.
- Different game genres attract different types of people, it might be interesting to recreate a similar experiment, comparing both active and passive Flow adjustment versions of another game with a different genre.

Bibliography

[1] Csikszentmihalyi, M.(1990). Flow: The Psychology of Optimal Experience.New York: Harper & Row

[2] Chen, J. (2006). Flow in games. School of Cinematic Arts Los Angeles, USA, University of Southern California. **MFA in Interactive Media.**

[3] Sweetser, P. and Wyeth, P. GameFlow: a model for evaluating player enjoyment in games. Computers in Entertainment (CIE), 3, 3 (2005).

[4] J. Togelius, N. Shaker, and M. J. Nelson, “Introduction,” In Procedural Content Generation in Games: A Textbook and an Overview of Current Research, N. Shaker, J. Togelius, and M. J.Nelson, Eds. Springer, 2016, ch. 1.

[5] J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis, “What is a procedural content generation?: Mario on the borderline,” in Proceedings of the 2nd International Workshop on Procedural Content Generation in Games, ACM, 2011

[6] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey.,” in IEEE Transactions on Computational Intelligence and AI in Games 3, pp. 172–186, ACM, 2011.

[7] G. Smith, J. Whitehead, and M. Mateas, “Tanagra: A mixed-initiative level design tool,” in Proceedings of the Fifth International Conference on the Foundations of Digital Games. ACM, 2010, pp.209–216.

[8] Smeddinck, Jan & Mandryk, Regan & Birk, Max & Gerling, Kathrin & Barsilowski, Dietrich & Malaka, Rainer. (2016). How to Present Game Difficulty Choices?: Exploring the Impact on Player Experience. 5595-5607. 10.1145/2858036.2858574.

[9] J. Catarino and C. Martinho, Holiday Knight: a Videogame with Skill-based Challenge Generation, 2019

[10] K. L. Norman, ”GEQ (Game Engagement/Experience Questionnaire): A Review of Two Papers,” in Interacting with Computers, vol. 25, no. 4, pp. 278-283, July 2013, doi: 10.1093/iwc/iwt009.

[11] McAuley E, Duncan T, Tammen VV. Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: a confirmatory factor analysis. Res Q Exerc Sport. 1989 Mar;60(1):48-58. doi: 10.1080/02701367.1989.10607413. PMID: 2489825.

[12] Johnson, Daniel & Gardner, John & Perry, Ryan. (2018). Validation of two game experience scales: The Player Experience of Need Satisfaction (PENS) and Game Experience Questionnaire (GEQ). International Journal of Human-Computer Studies. 118. 10.1016/j.ijhcs.2018.05.003.

[13] Kovaleva, Anastassiya, The IE-4: Construction and Validation of a Short Scale for the Assessment of Locus of Control, GESIS - Leibniz-Institut für Sozialwissenschaften, https://nbn-resolving.org/urn:nbn:de:0168-ssoar-371199

[14] "GAMES YOU DIDN'T KNOW FEATURED DYNAMIC DIFFICULTY", https://www.svg.com/138490/games-you-didnt-know-featured-dynamic-difficulty/ Nov, 2018, accessed: 23-12-2020

[15] "Gamasutra Cognitive Flow: The Psychology of Great Game Design", https://gamasutra.com/view/feature/166972/cognitive_flow_the_psychology_of_.php, 2012, accessed: 22-12-2020

[16] "Shoot 'em up", https://en.wikipedia.org/wiki/Shoot_%27em_up, accessed: 01-010-202

[17] "Video game market value worldwide from 2012 to 2023", https://www.statista.com/statistics/292056/video-game-market-value-worldwide/ Christina Gough, Aug 28, 2020, accessed: 31-12-2020

6

Questionnaire

Active and Passive Flow in Games

This study is part of the Master Thesis of Alexandre Chícharo at Instituto Superior Técnico and aims to compare two different approaches to control flow in games. This work is supervised by Prof. Carlos Martinho.

You will be asked to play two different versions of a short game, "ClusterTechRush", a top-down shooter, and answer a few questions after each version to provide feedback regarding your play experiences and how they compare. You will also be asked to send us a file containing the logs of your play through the game.
For this experiment you'll need a computer with a graphics card, and a mouse and keyboard. Each version should take about 15 - 20 minutes to complete, and the whole study should take about 35 - 45 min.

Feel free to contact alexandre.chicharo@tecnico.ulisboa.pt for any questions.

*Required

Downloading the game

If you are interested in participating in this study start by downloading the game using the following link, while you read the consent form and answer a few questions about you.
https://github.com/Drannnor/ClusterTechRush/releases/download/v1.22/ClusterTechRush.zip

Informed Consent

All data collected is confidential and will be used exclusively for the purpose of the above mentioned Master Thesis. This data will only be available to the Thesis's Investigator, Supervisor and Colaborators previously mentioned. No third party will have access to this data.

Before agreeing to take part in this study, you must agree with the following statements:

1. I understand that my participation in this study is voluntary and I can quit at any given time, having the data about my experience deleted. *

   Mark only one oval.

   ◯ Yes

2. I authorize the usage of my demographic profile data in this study. *

   Mark only one oval.

   ◯ Yes

3. I authorize the treatment of my data in this study under the conditions above mentioned. *

   Mark only one oval.

   ◯ Yes

4. I agree to participate in this study. *

   Mark only one oval.

   ◯ Yes

By clicking the next button, it is considered that the participant has given his consent to the test that will follow.

Demographic Profile

This section of the questionnaire aims to collect data about you as a person and as a video game player to look into possible connections between who you are and how you felt about the experience.

5. Gender: *

Mark only one oval.

- ◯ Male
- ◯ Female

6. Age: *

_____

7. How frequently do you play video games? *

Mark only one oval.

- ◯ I make some time in my schedule to play video games.
- ◯ I play video games occasionally when the opportunity presents itself.
- ◯ I very rarely play video games.

8. Do you enjoy Top Down Shoot'em Up Games (e.g. Enter the Gungeon, Nuclear Throne)? * *

Mark only one oval.

- ◯ I played/watched others play them enough to understand I do not appreciate them.
- ◯ I enjoy them and have played/watched others play them multiple times.
- ◯ I am not familiar with these games and/or have no formed opinion on them.

9. I am perfectly at ease with using keyboard and mouse to play top-down shooters: *

Mark only one oval.

	1	2	3	4	5	
Totally disagree	◯	◯	◯	◯	◯	Totally agree

10. For each of the following statements, pick the options you agree with the most. *

Mark only one oval per row.

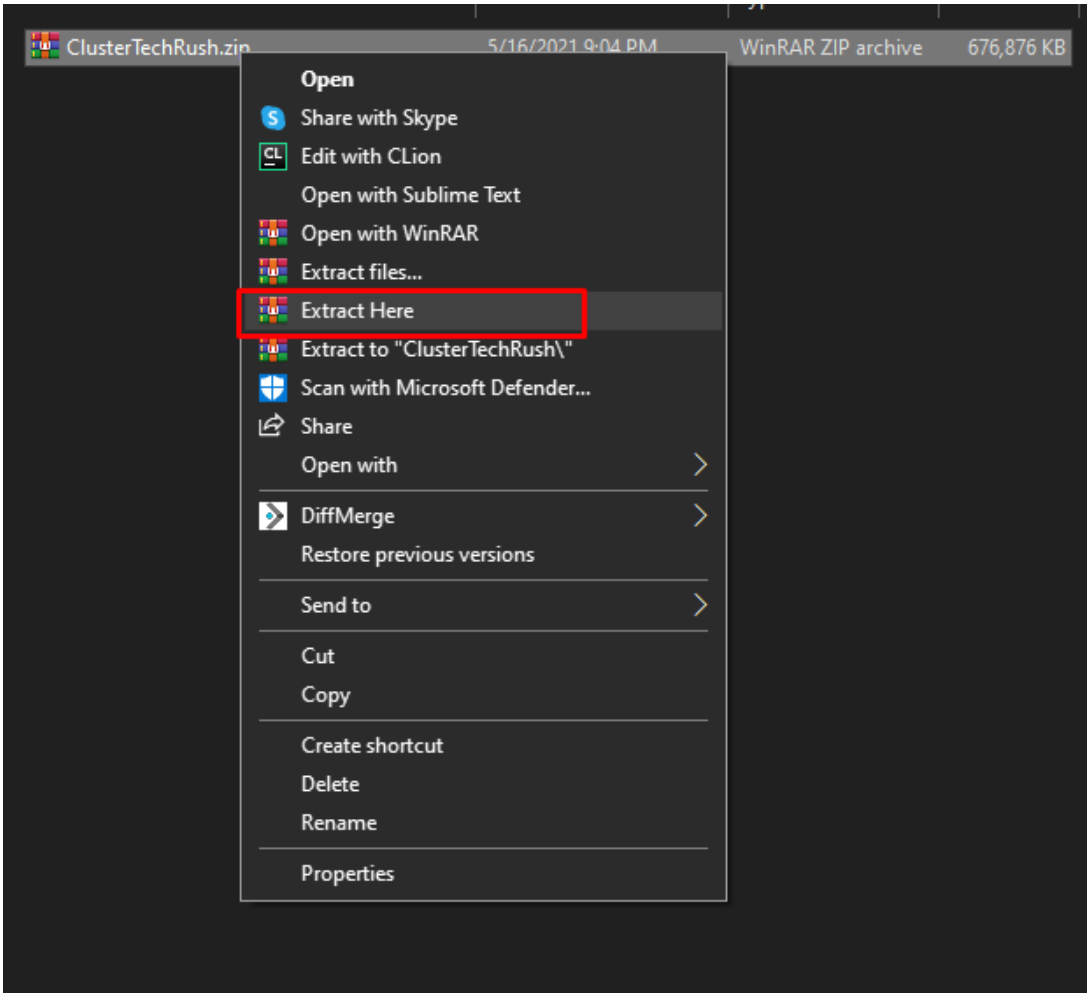
	doesn't apply at all	applies a bit	applies somewhat	applies mostly	applies completely
If I work hard, I will succeed.	◯	◯	◯	◯	◯
I'm my own boss.	◯	◯	◯	◯	◯
Whether at work or in my private life: What I do is mainly determined by others.	◯	◯	◯	◯	◯
Fate often gets in the way of my plans.	◯	◯	◯	◯	◯

Getting Started

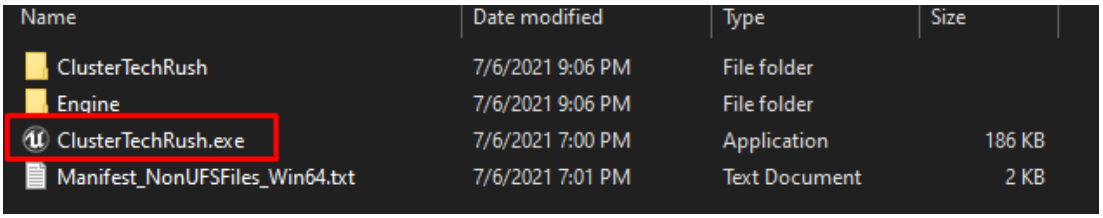
Just in case you haven't downloaded the game yet here is the link again:

https://github.com/Drannnor/ClusterTechRush/releases/download/v1.22/ClusterTechRush.zip

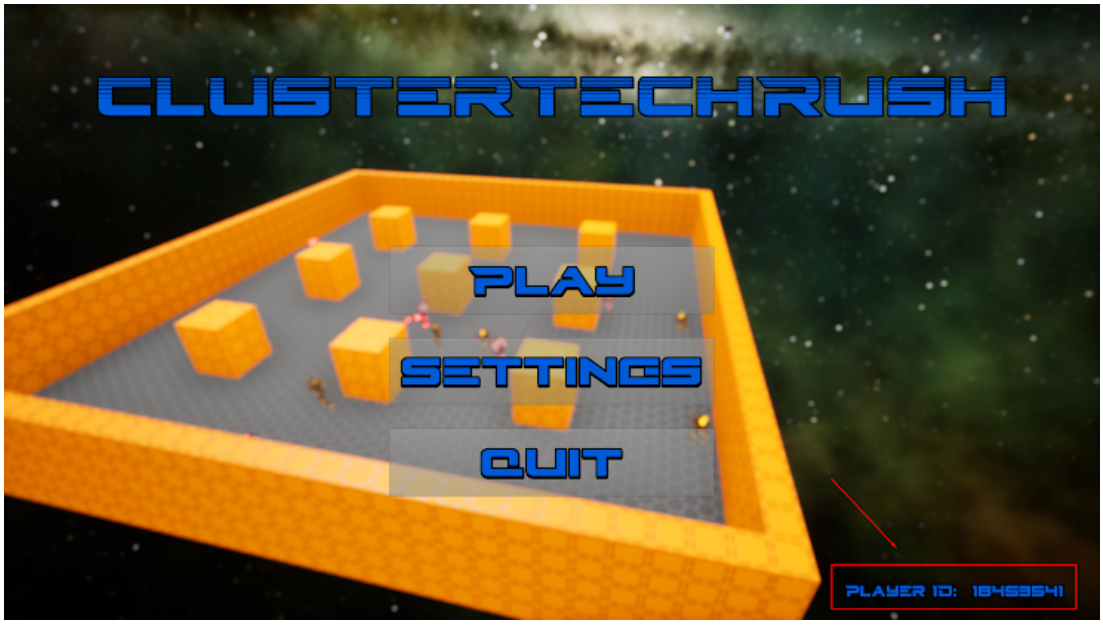
With file downloaded all you need to do is extract it:



To run the game simply double click the .exe file



11. The game will give you an ID, what is your ID? *



Let's get started

Version A

Press Play and Select Version A

In version A, you are free to enter the portal leading to the next level at any time. You can also press 'Q' at any time to teleport back to the previous level. If you die you will restart on the same level, but lose your progress since you entered the level. The resources you will lose if killed are in brackets.
It's up to you to controll your experience, has you go forward the game will get harder, but you can always go back and become stronger before you continue.
Try and finish the game has fast has you can!

Controls:
WASD for movement
Left Mouse click to shoot
Right Mouse Click / Shift to dash
E to interact with portals
Q to go down a level

Select Version A

The guide will let you know all you need about Version A. Read it carefully. Then, once you are ready to start press the tutorial button for a quick introduction to the game.

Reach the End

After playing the tutorial the game will automaticly start in Version A
Once you have reached the end of the game Press Next

Version A Questions

Answer this questions after having played one full run with Version A

12. *

Mark only one oval per row.

	not at all	slightly	moderatly	fairly	extremely
I was interested in the game's story					
I felt successful					
I felt bored					
I found it impressive					
I forgot eveything arround me					
I felt frustrated					
I found it tiresome					
I felt irritable					
I felt skilful					
I felt completely absorbed					
I felt content					
I felt challenged					
I had to put a lot of effort into it					
I felt good					

13. Did you experience any technical problems during your run? if the awnser is yes, describe the problems:

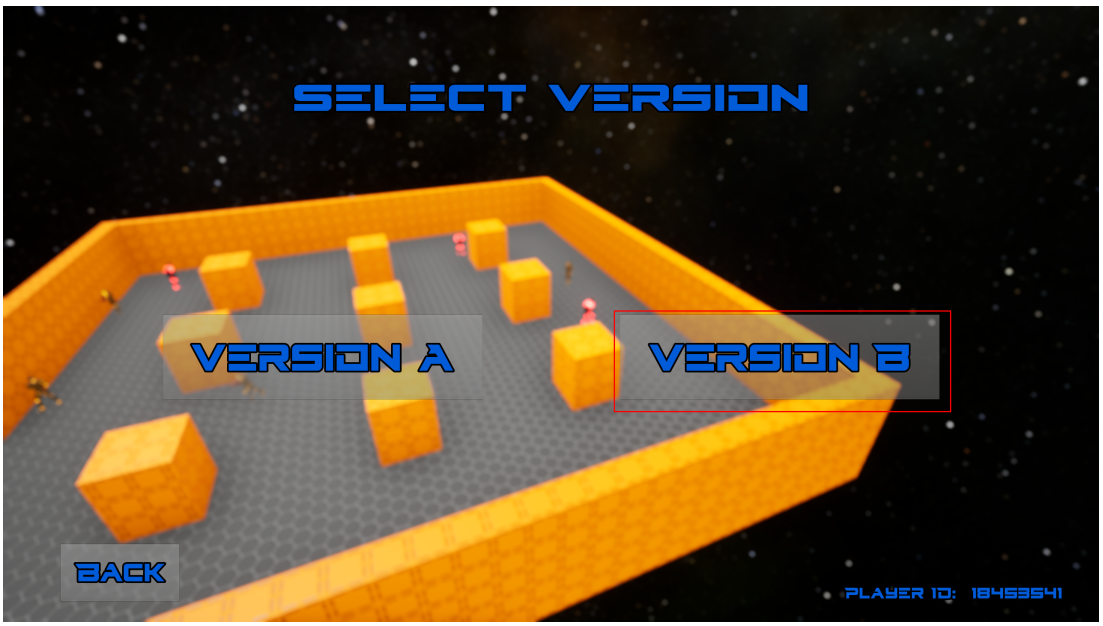
Version B

Press Play and Select Version B

In Version B, you will be automatically teleported to the next level after you defeat a set amount of enemies (by filling the vertical blue progress bar on the right side of the screen) and automatically sent back one level if you your health gets under a threshold and you can't recover before the timer runs out.
Try and finish the game has fast has you can!

Controls:
WASD for movement
Left Mouse click to shoot
Right Mouse Click / Shitft to dashl

Select Version B



The guide will let you know all you need about Version B. Read it carefully. Since you have already completed the tutorial, You can just press Play to skip it.

Reach the End

Once you have reached the end of the game Press Next

Version B Questions

Answer this questions after having played one full run with Version B

14. *

Mark only one oval per row.

	not at all	slightly	moderatly	fairly	extremely
I was interested in the game's story					
I felt successful					
I felt bored					
I found it impressive					
I forgot eveything arround me					
I felt frustrated					
I found it tiresome					
I felt irritable					
I felt skilful					
I felt completely absorbed					
I felt content					
I felt challenged					
I had to put a lot of effort into it					
I felt good					

15. Did you experience any technical problems during your run? if the awnser is yes, describe the problems:

Comparing the two versions

Now that you have played both versions, please compare your overall experience with each version.

16. Overall experience with Version A: *

Mark only one oval.

1	2	3	4	5	6	7
◯	◯	◯	◯	◯	◯	◯

17. Overall experience with Version B: *

Mark only one oval.

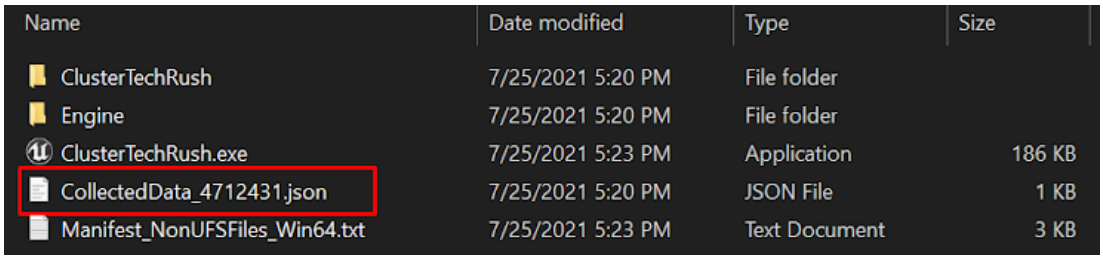
1	2	3	4	5	6	7
◯	◯	◯	◯	◯	◯	◯

18. If you rated the two versions differently, what was the main reason for rating one higher that the other?

Game Data Collection

Finally, we kindly ask you to send us the ".json" file generated by the game with some data about your runs.

19. The file should be in the same directory has the .exe file, and it should look something like this: *



Files submitted:

Thank you for participating!!