Active vs Passive Flow Adjustment in Games*

1st Alexandre Chícharo
Instituto Superior Técnico
Instituto Superior Técnico
Lisbon, Portugal
alexandre.chicharo@tecnico.ulisboa.pt

***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.

***Index Terms*—Flow, Experience, Player Control, Active Flow Adjustment, Passive Flow Adjustment, Dynamic Difficulty Adjustment.**

I. Introduction

A. 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.

B. 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.

Identify applicable funding agency here. If none, delete this.

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?.

C. 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?

II. Related Work

A. Flow

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.

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.

3) 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.

4) 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.

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.

5) 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.

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.

6) 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.

B. Evaluation

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) 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.

III. 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.

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.

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.

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 **??** and **??**. 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 **??**

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 **??**).

A. 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.

B. 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.

leftmargin=*,labelwidth=*,before=

- **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.
- **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.
- **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.
- **Final Boss**

  The final boss a big stationary turret that alternates between holding four shotguns, eight assault rifles or activating a shield and spawning regular enemies. 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.

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.

C. 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, 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.

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

leftmargin=*,labelwidth=*,before=

- **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%.

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%.

D. 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:

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.

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.

E. 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.

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.

F. 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.

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.

G. 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.

IV. Evaluation

A. Preliminary Evaluation

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.

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 **??**) 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.

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

[0]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.

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.

B. Final Evaluation

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?

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 IV-A2.

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. 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.

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: We observed that 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.

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: 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.

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. Most participants felt comfortable 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.

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.

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) 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.

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).

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) 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.

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) 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). 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

[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

differences for the more casual players (Z = 0.011, p = -2.549) 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) 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) 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.

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. 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). 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.

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) 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.

C. 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 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.

V. 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.

REFERENCES

[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