# Evaluation of Game Level Design Using Machine Learning

submitted by

Azeem Ahmed Khan

for the degree of Doctor of Engineering

of the

University of Bath

Department of Computer Science

September 2021

**COPYRIGHT**



Signature of Author ..............................................................

Azeem Ahmed Khan

## Abstract

Game level designers need to know whether the levels they generate will fulfil the desires of their audience. The main goal of the work presented in this thesis is to build a tool which can assist a level designer as they work, providing them with real-time evaluation by predicting how much enjoyment their design will bring about in prospective players. This tool is built via the application of data mining and machine learning algorithms to gameplay, level geometry and feedback data.

Telemetry data from playthroughs of the medieval action game *For Honor* is collected. This data includes player positions and actions, as well as information about the level geometry surrounding the players. The feedback submitted at the end of each match is also gathered. Feature representation is then used to express these playthroughs as a series of moments. We explore two distinct methods of interpreting the influence of these moments on the given rating:

1. Using Weakly Supervised Learning to identify the single most influential moment within each playthrough.

2. Building an ensemble of probabilistic regressors so the influence of all moments are taken into account.

The moments and their corresponding ratings are fed into a neural network which takes geometry as input and outputs some metric representing the predicted rating. The trained model evaluates a game level by visualising the feedback in the form of a "heat map of enjoyment", highlighting the areas of the level that will lead to high or low amounts of enjoyment. The accuracies of these heat maps are assessed by comparing them to coloured maps produced by users who participated in a study.

Our results show that both methods produce heat maps which are in good agreement with the user maps for at least one user. However the outputs of method 2 are less susceptible to noise and, unlike method 1, did not suffer from overfitting during the training process.

# Acknowledgements

This thesis would not have been possible without the help of many people. To my parents, thank you so much for supporting me throughout my education.
I would like to express my gratitude to my academic supervisor Dr Tom Fincham Haines for his patience and clarity when explaining complex concepts, as well as his guidance throughout the entire project. I would also like to thank the University of Bath and the Centre for Digital Entertainment for providing me the opportunity to do this course.
Many thanks are given to my supervisors at Ubisoft Reflections as well as others who oversaw the project: Michael "Mike" Troughton, Michele Condò, Mark Leadbeater, Dario Sancho, Gilles Matouba and Jose Parades. I would especially like to express my gratitude to Mike for granting me the opportunity to undertake research at a company whose games I used to play as a child; his invaluable help in directing me to the relevant people and resources within the company, and encouraging me to participate in Ubisoft's internal events to present my work. I also cannot thank Michele enough for his support and enthusiasm for the project, as well as his influence in starting my career at Ubisoft Reflections.
I want to thank Vincent Gagnon and his team at Ubisoft Montréal for granting us access to their data and materials, without which our research could not have been accomplished.
Special thanks to the people who took the time to participate in our user study, your feedback has been extremely useful in justifying our results.
I also appreciate the assistance by Lee Donaldson from IT, who worked hard to set up an extra machine allowing me to gather results much more quickly.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**2-AFC** 2-Alternative Forced Choice.

**ANN** Artifical Neural Network.

**AV** Arousal-valence.

**BMU** Best Matched User.

**BOW** Bag-of-Words.

**BP** Blood Pressure.

**BSP** Binary Space Partitioning.

**CDE** Centre for Digital Entertainment.

**CDF** Cumulative Distribution Function.

**CET** Cognitive Evaluation Theory.

**CNN** Convolutional Neural Network.

**ECG** Electrocardiogram.

**EDPCG** Experience Driven Procedural Content Generation.

**EMG** Electromyography.

**GA** Genetic Algorithm.

**GSR** Galvanic Skin Response.

**HCI** Human-Computer Interaction.

**HMA** Heat Map Accuracy.

**HR** Heart Rate.

**IoU** Intersection over Union.

**MDA** Mechanics Dynamics Aesthetics.

**MIL** Multiple Instance Learning.

**NEA2** Niching Evolutionary Algorithm 2.

**NNN** Negative Nearest Neighbour.

**NPC** Non-playable Character.

**PCA** Principal Component Analysis.

**PCG** Procedural Content Generation.

**PEM** Player Experience Modelling.

**PENS** Player Experience of Need Satisfaction.

**PRE** Probabilistic Regression Ensemble.

**RTS** Real Time Strategy.

**SDT** Self-determination Theory.

**STIP** Spatio-Temporal-Interest-Point.

**TCTD** Tom Clancy's The Division.

**UPEQ** Ubisoft Perceived Experience Questionnaire.

**VTK** Visualization Toolkit.

**WMU** Worst Matched User.

**WSL** Weakly Supervised Learning.

# Chapter 1

# Introduction

The goal of the work presented in this thesis is to build a tool which can assist a level designer as they work, evaluating their creation in real-time by visualising the amount of predicted enjoyment it will elicit in potential players. This is meant to address the problem of subjectivity in game level design; the differing opinions of designers warrant a more neutral source of evaluation. Therefore we apply machine learning to gameplay, level geometry and feedback data. The resultant trained models are used to predict the feedback of a given level, and the accuracy of this prediction is measured by comparing its output to that of human users.

## 1.1 Background

The video game industry has experienced substantial growth since the release of Pong and Space Invaders in the 1970s. Video games may have experienced their greatest evolution during the 1990s, with the advent of CDs through which software could be stored and distributed, as well as advancements in 3D computer graphics which became the standard for visual representation in games [35]. The 21st century has seen the emergence of independent game development as well as the use of virtual reality and motion capture in video games, bringing them to the same level as films in terms of how they are viewed as a form of entertainment. In fact, games have begun to surpass films as the best-selling media [16]. As games have evolved in terms of graphics and play time, the effort and manpower that goes into their development has constantly increased. It is now more common for a AAA game to be developed by hundreds of people over a period of a year or more, leading to fewer profits [54]. However automation of certain components would substantially shorten development costs — a major motivation for Procedural

Content Generation (PCG). This is defined as "the algorithmic creation of game content with limited or indirect user input" [63]. PCG was used as a solution to the limited space in which content could be stored in early games; a classic example of its use is Rogue, a 1980s dungeon-crawling game in which levels are randomly generated every time a new game starts [54]. PCG can also help level designers be more creative, as algorithms can produce content which is vastly different from that of a human designer. Large game companies such as Ubisoft have benefited from automating content generation, for example the successful launch of Tom Clancy's The Division (TCTD) can be attributed to the creation of a PCG design tool. The key difference to typical PCG technologies is that the play environment was created to satisfy the needs of gameplay, rather than trying to fit gameplay into a procedurally generated world [58].

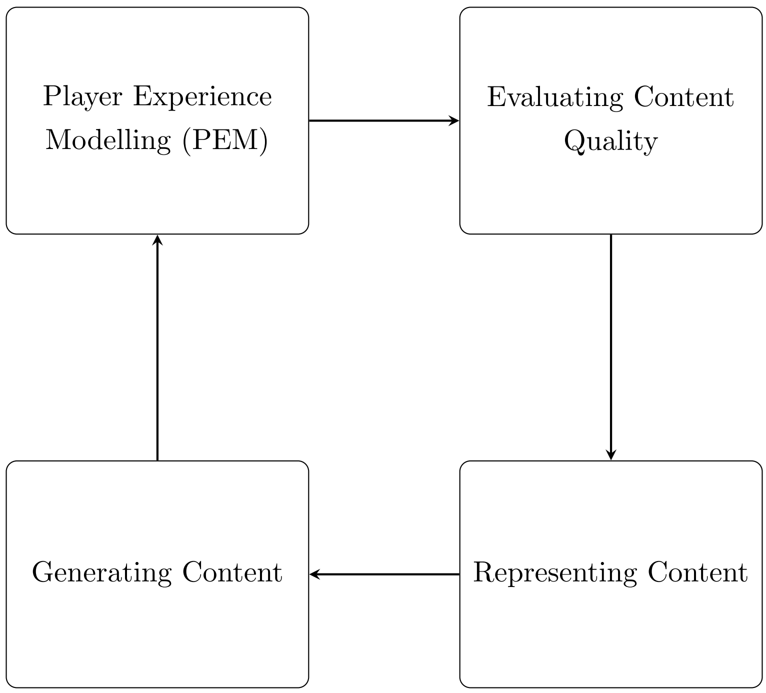



Figure 1.1: The four key components of the EDPCG framework, adapted from [76].

A more exciting prospect of PCG is that the generated content can be tailored to the desires of the user playing the game. This is known as Experience-driven Procedural Content Generation (EDPCG) and it consists of four key components [76] which are illustrated in Figure 1.1. The top two components will be the main areas covered in this thesis.

1. *Player Experience Modelling* (PEM) - this can be performed using three different

kinds of extracted data:

- Subjective — this involves the use of interviews or surveys where the questions can be asked during gameplay (free-response) or afterwards (forced-response).
- Objective — here the physiological responses and bodily expressions of the player are monitored and analysed. One can either use existing emotional models derived from emotion theories (model-based), or construct some unknown mapping between player input and emotional state (model-free).
- Gameplay-based — an analysis of the player's in-game actions, either via a general framework of behavioural analysis (model-based) or identifying patterns in the data to predict player intentions (model-free).

2. *Evaluating Content Quality* — an evaluation function is needed to assess a level and assign a value reflecting its quality or suitably for use in a game. This function is created depending on what a designer wishes to optimise, and falls into one of three classes [64]:

   - Direct evaluation functions — these directly map extracted features (e.g. number of entry points, firing rate) to a quality value. They can be either data-driven (collecting data and using algorithms to map from content to player experience and then to an evaluation function), or theory-driven (where the designer relies on intuition or some qualitative theory to perform this mapping).
   - Simulation-based evaluation functions — this involves allowing an artificial agent to play through the level being evaluated, then extracting features from the agent's performance. These functions can be static (the agent is assumed to not change during the game) or dynamic (the agent changes and the quality value incorporates this change).
   - Interactive evaluation functions — in this, the quality value is based on the player-game interaction. Here the quality is evaluated during actual gameplay, and data can be gathered either explicitly (questionnaires or verbal cues) or implicitly (monitoring eye-gaze fixation, facial expressions, time the player quit e.t.c.).

3. *Representing Content* — the content of a game needs to be represented in a way which improves the efficiency, performance and robustness of a generator. This could be symbolically within a tree or graph data structure. Different representations can be distinguished by how directly or indirectly they are encoded. As an example

one can consider a level in a 2D platform game - this can be directly represented as a 2D grid where the contents of each cell is specified. A more indirect representation could be a list of positions, shapes, enemies and items. In the case of TCTD, content representation was in the form of a high level script that outlined key aspects of a mission template.

4. *Generating Content* — the generator needs to search within a resulting search space for content that maximises particular aspects of a player's experience. The more direct the representation, the larger the search space. For TCTD multiple different underground dungeons were generated that satisfied the aforementioned mission template.

EDPCG could be instrumental in closing the affective loop in games — a human-computer interaction (HCI) goal in which an artificial system understands and reacts to a user's emotional state [60].

## 1.2 Research Problem Overview

It was discussed in the previous section that PEM and content quality evaluation are key parts of the EDPCG framework. The link between these two parts is especially important to level designers; in order for them to have a measure of how enjoyable their levels are and produce successful games, they must be able to know what players enjoy. A game level consists of various elements/artifacts which can be arranged in a number of different ways — this is the game content [51]. However the level is meaningless unless someone actually plays it and has an interactive experience within it. This is ultimately what the level designer cares about when working — the player experience. Unfortunately this presents another challenge — experiences cannot be directly measured, only described. Many level designers will use introspection i.e. use their own experiences as a basis for creating levels; others may decide that only playtesting i.e. the experiences of others can be trusted [53]. Each of these methods has its own flaws: the former only relies on one person's opinions, and some level designers may have unpopular tastes. The latter relies on a process that occurs infrequently during game development, in addition to the fact that there must already be a working game to playtest. In many cases, a designer may be given certain guidelines or criteria to fulfil when tasked with creating levels e.g. create a race track which lasts at least $X$ seconds per lap [39]. However even these can be vague. This is a fundamental problem that exists within level design — it is a subjective discipline because different people will have different views on whether a level is good or bad [1]. The point at which a level is played in relation to others e.g. in an open world game where side missions can be played in almost any order, can also affect a given player's opinion. Additionally a player may not be able to effectively dissect their experience and articulate why they found a particular level fun, therefore providing the level designer with insufficient information about how they could improve the level. Even the concept of "fun" has multiple meanings, making it difficult to fully capture what is happening in terms of keeping the player motivated [51]. For example game designer Marc LeBlanc defined eight types of fun: sense-pleasure, make-believe, drama, obstacle, discovery, self-discovery and expression, social framework and surrender [26]. When a player gives their feedback after experiencing a level, it is based on their personal conception of fun. This feedback potentially links to the structural design of the level, and machine learning could be the key to extracting this relationship.

## 1.3 Aims & Objectives

The goal of this project is to create a tool that can provide real-time feedback to a level designer as they work, giving them information as to which areas of the proposed level will result in the most enjoyable experiences for potential players. The system will learn the relationship between level structure and player enjoyment, via the application of machine learning and data mining to existing level playthroughs as training data. This information will also be used to attempt to predict the feedback of playthroughs. The main objectives of the project are as follows:

- Collect data from play sessions and maps of a particular game, most importantly level geometry and player feedback.
- Simplify and convert the data into a suitable representation.
- Identify key moments which most strongly influenced player feedback.
- Train a model which can predict feedback based on level geometry.
- Visualise the predicted feedback in a form helpful to designers.

[1]This description of level design was stated during a presentation by Daniel Molnar, a level design manager at Ubisoft Reflections on 06/03/2018.

## 1.4 Industrial Context

The work presented in this thesis is the culmination of a collaboration between Ubisoft Reflections and the Centre for Digital Entertainment (CDE).
Based at the University of Bath and Bournemouth University, the CDE is a doctoral training centre which funds research students in fields such as games, animation and visual effects [10]. These students are usually based at companies requiring these skills on an industrial placement, using concepts they have studied in academia to solve problems within industry.
Ubisoft Reflections is a game development studio based in Newcastle Upon Tyne in the United Kingdom. It is part of Ubisoft, one of the largest video game publishers in the world. The studio has collaborated with other Ubisoft studios around the world to create many AAA games such as Far Cry 5 and Assassin's Creed Syndicate. It is also responsible for the creation of smaller games such as Grow Up and Atomega [65].

## 1.5 Research Outputs

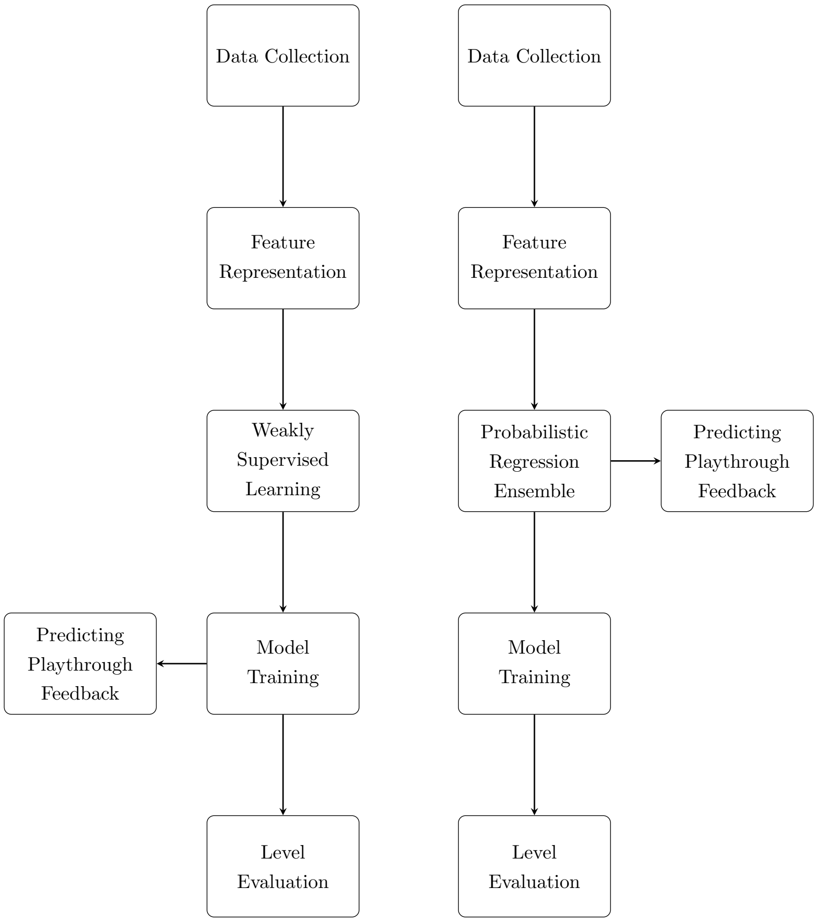
The following is a list of research outputs of the work presented in this thesis.

| Event | Title | Format |
|---|---|---|
| CVMP 2018 | Evaluating the "fun" Factor of Levels | Poster |
| CVMP 2019 | Data-driven Game Content Evaluation | Poster |
| CVMP 2020 | Evaluating the Content Quality of Game Levels | Poster |
| LaForge Open House 2021 | Evaluating the Content Quality of Levels | Poster & Presentation |
| UDS 2021 | Data-driven Content Evaluation for Level Designers | Presentation |

## 1.6 Document Roadmap

So far this thesis has introduced game content evaluation within the context of EDPCG, and outlined a data-driven method for achieving this. Chapter 2 is a comprehensive literature review concerning the existing research in the fields of PEM and level evaluation, as well as other key concepts involved in the project.
Chapter 3 describes the game which was used for the project and the specific types of data which were collected from it, for both training the system and assessing its

performance.
Chapter 4 goes into detail about the implementation of the algorithms used to complete each objective in the project's pipeline. It begins with feature representation before branching off into two separate methods for selecting the most important features (moment detection), and showing how these are used to train models for PEM and generating visualisations of the model predictions. This also includes the visual outputs of each step. Figure 1.2 illustrates a way in which the two distinct moment detection methods produce two separate pipelines within the project.
Chapter 5 concerns the final results produced by both of the presented methods, specifically an analysis of the system's performance associated with both level evaluation and predicting playthrough feedback.
Chapter 6 concludes the thesis by first summarising what has been achieved, before describing the limitations of the project; how these may have affected the performance, and how these may be overcome. It then proposes potential extensions to the project, and how they may be pursued. Finally the impact and implications that this project has on the field of level design and the game industry as a whole is discussed.




(a) Weakly Supervised Learning. (b) Probabilistic Regression Ensemble.

Figure 1.2: Pipelines for the two methodologies used in the project.

# Chapter 2

# Literature Review

It has been stated that the two components of experience-driven procedural content generation (EDPCG) which will be discussed in this thesis are player experience modelling (PEM) and evaluating content quality. This is because in order to evaluate levels on their enjoyment potential, one must have a way of modelling how the features of those levels translate into player enjoyment. This chapter begins by reviewing the various approaches to PEM over the last several decades. We then discuss the features of an existing level evaluation tool - the Sentient Sketchbook. Finally since one of our approaches uses the machine learning technique known as Weakly Supervised Learning, this concept is introduced and explained using examples of studies in which it has been applied to images and videos.

## 2.1 Modelling Player Experience

There has been much research into modelling or evaluating the entertainment value of games i.e. identifying what aspects of a game engage the people who play them, and cause them to enjoy the experience. Many of these are rooted in psychological and HCI studies.

### 2.1.1 Qualitative Approaches

Thomas Malone carried out a series of studies to investigate the features that make games so captivating, and how these features can be used to create environments in which students are motivated to learn as efficiently and effectively as possible. These studies involved surveying the computer game preferences of elementary students, and testing multiple versions of certain games, differing in their focus on specific features.

According to Malone, the characteristics of intrinsically motivating environments fall under one of three categories [36]:

- *Challenge* — an environment is challenging when it has multiple goals, with uncertain outcomes to keep the player engaged and motivated. Also an environment should have a variable difficulty level so the learner can work at a level appropriate to their ability this also ensures that their self-esteem is not lowered to the point where they are disinterested in the game.

- *Curiosity* — environments should be novel and surprising, but not completely incomprehensible. This can be divided into two types: *sensory curiosity* and *cognitive curiosity*. The former refers to how changes in light, sound or sensory stimuli of an environment attract the attention of the user. The latter refers to presenting the user with just enough information to make their existing knowledge seem incomplete, thus engaging their curiosity and encouraging them to learn more.

- *Fantasy* — this refers to mental images created by the player as a result of their interaction with the game environment. Fantasies can be both intrinsic and extrinsic, with the intrinsic being more interesting and instructional. A cognitive advantage of intrinsic fantasies is their ability to improve the memorability of the material by provoking vivid images related to it.

The above principles were applied to existing educational tools, turning them into learning games in order to improve their quality and effectiveness [36]. The *challenge* and *curiosity* categories were also used as the basis for Yannakakis and Hallam's experiments to derive quantitative models of entertainment, for both computer games [70] and physical games [71]. These are discussed in further detail in Section 2.1.2.

In 2004 Nicole Lazzaro performed a field study in which adults were asked to share their thoughts and feelings while playing their favourite game. This was in order to know more about the role of emotion in games, and mechanisms other than cut scenes which evoke these emotions [29]. To obtain the opinions of non-gamers, the friends and family of the participants were also interviewed. Three types of data were collected: video recordings of players during play sessions, questionnaire responses and emotional cues (such as facial expressions) during gameplay. Lazzaro concluded that what players like about games falls into four "keys of emotion" [29]:

- *Player* (Internal experience key) — how the game makes them feel inside, as well as changes in their internal state during and after play. This focuses on how the game aspects create emotions inside the player. The kinds of players which fall under this category said they like clearing their mind by completing the level, avoiding boredom and feeling better about themselves.

- *Hard fun* (Challenge and strategy key) — people play games to overcome obstacles. The game creates emotion by structuring the experience towards the pursuit of some goal. Players who fell more into this key said they like playing to assess how good they are; playing to beat the game; having multiple objectives and winning through strategy rather than luck.

- *Easy fun* (Immersion key) — this key focuses on the enjoyment of experiencing the game. Focus is maintained with the player's attention rather than a winning condition. It entices the player to consider their options and investigate more. Players falling into this category said they enjoy exploring new worlds, excitement and adventure, and seeing what happens in the story.

- *Other players* (Social experience key) — this refers to enjoyment from playing with others inside or outside the game. Some participants admitted that they might play games they don't like in order to spend time with friends. Players to which this key applies see games as mechanisms for social interaction — they say it is the people who are addictive, not the game. They also want an excuse to invite friends over. Even though they don't play games, they enjoy watching others play.

Arriving at these four factors involved observing emotions produced during gameplay via facial gestures, body language and verbal comments [29]. Lazzaro's "fun clustering" was the inspiration for Raph Koster's personal breakdown of player enjoyment [26]. Koster suggested that games are primarily a learning experience; the enjoyment experienced during a game is a result of the brain being taught new and interesting patterns, therefore games focus mostly on *hard fun*. Our approach investigates the link between enjoyment and the geometry of a level, and hence leans towards the *easy fun* factor more so than it does towards the others.

The MDA framework defined by Hunicke et al. divides games into three separate components [21]:

- *Mechanics* — this describes the game in terms of data representation and algorithms, and may include things such as the basic rules of the game and the information that goes into constructing it.

- *Dynamics* — this describes the way the game actually plays based on the mechanics i.e. the events that occur within the game as experienced by the player.

- *Aesthetics* — this describes the desirable emotional responses in the player as they interact with the game. Within aesthetics, elements that make the game attractive include sensation (when the player experiences something unfamiliar); fantasy (getting caught up in an imaginary world); narrative (an engaging story); challenge (the need to master something); fellowship (forming and actively taking part in a community); discovery (the player's need to explore); expression (playing to their creativity or leaving their mark) and submission (referring to the game as a pastime) [1]. Games possess multiple of these aesthetic elements to various degrees.

The MDA model is very useful for understanding the way games work — it is simplistic but offers a sufficient distinction between various elements of a game, and highlights the ways in which games are systems rather than linear pre-determined structures like books or films. There are limitations to this framework — it does not take into account things like the context in which one plays the game or the culture which frames the game [13]. However from a design perspective, the aesthetic elements have large overlap with procedural content generation (PCG) — some have been explored with PCG while others may provide the basis for new PCG systems [57].

There are many models which have been developed to explain and analyse media enjoyment. Disposition theory relates attitudes toward media characters to moral evaluations of their actions i.e. enjoyment increases when liked characters are successful and disliked characters encounter misfortunes [48]. Transportation theory suggests that enjoyment is heightened by immersion in a narrative world, as well as the consequences of this immersion [18]. Parasocial interactions refer to the relationship that an audience member develops with a character by talking to them, imagining or discussing their life. Then there is cognition, where viewers make judgments on a character's attributes. Examples of these could be their ethics, interest and intelligence [42]. However these models are individually fairly narrow as they understand enjoyment in terms of one concept. This is where *flow theory* [11] is advantageous — it is based on the premise that elements of enjoyment are universal, and can be summarised in a general model. The concept of flow originates from research conducted by Csikszentmihalyi into what makes experiences enjoyable. This was based on interviews, questionnaires and other data collected over the course of twelve years with several thousand participants. According to Csikszentmihalyi, flow consists of eight elements, the combination of which causes a sense of deep enjoyment so rewarding that people feel it is worth expending a great deal of energy to achieve it. Sweetser and Wyeth took this concept and adapted it to computer games, creating the concept of GameFlow [61] which contains the following elements:

- *Concentration* — games should provide stimuli from different sources, grab the player's attention and maintain their focus.
- *Challenge* — games should provide challenges which match the player's skill level, as well as increasing in difficulty to improve the player's skill level at an appropriate pace.
- *Player Skills* — players should be able to play the game without reading the manual, instead being taught through tutorials, as well has having access to online help. Also the interfaces and mechanics should be easy to use.
- *Control* — players should feel a sense of control over their actions within the game, as well as feeling that their actions are having a significant impact in the game.
- *Clear goals* — games should provide the player with clear goals at appropriate times.
- *Feedback* — players should receive feedback on their progress, actions and score.
- *Immersion* — players should become less aware of their real-world surroundings, as well as becoming emotionally invested in the game and experiencing an altered sense of time.
- *Social Interaction* — games should support cooperation between players and social communities inside and outside the game.

In order to validate the GameFlow elements and their criteria, two fantasy games (Warcraft 3 and Everquest) were evaluated and compared using expert review. This was also performed to identify any potential weaknesses or ambiguities in the model. The games were given a score for each criterion, and these were averaged for each element. The results were consistent with professional ratings of the games, however it was acknowledged that some criteria were not applicable to them, or were difficult to measure without further evaluation e.g playtesting. Given that only two games were evaluated in this study, the versatility of the model is still unclear. The authors also concluded that the GameFlow criteria could be used as guidelines for expert reviews or playtesting, but are not suitable for use by game developers as an evaluation tool [61].

Another model for predicting fun/enjoyment is the Player Experience of Need Satisfaction (PENS) model [50]. Developed by Scott Rigby and Richard Ryan, its roots lie in more than thirty years of research into human motivation and psychological health, and seven years of research on games [51]. More specifically it was elaborated from Self-determination Theory (SDT) — a theory of motivation that concerns intrinsic and extrinsic motives for acting, as well as the relationship between motivation and growth/well-being [52]. A mini-theory of this which is only concerned with intrinsic motivation, known as Cognitive Evaluation Theory (CET), was applied to video games. Therefore the model is based on the idea that games satisfy specific psychological needs that exist in potential players; these satisfactions provide the games' pull. According to PENS, video games are most successful, engaging and fun when they satisfy the following intrinsic needs:

- *Competence* — this refers to the innate desire to grow abilities and gain mastery of new situations and challenges.
- *Autonomy* — this reflects one's inherent desire to take action out of personal volition and not because one is "controlled" by circumstances.
- *Relatedness* — this refers to the need to have a meaningful connection to others.

In order to investigate the relations between these needs and game characteristics, several studies were carried out in which participants were asked to play a game (or multiple games) and then answer a questionnaire which used a uniform 7-point Lickert-type scale [52]. The questions themselves were designed to assess how much the players felt that the above intrinsic needs were satisfied, for example "I felt very capable and effective" or "I felt controlled and pressured to be a certain way". These items were averaged to produce an in-game score corresponding to each intrinsic need. The scores associated with game variables such as *presence* (a measure of immersion) and *game enjoyment* were regressed on to in-game autonomy and competence simultaneously. The results showed that some variables were significantly associated with both autonomy and competence, whereas others only related to one or the other. For relatedness, similar studies were performed using multiplayer games [52].

It is interesting to note the large amount of overlap between the concepts presented in Lazzaro's fun factors, GameFlow and PENS — this is summarised in Table 2.1. For example there is always an element allowing players to improve their skills — this corresponds to *hard fun*, *challenge* and *competence* in these respective models. Also the GameFlow concept of *control* is extremely similar to *autonomy* in PENS as they both emphasise the idea of the player choosing their own actions that they believe will impact the game significantly. This may also be loosely connected to the *player* factor in Lazzaro's model, as the player may feel satisfied knowing it was their choices that led to successful completion of the game. The social aspect of games i.e. interacting with other players cooperatively or competitively is also consistently present throughout all of these models. Finally the GameFlow concept of *immersion* finds its analogues in Lazzaro's *easy fun* and the *presence* variable in PENS.

| Lazzaro's factors | GameFlow | PENS |
|---|---|---|
| Player | Control | Autonomy |
| Hard fun | Challenge | Competence |
| Easy fun | Immersion | Presence |
| Other players | Social interaction | Relatedness |

Table 2.1: Analogue concepts in three different qualitative models of player experience.

SDT was used as the basis for a model of motivation defined by Melhart et al. [41] in cooperation with Ubisoft Massive. Gameplay data was collected from more than 400 players of Tom Clancy's The Division, along with their reported levels of competence, autonomy, relatedness and presence using the Ubisoft Perceived Experience Questionnaire (UPEQ). Four different player types were produced via k-means clustering. Preference learning was then used to derive a mapping between these player types and UPEQ responses, then between gameplay and UPEQ responses. Models derived from the latter mapping proved to be more accurate and robust than those of the former. Our approach also utilises k-means clustering, and Melhart's use of it to derive specific player types could potentially improve our system — this is discussed in further detail in Section 6.2.

Guckelsberger et al. [19] carried out an exploratory study based on empowerment — a quantity which measures an agent's influence on its environment, as well as its ability to perceive this influence afterwards. They proposed that empowerment was also linked to the concepts of autonomy and relatedness in CET. Their study was motivated by the challenge of evoking specific player experiences in levels created via PCG. The authors aimed to automatically predict player experience using computational models of intrinsic motivation, without including a human in the loop. Levels of an infinite runner game RoboRunner were procedurally generated and their predicted experiences were computed using a simulation-based approach. Then several human participants were asked to play these levels, think aloud during gameplay and answer questions afterwards. Their commentary and responses were analysed and used to identify the following themes which are also found in existing PEM theories: challenge, involvement, learning, emotion, attention and engagement.

Outside of collaboration between academia and industry, the qualitative concepts mentioned in this section have been used by video game companies in their guidelines for level design. For example Ubisoft states that designers build fun experiences by perfecting the following three pillars [4]:

- *Guidance* — teaching how the game plays and ensuring the player understands their objectives.
- *Challenge* — constantly testing the player's skills.
- *Immersion* — drawing the player into the game.

Designers may adapt these concepts in a way which aligns most with the genre of the game on which they are working — in the case of race tracks, the designer may focus on ensuring new players experience speed, momentum and drama [39].

### 2.1.2 Quantitative Approaches

The first attempt at a quantitative study of fun was the work of Iida et al. [22] concerning entertainment metrics of boardgames. They proposed the following estimate of measure of entertainment $E$ for a given game $G$:

$$E(G) = \frac{D}{b}, \tag{2.1}$$

where $D$ is the length of the game and $b$ is the average number of plausible moves for the player. This estimate assumes that a player would make their decision with probability $\frac{1}{b}$ at each position. They also assume that there is a direct relation between $b$ and the player's strength $s$:

$$b = B^{\frac{1}{s}}, \tag{2.2}$$

where $B$ is the average number of possible moves. An omniscient player would select the most optimal moves at any position, whereas a novice would usually be unable to distinguish between good and bad moves i.e. all possible moves are plausible moves. Therefore equation (2.1) can be re-written as

$$E(G) = DB^{-\frac{1}{s}}. \tag{2.3}$$

This formula was used to investigate the evolution of chess variants. However since no grandmaster games for old chess variants were available, self-play experiments were introduced. Each game was played between two identical copies of a computer program. For each experiment, 1000–2000 games were played to gather data on $B$ and average $D$. It was concluded that the evolutionary change of rules in chess variants took at least two paths — increase in search-space complexity and increase in entertaining impact. Modern chess is the result of natural selection while being well-balanced in both of these cases [22].

Iida's measure is disadvantageous in that it uses concepts which have no equivalent in modern computer games. Lankveld et al. introduced the concept of *incongruity* from psychological literature as a measure of entertainment value in computer games [27]. Incongruity is defined as the difference between game complexity (difficulty of the game environment) and mental complexity (a reflection of the player's understanding of the game environment). Lankveld et al. assumed that interest is maximised when a game is well-balanced — by adapting the game complexity to the mental complexity of the player so that incongruity is constant, the player should continually experience interest. This was inspired by the concept of flow [11] mentioned in Section 2.1.1. They developed a side-scrolling arcade game in which enemies can deal damage to the player, but defeating these enemies will allow the player to gain health. During gameplay the amount of damage dealt to the player by each enemy was measured — this was taken to be a measure of the complexity of each enemy. The environmental complexity was defined as the total of these individual complexities. The mental complexity was measured by keeping a score of the player's progress and their sustained damage. Incongruity was said to be at a balanced level if the player maintains no more than just enough health to be able to complete the game. Lankveld et al. assumed that with this balanced setting, the player is encouraged to learn and increase their mental complexity to meet the demands of the game's complexity. However no experiments were performed to validate the study's hypotheses [27].

Yannakakis and Hallam [68, 69, 72] carried out studies in which they measured the entertainment of predator-prey games e.g. Pac-Man. They based their studies on the hypothesis that entertainment is mainly dependent on player-opponent interaction, rather than audiovisual features and game narrative or genre. Their main goal was to derive a quantifiable metric for entertainment by first quantifying the criteria that define interest in any predator-prey game, before combining these into a single formula. The criteria are:

1. An appropriate level of challenge i.e. when the game is neither too easy nor too hard.

2. Behaviour diversity i.e. when the NPCs are able to hunt and kill the player in different ways.

3. Spatial diversity i.e. when predators move constantly all over the game world and cover it uniformly. This gives the player the impression of an intelligent opponent.

To quantify these criteria, Yannakakis and Hallam let the examined group of opponents (the Ghosts in the case of Pac-Man) play the game $N$ times. Each game was played for a sufficiently large evaluation period of $t_{max}$ simulation steps. They recorded the number of steps $t_k$ taken to kill the player, as well as the total number of opponents' visits $v_{ik}$ at each cell $i$ of the game field grid for each game $k$. For continuous game motion, the grid was created by discretising the game field up to the character's size. The metric for the first criterion is as follows:

$$T = \left(1 - \frac{E\{t_k\}}{max\{t_k\}}\right)^{p_1}, \tag{2.4}$$

where $E\{t_k\}$ is the average number of simulation steps taken to kill the player over the $N$ games; $max\{t_k\}$ is the maximum $t_k$ over the $N$ games and $p_1$ is a weighting parameter. $p_1$ is selected to be less than 1 in order to give a boost to $T$ when there is even a slight difference between $E\{t_k\}$ and $max\{t_k\}$.
The quantification of the second criterion is given by $S$, which promotes predators that produce high diversity in the time taken to kill the player:

$$S = \left(\frac{\sigma}{\sigma_{max}}\right)^{p_2}, \tag{2.5}$$

where

$$\sigma_{max} = \frac{1}{2}\sqrt{\frac{N}{N-1}}(t_{max} - t_{min});$$

$\sigma$ is the standard deviation of $t_k$ over the $N$ games; $t_{min}$ is the minimum number of steps required to kill the player and $p_2$ is a weighting parameter which is set so that $\sigma$ has a linear effect on $S$.

The third criterion is quantified through the entropy $H_n$ of the predator's cell visits in a game $n$ - this quantifies the completeness and uniformity with which the opponents cover the environment. Therefore for each game, the entropy is calculated and rescaled by the following:

$$H_n = \left[-\frac{1}{\log V_n}\sum_i \frac{v_{in}}{V_n}\log\left(\frac{v_{in}}{V_n}\right)\right]^{p_3}, \tag{2.6}$$

where $V_n = \sum_i v_{in}$ is the total number of visits of all visited cells and $p_3$ is a weighting parameter which is set greater than 1 to promote high $H_n$ values.

These three metrics can be combined to form one metric $I$ which is the interest value of a predator/prey game:

$$I = \frac{\gamma T + \delta S + \epsilon E\{H_n\}}{\gamma + \delta + \epsilon}, \tag{2.7}$$

where $E\{H_n\}$ is the average value of $H_n$ over the $N$ games and $\gamma$, $\delta$ and $\epsilon$ are criterion weight parameters [68, 69, 72]. These parameters are manually selected based on the specific game, and adjusted so that $I$ increases as the opponent behaviour changes from random to near-optimal.

Experiments were performed in order to optimise entertainment in predator/prey games. The Ghosts in a Pac-Man game were trained to learn and adapt to new playing strategies. This involved the use of a neuro-evolution offline learning mechanism to produce emergent behaviour in the Ghosts as they played against three different Pac-Man types. These Pac-Man players were computer-controlled and the player types/strategies were:

- Cost-based (CB) — the Pac-Man moves along a cost minimisation path but only in the local neighbour cell area.
- Rule-based (RB) — this is the same as CB but with an additional rule to improve pellet-eating behaviour.
- Advanced (ADV) — this generates a more global Ghost-avoidance behaviour built upon the RB Pac-Man's pellet-eating strategy.

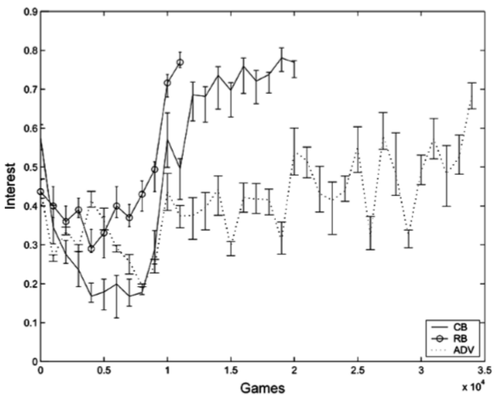
The generated Ghost behaviours from these simulations were:

- Blocking (B) — these Ghosts tend to wait for Pac-Man to enter a specific area that is easy for them to block and then kill.

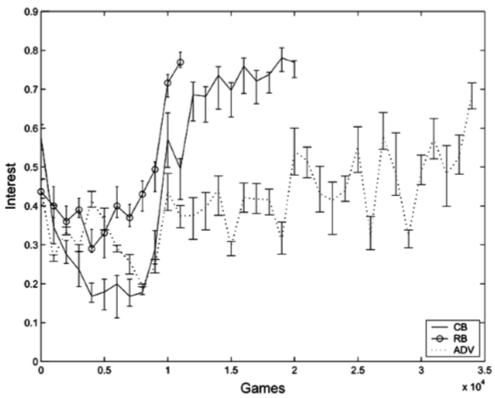
- Aggressive (A) — this kind of behaviour involves following Pac-Man all over the stage in order to kill it.

- Hybrid (H) — these Ghosts are referred to as such because they tend to behave as a hybrid between B and A.

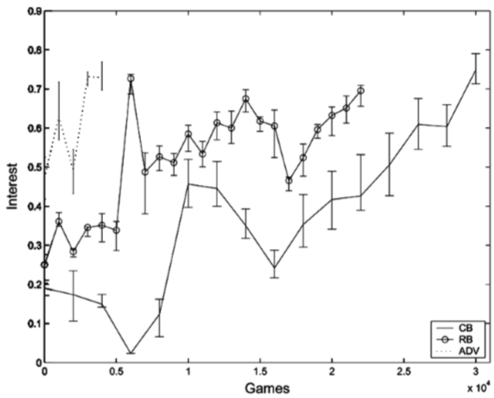
An online learning procedure was then applied in order to evolve the interest of these games, the results of which can be found in Figure 2.1. The results show that the learning mechanisms were able to produce games of higher interest than their original versions, as well as maintaining this high level of interest for a long period of time.




(a) Blocking.




(b) Aggressive.




(c) Hybrid.

Figure 2.1: Game interest according to equation (2.7) over the number of online learning games for each of the ghost behaviours. Taken from [72].

In order to verify whether the interest value derived in equation (2.7) is consistent with actual interest derived from human judgement, a survey was conducted in which human subjects were used as Pac-Man players. Five opponents whose interest values varied uniformly across the $[0, 1]$ space were selected and each subject played sets of games (five game per set) against three of these selected opponents. Each time a pair of sets was completed, the player was asked which of the two sets was more interesting. This approach, known as 2-alternative forced choice (2-AFC), is advantageous because it minimises assumptions about people's different notions of entertainment [74, 71]. Measuring the agreement between equation (2.7) and the human judgement of interest involved calculating the Kendall rank correlation coefficients [72]

$$c(\vec{z}) = \sum_{i=1}^{N} \frac{z_i}{N}, \tag{2.8}$$

where $N$ is the number of incidents to correlate and

$$\vec{z} = \begin{cases} 1, \text{ if subject agrees with (2.7);} \\ -1, \text{ if subject disagrees with (2.7).} \end{cases}$$

The binomial distribution was used for obtaining the correlation coefficient probabilities (also known as p-values $P(C \geq c)$). For reference, the observed effect is "highly significant" if $P(C \geq c) < 1\%$; "significant" if $1\% < P(C \geq c) \leq 5\%$. The total agreement coefficient ($c = 0.3888$) and its p-value ($P(C \geq c) = 1.31 \times 10^{-7}$) showed that a human player's notion of interest correlates strongly with that of equation (2.7). However there were some mismatches which indicated that the behaviour of a human player is different to that of a computer-controlled player. Also it was demonstrated that subjects which disagreed with equation (2.7) judge interest by their score or other personal criteria such as game control and graphics [72].

A similar series of experiments [44, 45, 46] to the one discussed above were carried out on a platformer, in order to build a quantitative model of player experience for this genre of games. The test-bed platformer was Infinite Mario Bros, a version of the classic Super Mario Bros with the additional feature of automatically generated random levels. The purpose of building this model was to form the basis of experience-driven PCG in platformers.

The work of Yannakakis et al. [68, 69, 72, 74, 71] is the quantification of Malone's concepts [36] discussed in Section 2.1.1. Specifically their work on the Playware Playground [74, 71] was an attempt to introduce quantitative measures for the challenge and curiosity entertainment factors. This consisted of a set of $6 \times 6$ tiles on which twenty eight children were asked to play a Bug-Smasher game (see Figure 2.2) — the goal is to step on as many lighted tiles as possible. In order to increase the fantasy factor, different sounds and colours represented different bugs on appearance and smashing. Each subject played two out of eight selected game states in all permutations of pairs, each game differing in one or more of the levels of entertainment factors, before being asked which of the two was more interesting (the 2-AFC approach). The answers to these questions were used to guide the training of an artificial neural network (ANN) model of entertainment. The solutions emerging from this approach successfully mapped correlations between entertainment, challenge and curiosity, and these correlations appeared to follow the principles of Malone's qualitative studies. Also regarding player response time, the results showed that fast responding children show a preference for low challenge and low curiosity games; slow responding children preferred games of high challenge and low curiosity [75].

Figure 2.2: A child playing the bug-smasher game, taken from [73].

Another study was carried out involving the Playware game platform, however this time the subject's heart rate (HR) was recorded as well as their judgement of entertainment. This was motivated by lack of research into the effect of entertainment on a player's physiological state [71, 73]. The HR data was gathered via a wireless Electrocardiogram (ECG) device placed on the child's chest. In order to identify the features of the HR dynamics that correlate with entertainment, several statistical parameters were computed, including average HR, HR signal variance, maximum and minimum HR and the difference between them. Also three different regression models (linear, quadratic and exponential) were used to fit the HR signal. The obtained results indicated that average HR was the only feature strongly correlated with player satisfaction [73]. This study is an example of a model-free approach to PEM, where the type of data gathered is objective (see Section 1.1).

Physiological approaches to modelling a user's emotional state and experience were carried out by Mandryk et al. in [37] and [38], respectively. In [37], several participants were asked to play a sports game (NHL 2003) under three conditions: against a computer, co-located friend and co-located stranger. The following physiological signals were measured:

- Galvanic Skin Response (GSR) — this is a measure of the conductance [1] of the skin; it is affected by specific sweat glands located in the palms of the hands and soles of the feet, which respond to psychological stimulation. This was measured using surface electrodes sewn in Velcro straps placed around two fingers on the same hand.

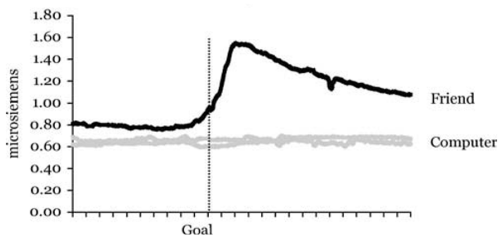
- Cardiovascular activity — HR, interbeat interval, HR variability, blood pressure (BP) and BP variability were all measured using an ECG. Three pre-gelled surface electrodes were placed on the subject's body — two on the chest and one on the abdomen.

- Electromyography (EMG) — this measures muscle activity by detecting surface voltages that occur during muscle contraction. EMG from smiling ($\text{EMG}_{\text{smiling}}$) and frowning ($\text{EMG}_{\text{frowning}}$) activity were collected via surface electrodes placed on the specific muscles associated with these activities.

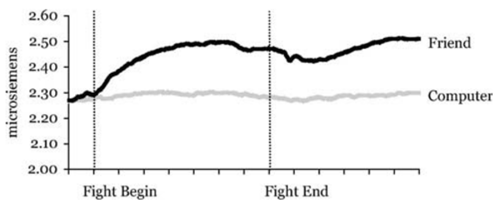
These signals were normalised and used as inputs for a fuzzy logic model which was used to transform them into arousal-valence (AV) space. A second fuzzy model was then used to convert AV into five emotions: fun, challenge, boredom, frustration and excitement. The results indicated that high HR and GSR leads to high levels of AV, which in turn correspond to fun and excitement [37].

[1] Usually measured in microsiemens ($\mu$S).

In order to investigate whether physiological measures can be used to objectively measure a player's experience with entertainment technology, a similar set of experiments was carried out in which respiration rate of the participants was measured in addition to the four signals mentioned earlier [38]. In the first experiment, subjects played a sports game in four conditions of difficulty: beginner, easy, medium and difficult. In the second, the conditions were against a co-located player and against a computer. As well as physiological data, subjective data was also collected in the form of pre and post-experiment questionnaires — after playing each condition, participants were asked to rate the experience on a five-point scale in terms of the five emotional states mentioned in the previous study. In experiment 1, it was reported that subjects who lost to the computer rated the condition as significantly more boring than those who beat it. Regarding experiment 2, all subjects reported that they found playing against a friend more enjoyable than playing against a computer — this was clearly reflected in the GSR responses (Figure 2.3). Because of this, it was hypothesised that a correlation may exist between GSR and one of the emotional states. Using Pearson's correlation coefficient $r$, it was found that normalised GSR was correlated with normalised fun, with a coefficient value of $r = 0.7$ and a p-value of $p = 0.026$ (Figure 2.4).




(a) Participant 2 scoring a goal.




(b) Participant 9 engaging in a hockey fight.

Figure 2.3: GSR responses against time for specific game events for certain participants. Taken from [37].

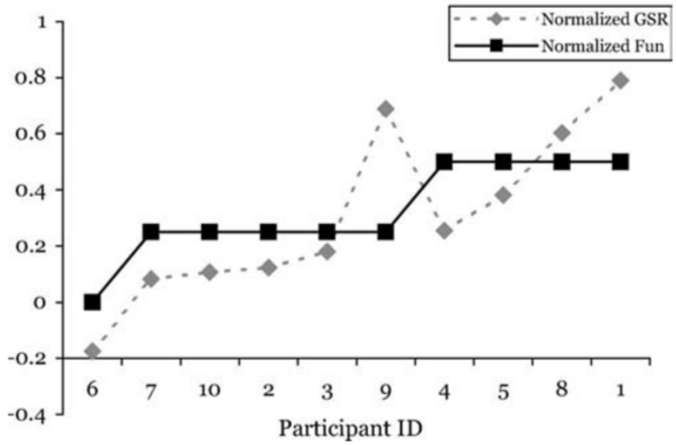



Figure 2.4: Graph showing the correlation between normalised GSR and normalised fun. Taken from [38]. Note: the participants have been ordered according to normalised fun, however the lines drawn between the points carry no meaning.

When one considers the use of human participants, neural networks and the aim of building a model which produces a quantity representing the entertainment value of a game; the quantitative studies involving Pac-Man and the Playware Playground have the largest overlap with our work than any of the other existing PEM literature. However one major difference is their derivation of formulae with the inclusion of arbitrary parameters — while we carry this out for predicting playthrough feedback, this is not the method we employ for quantifying the enjoyment value for game levels. Instead we map features from the players' gameplay to a quantity.

## 2.2 Evaluating Levels — the Sentient Sketchbook

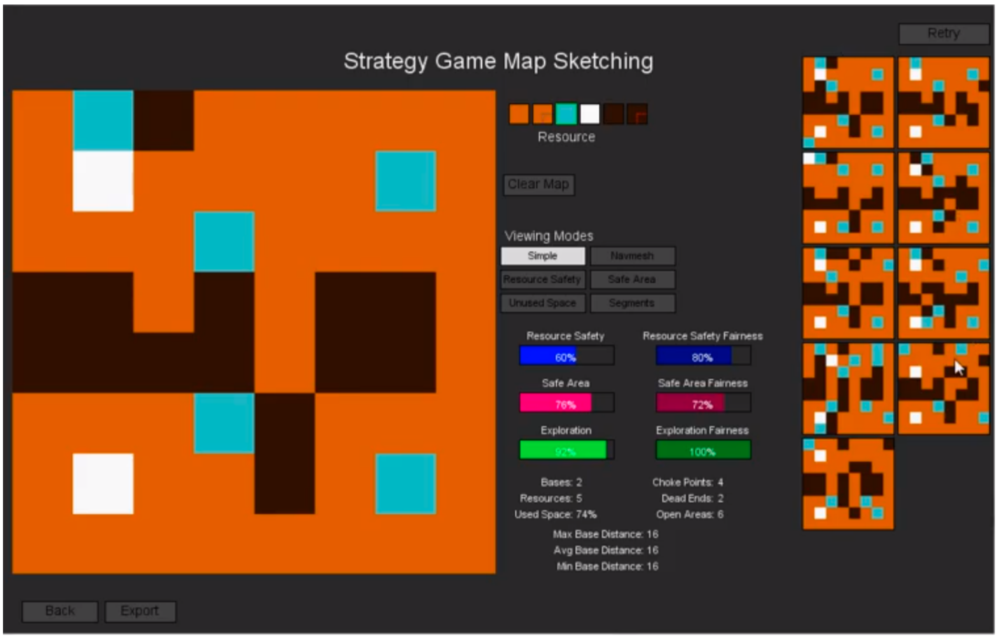
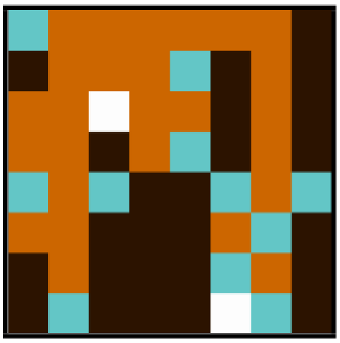


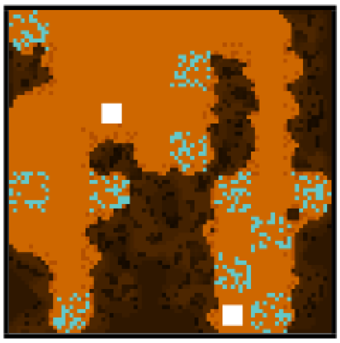

Figure 2.5: User interface of Sentient Sketchbook. The user draws a sketch on the left, the suggestions are on the right, and the evaluations are in the middle. Taken from [31].

The Sentient Sketchbook [33] is a tool which gives feedback in the form of metrics and suggested improvements, in order to assist designers in the creation of game levels. This is achieved via a computer-aided sketching interface, shown in Figure 2.5. The motivation for its creation was the fact that the various models discussed in section 2.1.2 cannot be applied to content outside their domain. Reaching a domain-independent framework requires devising more general functions that abstract away from game-specific features towards more high-level concepts, while retaining their applicability to a particular game. This involved the use of game design patterns [7]. Originally introduced in 2004, they describe part of the interaction possible in a game. The patterns which translate well to level design are:

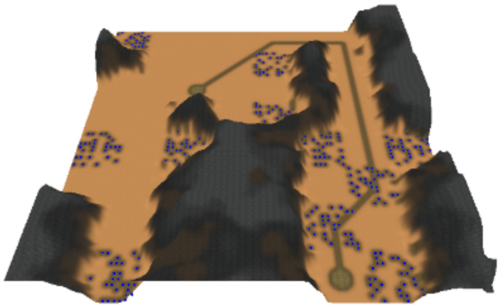
- Control — giving access to otherwise unavailable actions and making the use of tactics and actions easier. It encompasses both area control and resource control.
- Exploration — the goal of learning the layout of the world.
- Symmetry — ensuring players have equal opportunities and instantiating team balance.



(a) Map sketch.



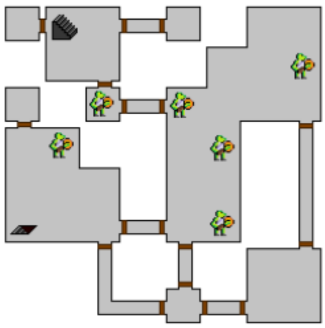
(b) Level (2D).



(c) Level (3D).

Figure 2.6: Map sketch with the corresponding RTS level. White tiles represent bases, cyan tiles are resources and dark tiles are impassable. Taken from [34].

(a) Map sketch.



(b) Level (2D).

Figure 2.7: Map sketch with the corresponding roguelike dungeon level. White tiles represent entrances/exits, cyan tiles are enemies and dark tiles are impassable. Taken from [47].

Quantifiable metrics [32] based on these patterns were used to evaluate abstractions of the game levels known as map sketches (see Figures 2.6 and 2.7). A map sketch consists of a grid layout with empty tiles (allow movement), impassable tiles (block movement) and special domain-dependent tiles (e.g. bases, spawn-points and traps). Area control and exploration require a set of two or more reference tiles ($S_N$) — these have a special purpose in-game e.g. player bases in RTS games. For control measures, a group of tiles can be "owned" by a reference tile if that reference tile is closer to those tiles than the other reference tiles. This is reflected in the safety of a tile $t$ to a reference tile $i$, which is given by:

$$s_{t,i}(S_N) = \min_{\substack{1 \leq j \leq N \\ j \neq i}} \left\{ \max \left\{ 0, \frac{d_{t,j} - d_{t,i}}{d_{t,j} + d_{t,i}} \right\} \right\}, \tag{2.9}$$

where $d_{t,i}$ is the distance from tile $t$ to element $i$. $s_{t,i} > 0$ for the closest reference tile $i$, while $s_{t,i} = 0$ for the remaining $i$ (see Figure 2.8(a) for an illustration of safety).
The exploration required from reference tile $i$ to all other reference tiles is:

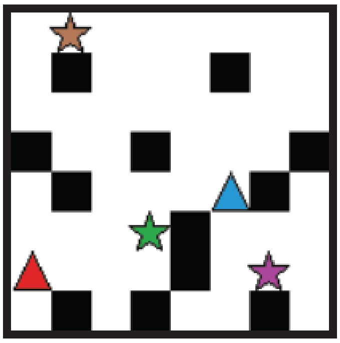
$$E_i(S_N) = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{E_{i \to j}}{P}, \tag{2.10}$$

where $E_{i \to j}$ is the map coverage when a four-direction flood fill algorithm is applied starting from $i$ and stopping once $j$ has been found (see Figure 2.8). $P$ is the number of passable tiles in the map. $E_i$ is high when a large part of the map must be covered in order to discover one reference tile when starting from another reference tile. For resource control, a definition of the tiles representing strategic resources is required — these are known as target tiles ($S_M$). Finally to ensure that reference tiles are symmetric in terms of control and exploration, evaluations of balance are introduced. This gives rise to a total of six objective functions used to evaluate levels in the Sentient Sketchbook: strategic resource control ($f_s$), area control ($f_a$), exploration ($f_e$), strategic resource control balance ($b_s$), area control balance ($b_a$) and exploration balance ($b_e$);

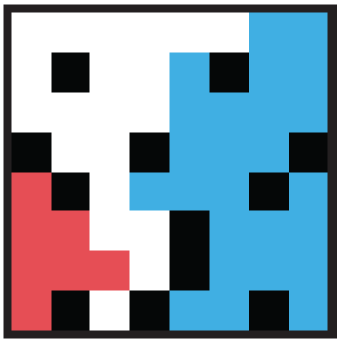
their mathematical formulations are:

$$
\begin{aligned}
f_s(S_N, S_M) &= \frac{1}{M}\sum_{k=1}^{M}\max_{1\le i\le N}\{s_{k,i}\}\\
f_a(S_N) &= \frac{1}{P}\sum_{i=1}^{N}A_i\\
f_e(S_N) &= \frac{1}{N}\sum_{i=1}^{N}E_i\\
b_s(S_N, S_M) &= 1 - \frac{1}{MN(N-1)}\sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}|s_{k,i} - s_{k,j}| \qquad (2.11)\\
b_a(S_N) &= 1 - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}\frac{|A_i - A_j|}{\max\{A_i, A_j\}}\\
b_e(S_N) &= 1 - \frac{1}{N(N-1)}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}\frac{|E_i - E_j|}{\max\{E_i, E_j\}},
\end{aligned}
$$

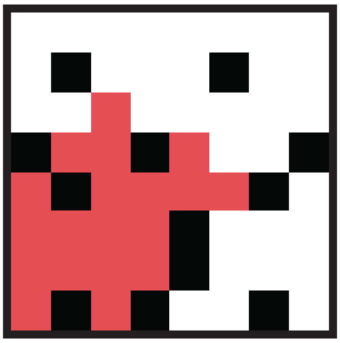
where $A_i$ is the map coverage of safe tiles for element $i$ (see Figure 2.8(b)).
Map sketches are optimised towards the above measures of quality via artificial evolution, and are encoded in a way which reduces representational biases and does not hinder optimisation [32].
The evaluation method of Sentient Sketchbook was demonstrated on a multiplayer RTS and a singleplayer roguelike dungeon game. Table 2.2 shows what some of the evaluation metrics correspond to in each of these genres.
When suggesting alternative map sketches, there is a minimal criteria for playability — all special tiles must be connected to each other via a passable path. This was secured through a Feasible-Infeasible two-population genetic algorithm (GA).



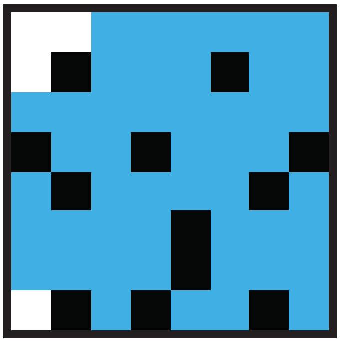
(a) Map sketch.



(b) Map coverage $A_1$ and $A_2$.



(c) Exploration $E_{1\to2}$.



(d) Exploration $E_{2\to1}$.

Figure 2.8: Sample metrics for a map sketch (a) with $S_M$ (stars), $S_N$ (triangles) and impassable tiles (black). The purple star is closer to the blue triangle than the red triangle, therefore it has a high safety value. The other stars are either equally close to (green star) or equally far away from (brown star) both triangles, therefore they have low safety values. For map coverage of safe tiles from the red and blue triangles (b), $A_1$ is shown in red and $A_2$ in blue. For map coverage during exploration (c and d), the exploration from the red triangle to the blue triangle $E_{1\to2}$ is shown in red, while that from the blue to the red $E_{2\to1}$ is shown in blue. Taken from [32].

| **Metric** | **RTS** | **Roguelike Dungeon** |
|---|---|---|
| Strategic resource control | Easy access to resources. | Placement of treasures close to monsters. |
| Area control | Area control around bases. | Distribution of enemies. |
| Exploration | Discovery of enemy bases. | Winding path from entrance to exit. |

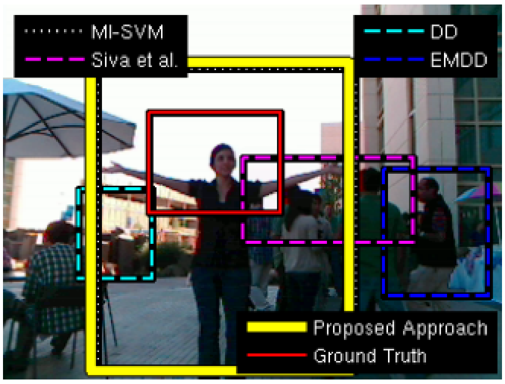
Table 2.2: Metrics used in level evaluation for Sentient Sketchbook, along with what they correspond to in RTS and roguelike dungeon games.

As well as quality, diversity is also important when generating suggestions for game levels. Therefore several search-based approaches [47] were compared in their ability to produce good and diverse content. Similar to “fun”, the definition of diversity within the context of level design is also subjective. In [47], three different measures of diversity for map sketches are used:

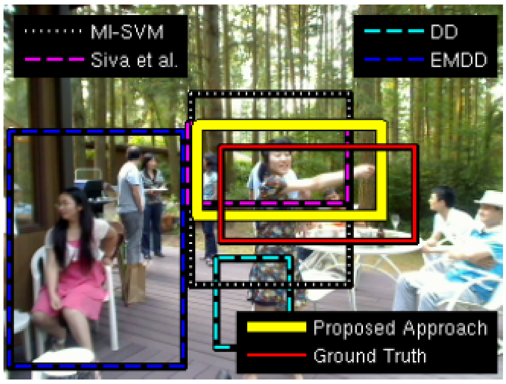
- Tile-based diversity — the number of tiles that differ from one map sketch to another.
- Objective-based diversity — this is measured in terms of the metrics in equation (2.11).
- Visual impression diversity — measured along metrics related to balance or grouping, which are used to extract visual features of maps.

Four optimisation algorithms were used in the study, and the domain was limited to map sketches for RTS games only. All of the algorithms are effective at producing the desired content, provided that appropriate distance measures are used. More specifically novelty search is able to create diverse individuals, however their quality is not as high as those produced from niching evolutionary algorithm 2 (NEA2) [47].

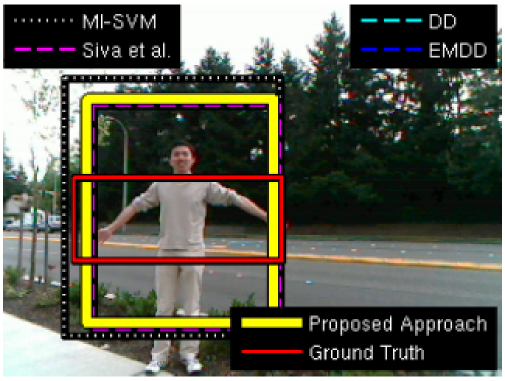
## 2.3 Weakly Supervised Learning




(a) Handwaving.




(b) Boxing.




(c) Clapping.

Figure 2.9: Examples of automated annotation of training data using different MIL algorithms. Taken from [56].

Weakly supervised learning (WSL) [78] is a machine learning task in which all of the elements in the training data set are "weakly labelled" i.e. when the training labels provide less information than what the algorithm outputs at runtime. As an example, consider object detection in images, where the desired output is a bounding box around the object of interest. A weak supervision approach would involve each training image being annotated with a label indicating whether the image contains the object of interest or not. This is different to a fully supervised approach, in which the training images would already include ground truth bounding boxes. The motivation for WSL is that manually annotating all of the elements with ground truth labels incurs high cost [78]. This is why it is a potentially valuable technique for evaluating game levels; giving a single overall rating for a level is less time-consuming than rating individual areas of

that level. As another example WSL has also been applied to microscopy images of cells to better understand the relationships between different drug treatments [8].
There are several types of weak supervision [78]. However in this thesis, only those involving a multiple instance learning (MIL) problem will be discussed [3]. In MIL each element in the training set is represented as a bag containing a set of feature vectors, known as *instances*. A bag is labelled either positive if it contains at least one positive instance (one with the desired information), and negative if it contains no such instances. Therefore, while the label of the bag is known, the individual labels of the instances that conform the bag are unknown. When carrying out MIL for classification, one can be thought of as asking the question: "What am I seeing in these positive bags that I am not seeing in these negative bags?". Classic MIL uses two types of information to train a classifier:

- *Intra-class* — this focuses on the similarities between the positive instances.
- *Inter-class* — this focuses on the differences between the positive and negative instances.

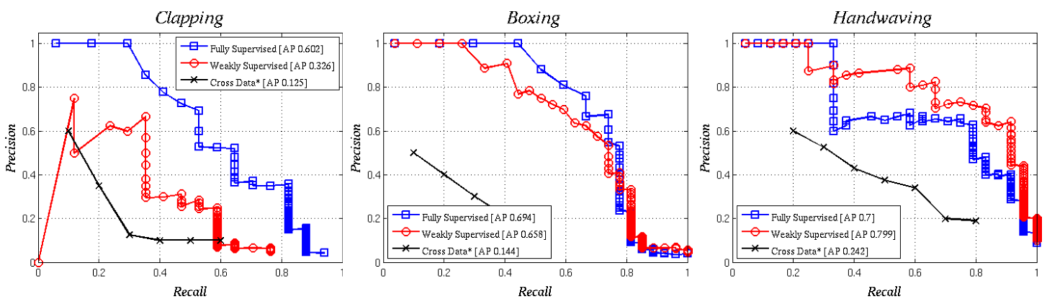



Figure 2.10: Test data detection precision recall curve, taken from [56]. AP = Average Precision. Cross data results as published in [9].

An example of the use of MIL for action detection in videos is the approach taken by Siva and Xiang [56], where their goal was to train a classifier which could detect and identify a specific action taking place in a video. In this problem, the MSR2 training data set consists of a set of positive videos containing at least one action of interest, and negative videos containing no actions of interest. An action is represented by a spatio-temporal cuboid/volume in the video, also known as an action cuboid. Each action cuboid is described by a bag-of-words (BOW) histogram containing 2000 bins derived from Spatio-Temporal-Interest-Point (STIP) features [28]. 100000 randomly selected STIPS from the dataset are clustered into 2000 code words via k-means clustering. Since a typical video instance contains approximately $10^9$ action cuboids, MIL would be intractable. Therefore the number of feasible instances needs to be limited. First an initial set of instances $C'$ is created using a person detector run on every $F$th frame. These are then ranked based on STIP density and temporal spread and the first $M$ cuboids from this ranked list are selected, giving the following sets:

$$\begin{aligned} C_i^+ &= \{C_{i1}^+, C_{i2}^+, \ldots, C_{iM}^+\} \\ C_i^- &= \{C_{i1}^-, C_{i2}^-, \ldots, C_{iM}^-\}, \end{aligned} \tag{2.12}$$

where $i = 1, 2, \ldots, N^{\pm}$ is the number of positive/negative videos. The goal is to select a set $G^* = \{c_1, c_2, \ldots, c_{N^+}\}$ containing one instance from each positive video such that each instance is the action of interest. The following cost function is minimised:

$$G^* = argmin \sum_{c_j \in G} \left[ D(c_j, G_{-j}, k_p) + (1 - D(c_j, C^-_{i=1,\ldots,N^-}, k_n)) \right], \tag{2.13}$$

where $G = \{c_1, c_2, \ldots, c_{N^+}\}$ is a set composed of one instance from each positive bag and $G_{-j}$ is $G$ excluding $c_j$. $D(c, M, k)$ is the distance from the instance $c$ to the set of instances $M$ with constant parameter $k_x$ ($x = p$ for positive and $x = n$ for negative) [56]. The first term in equation (2.13) aims to minimise intra-class distance and the second term aims to maximise inter-class distance. Solving equation (2.13) is achieved using a genetic algorithm — this involves a population of random candidate solutions evolving through reproduction and random mutation towards an optimal solution $G*$. Given $G*$ and a set of negative videos, a support vector machine (SVM) is then trained as an action cuboid classifier. Figure 2.10 features precision recall curves (PRCs) [77] comparing the detection performance of the weakly supervised detector with that of a fully supervised detector. On handwaving, it can be seen that using WSL achieves a higher average precision. However (as stated by the authors) this can be attributed to biases in the manual annotations — in some videos the ground truth bounding boxes do not encompass the entire extent of the hand motion (see Figure 2.9a), and therefore do not include all of the useful STIPs. For boxing and clapping, the weakly supervised detector achieved similar and worst performance compared to the fully supervised detector, respectively. However in the latter case the weakly supervised detector was still able to achieve over 50% of the precision of the fully supervised detector [56]. The authors suggest that the relatively poor performance of the clapping detection is due to the clapping class having fewer training samples, as well as the highly symmetric nature of clapping.

The above approach provides the main inspiration for our use of WSL in game levels to identify the most significant enjoyable moments.

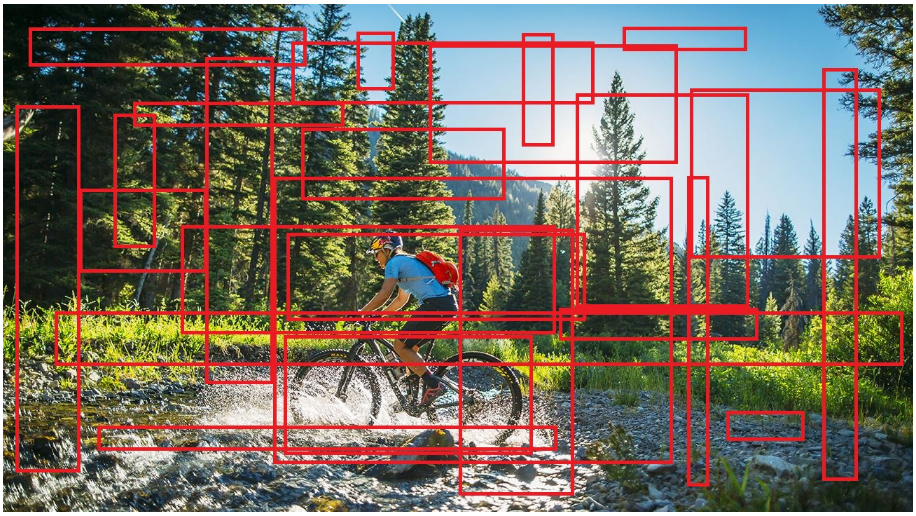


Figure 2.11: Initial instances generated for an image to be used in an MIL algorithm, taken from [62].

Another method by Siva and Xiang [55], with the goal of object detection in images, avoids using intra-class information. This technique is referred to as *negative mining*, as it uses inter-class information and another piece of information known as *saliency*. In this problem, the bags are the images and the instances are randomly generated rectangles which act as bounding boxes within the images (Figure 2.11). Each rectangle is given an "objectness" score reflecting the likelihood of it containing an object (not necessarily the object of interest). This measure of objectness was presented by Alexe et al. [2], and is based on four types of image cues combined in a Bayesian framework. A set number of the rectangles (e.g. the top 100) are then selected based on their objectness scores — these are the initial instances. Each instance is represented by a BOW histogram, whose bins are derived from the clustering of feature vectors extracted from the images [12]. Saliency refers to the knowledge about the size and location of the object in the photo — it is used to prune the space of possible locations a priori [2], and works on the assumption that the images were taken by a human. Negative mining is a viable option here due to the fact that the PASCAL VOC 2007 data set was used. In this data set, a typical class has around 300 positive images $I_i^+$ and 4700 negative images $I_i^-$. 100 candidate instances $x_{i,j=1,2,\ldots 100}$ are generated using a saliency measure, giving 470000 negative instances vs 300 objects located somewhere in 30000 images. This results in an intra-class distance based on less than 300 unlabelled similar positive vs an inter-class distance based on 470000 strongly labelled negative instances. Clearly one can gain substantially more information from the latter, hence the sole use of inter-class. The goal is to select a single instance $x_i^+$ from each $I_i^+$ corresponding to the location of the object — negative mining does this by selecting the instance that maximises the distance to the nearest neighbour in any image containing only negative instances $x_{i,j}^-$, reflected in the following cost function [55]:

$$x_i^+ = argmax_{x_{i,j}^+} ||x_{i,j}^+ - N(x_{i,j}^+)||_1, \tag{2.14}$$

where $||\dots||_1$ is the $\mathrm{L}_1$ norm and $N(x_{i,j}^+)$ is the negative nearest neighbour (NNN) of $x_{i,j}^+$, which is determined using a kd-tree based approximate nearest neighbour algorithm. If there is a saliency measure $\phi$ which serves as a prior of how likely an instance is to be positive, this can be added to equation (2.14), giving

$$x_i^+ = argmax_{x_{i,j}^+} \left[ ||x_{i,j}^+ - N(x_{i,j}^+)||_1 + \phi(x_{i,j}^+) \right]. \tag{2.15}$$

Figure 2.12 shows the results of various classifiers — it is clear that the combined negative mining and saliency method gives the best performance. The fact that this method relies on the abundance of known negative instances means no intra-class optimisation is necessary, leading to increased computational efficiency. It was also observed that using normalised histograms results in greater NNN distance for small instances than that for large instances; unnormalised histograms lead to a bias towards large instances. Using root-normalised histograms minimises the bias towards either size [55].
While negative mining proves to be a successful and efficient technique for object detection in images, it is not used in our WSL implementation due to limitations in the gameplay data.

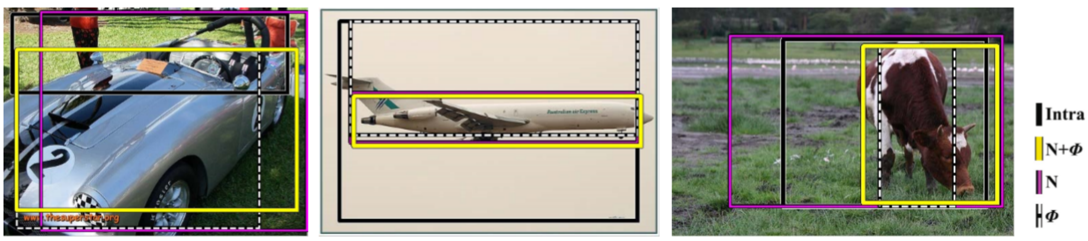



Figure 2.12: Results using intra-class, negative mining (N), saliency ($\phi$) and combined negative mining and saliency methods, taken from [55].

The PASCAL VOC 2007 data set was also used for training in the MIL approach taken by Song et al. [59] for object detection. Their approach differs from that of Siva and Xiang [55] in that intra-class information is used in addition to inter-class instead of relying entirely on negative mining and saliency. The initial instances are selected based on the search technique by Uijlings et al. [66] as opposed to the earlier mentioned objectness measure [2]. Positive instances are selected using a discriminative submodular cover algorithm and the MIL objective is optimised using a latent SVM with Nesterov smoothing [43]. This technique results in a 50% relative improvement in average precision over Siva and Xiang's approach. The authors also note that they use a different evaluation metric to report their results on the test set [59].

The WSL techniques discussed in this section all involved the training of SVMs; in our approach neural networks are trained instead. Additionally instead of initially selecting random instances of varying sizes, the sizes of our instances are equal across all playthroughs. Therefore biases towards instances of a particular size are not a concern.

# Chapter 3

# Experimental Design

The fundamental thing required for any machine learning task is training data. In this chapter we will discuss the game whose data was harvested for our approach, as well as the types of data which were gathered. This includes the data for both training the system and assessing its performance.

## 3.1 The Game

### 3.1.1 Requirements

The game needs to fulfil certain criteria, mainly it must have appropriate analytics. The goal of this research is to assist level designers, therefore the analytics must contain information about the player's interaction with their surroundings. Also these surroundings must be relevant to the gameplay. Therefore we require a game for which it is possible to gather behavioural data i.e. the actions of players, as well as regular updates of their locations within the level. Also because our system is intended to predict player enjoyment, we need a game containing features that allow players to signal their enjoyment (or lack thereof). Ubisoft gathers and stores analytics about most of its games. Using this pre-existing telemetry data is ideal as this project requires a large data set. While the types of available data would be confined within the scope of what is tracked by Ubisoft, creating/modifying a game and releasing it to players purely for the purposes of data collection is very time-consuming. Additionally while other forms of data e.g. physiology may be useful, this requires organising controlled play sessions which participants may find invasive — this has the potential of introducing biases into the data.
Several Ubisoft titles were considered and investigated for their suitability. The collaborative culture that exists across Ubisoft studios means that we do not have to limit ourselves

to games developed solely by Ubisoft Reflections. Accessing the data for a specific game requires permission from the main studio/team involved in its development — this presents a major obstacle, given the fact that some teams are unwilling to share their data with a project involving an external entity e.g. a university. This was the case with some games in the Tom Clancy's series, which were attractive options due to their player feedback features. Other teams, while willing to share their data, did not possess the appropriate analytics. This was the case with games such as Trials, Trackmania and Starlink, in which player positions are not tracked. These factors culminated in the selection of For Honor as the game to use for the project, since we successfully gained access to its telemetry data, which contained the required analytics. Additionally we were provided with tracking tags which allowed us to interpret the database with less difficulty.

### 3.1.2 Description

Figure 3.1: Screenshot of the game For Honor. Image approved for public use by Ubisoft.

For Honor is a medieval action game developed and released by Ubisoft Montreal in 2017 (Figure 3.1), it features both a singleplayer campaign and several multiplayer modes. Players fight against their opponents using class-specific melee weapons. Performing certain actions such as killing multiple enemies consecutively allows a player to gain Feats — additional perks which grant the player certain abilities. A maximum of four

Feats can be equipped at a time.

One of For Honor's multiplayer modes is called Dominion [14]. This consists of a four-versus-four match in which players must capture and hold multiple positions in a battlefield. There are three of these points, A, B and C (see Figure 3.5). Points are earned through occupying the zones and killing the significantly weaker AI enemy minions that fight at point B. Players earn double points for staying on points A and C. When one team reaches 1000 points, the other team starts to "break" — respawning is disabled for that team except through revival by other teammates. Once all of the breaking team's members are killed, the opposing team wins. The breaking team can make a comeback if they reduce the other team's score below 1000 by taking zones, thus regaining the ability to respawn and preventing a loss [1]. At the end of the game, players have the option of giving feedback in the form of a rating out of five stars.

## 3.2 Data Collection

There are two kinds of data which are used for the project: playthroughs and user feedback. The former is used to train the system while the latter is used as ground truth to evaluate the system's performance. The project is a Weakly Supervised Learning (WSL) problem in and of itself. This is because the playthroughs do not contain player feedback for individual regions of the levels (the intended output of the system), but have ratings for the overall match instead. We then compare the system's output to that of users, in a similar manner to how WSL-trained object detectors have their outputted bounding boxes compared to ground truth boxes (Section 2.3).

[1] The following link contains gameplay footage of a typical Dominion match: `https://www.youtube.com/watch?v=sp3NKQlJPuo`

### 3.2.1 Playthroughs

| Map | Playthrough count |
|---|---|
| Citadel Gate | 424 |
| Overwatch | 180 |
| Sanctuary Bridge | 189 |
| River Fort | 67 |
| High Fort | 78 |
| The Shard | 156 |
| **Total** | **1094** |

Table 3.1: No. of playthroughs associated with each For Honor map.

The collected data consists of aggregated information on the in-game activity of players in Dominion matches from Febuary to April 2017, played across 6 different maps (see Table 3.1 and Figure 3.5). The choice of only using Dominion matches is due to the fact that it consists of 1 round per match and at its core, is about control of the map. Therefore geometry plays a more significant role in this game mode than in others. Modes such as Duel consist of multiple rounds confined to a relatively small area where the geometry of the map would have no clear influence on the feedback. Also the presence of multiple rounds introduces an additional layer of complexity, since there is only one opportunity to give feedback — it would be non-trivial to determine how each round influences the final rating. Each playthrough is considered to be a single data point — a playthrough consists of the in-game activity of one player during one match. In total, over 1000 playthroughs (Table 3.1) were collected for training; the information contained within them can be divided into three categories:

1. Positions — the player's $(x, y, z)$ coordinates are recorded every 3 seconds. The orientation of the player at each position is estimated by computing the direction vector between two consecutive positions (see Figure 3.2). However this estimation does not account for motions such as reversing or strafing. The player's bounty level (number of equipped Feats they have unlocked) at each position is also recorded.

2. Actions — these include kills, deaths, attacks, defends, dodges, point changes and special kills. Some of these actions themselves consist of different types e.g. death by hit, falling, fire e.t.c. The attack and defend events also output player health —

this is used to calculate the net change in the player's health over specific periods of time.

3. Feedback — a rating out of five stars given by the player at the end of a match.

Map geometry is essential information to acquire for training. Each map is represented by a polygon mesh, which is used in combination with the positional data to infer the local geometry surrounding the player throughout the match. This is achieved using ray-casting with BSP trees [24]. A UV sphere is created for each position coordinate, with rays projecting outward from the surface normals of the polygon defining the sphere. Then the intersection between these rays and the mesh's polygons are computed. Figures 3.2 and 3.3 help to illustrate the information gathered as the player moves along a path.

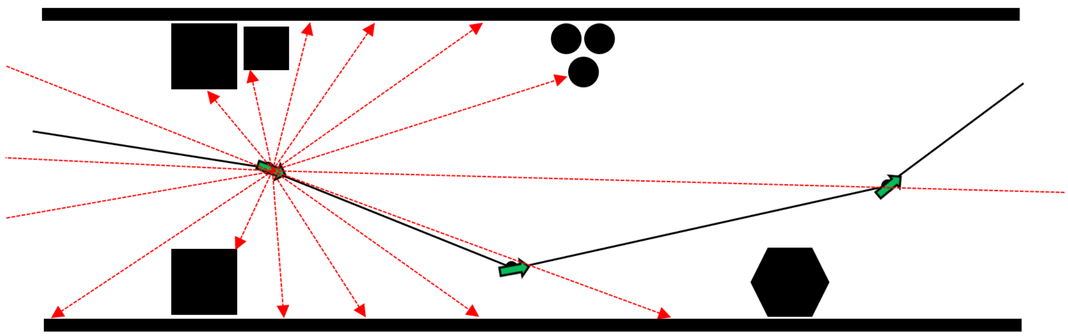


Figure 3.2: Top-down illustration of a typical player path (black line) through a level, their estimated orientation at each position (green arrow) and ray-casting from one of the positions (red arrowed lines).

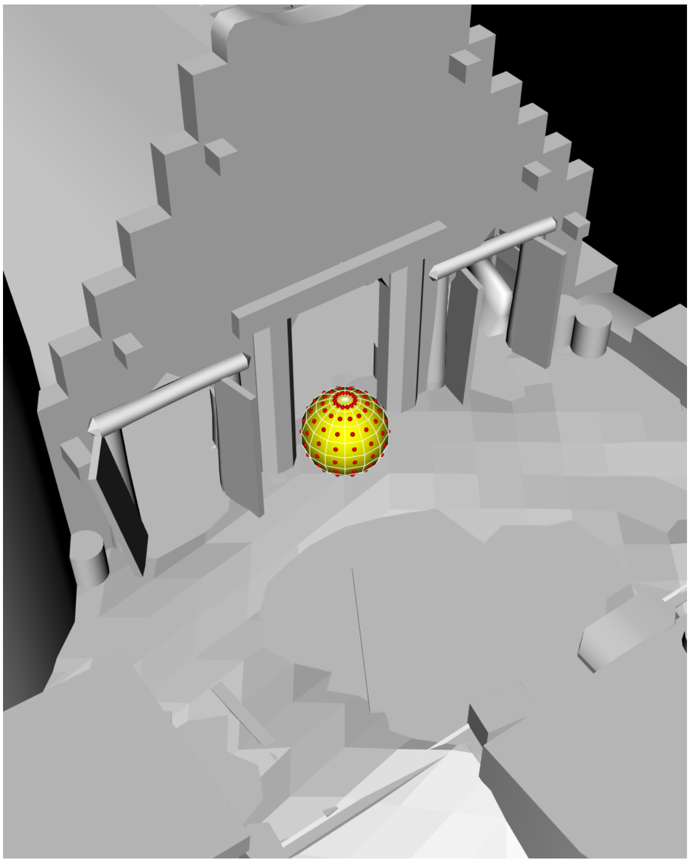


Figure 3.3: A UV sphere (yellow) at one of the player positions in a playthrough, with its corresponding map. Every smaller red sphere represents a point where a ray is emitted.

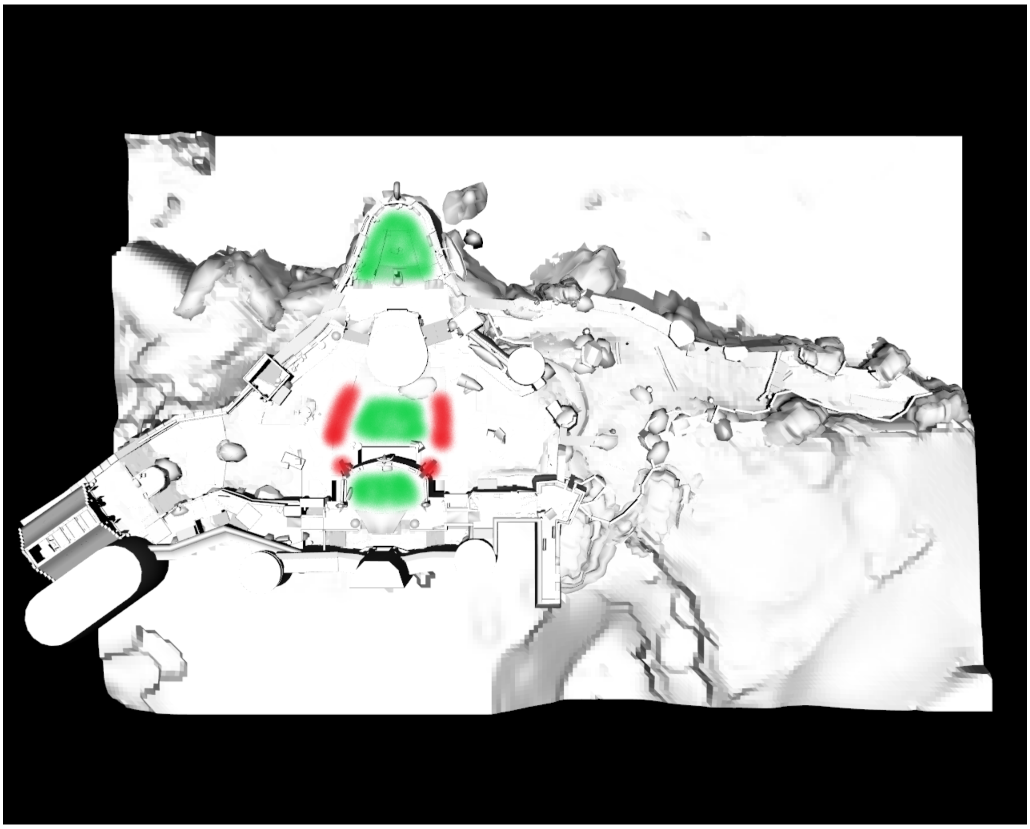


Figure 3.4: Example of a coloured map produced by a user for use as ground truth to evaluate the performance of our feedback tool. Areas highlighted in green (red) are those the user deemed to be good (bad), while areas left uncoloured are considered neutral.

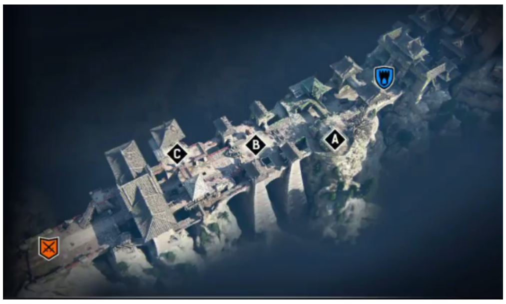
### 3.2.2 User Feedback

In order to establish the performance of the system, its outputs must be compared to a ground truth. Since the feedback is visualised by highlighting areas of the map, the ground truth would ideally be in a similar form i.e. maps which have been manually highlighted by humans. A study was carried out in which participants were asked to play Dominion matches on the For Honor maps used in the project, before highlighting the areas of the map they did/did not enjoy. The instructions given to each user can be found in Appendix A. The original intention was to utilise Ubisoft Reflection's User Research Lab to hold play sessions in which participants could play the game and colour the maps afterwards. However the COVID-19 pandemic had begun during the planning phase of the study, resulting in the closure of the lab. Therefore we reached out to all employees of Ubisoft Reflections and Ubisoft Leamington, who could access the game for free through their Uplay accounts, and play it within their own homes. Those who expressed interest in participating were given the instructions and told to play in their own time. It is important to note that this introduces a self-selection bias, the effect of which can be seen in the fact that the majority of individuals were familiar with For Honor and had played the game prior to participating in the study. After playing several matches on a particular map, the user would employ a graphical interface to highlight the areas of the map they deemed to be good, bad or neutral. An example of this is featured in Figure 3.4. Players were also asked to provide qualitative feedback on why they coloured the map the way they did. The weaknesses of this approach are discussed later in Section 6.1. The diversity of the participants is reflected in the feedback they gave — there were mixed opinions for several regions across the maps. For most areas the majority of users had similar feedback, with only one or two disagreeing. However for certain areas the feedback was almost unanimous. For further detail, Appendix B features the full set of coloured maps, along with consolidated qualitative feedback.

*Note:* for the rest of this report, the term "players" will be used to refer to the general audiences of computer games, as well as the people whose gameplay data was gathered in the form of playthroughs. The term "users" will refer to the participants in the study.

(a) Citadel Gate.

(b) Overwatch.



(c) Sanctuary Bridge.

(d) River Fort.

(e) High Fort.

(f) The Shard.

Figure 3.5: Overview of the For Honor maps used in the project, with Dominion spawn locations (orange and blue) and capture points (black), taken from [23].

# Chapter 4

# Methodology

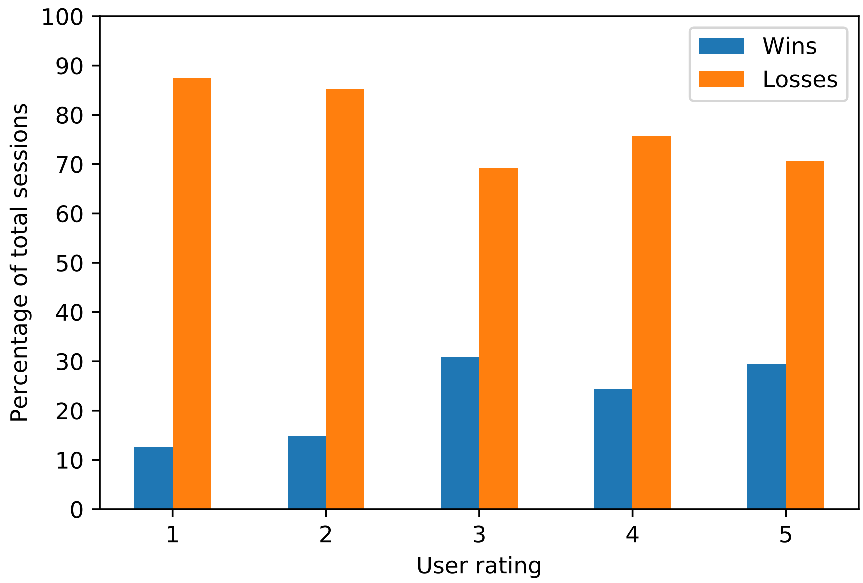



Figure 4.1: Graph showing percentage of playthroughs for each rating which resulted in a win/loss.

Our goal is similar to that of Liapis et al. [32, 33] in their development of the Sentient Sketchbook — to assist the level designer by automatically evaluating the content they have created. However unlike the Sentient Sketchbook which evaluates levels according to specific patterns such as symmetry and area control, the tool we aim to build focuses purely on enjoyment as judged by the player. In order for a system to “know” what features of the level will provide the most (or least) enjoyment to prospective players, it must first be able to derive a mapping between gameplay features and feedback i.e. carry out gameplay-based Player Experience Modelling (PEM) using existing gameplay and feedback data for training. Most of the previous studies on PEM involved the formation of a hypothesis about the root of player enjoyment, followed by an experiment to demonstrate support for this hypothesis. For this project we have made as few assumptions as possible about the sources of player enjoyment, instead relying on machine learning and data mining algorithms to search for patterns in raw gameplay data and analysing their outputs i.e. a model-free approach. The authors of Sentient Sketchbook acknowledge that while map sketches can be applied to genres other than those for which they have performed experiments, the objective functions would need to be adapted and may be less straightforward to optimize [47]. These hard-coded rules may make it non-trivial for a designer who has a specific game in mind — machine learning has the advantage here as it can adapt to specific scenarios and players.

One may be inclined to assume that the player’s feedback is most strongly dependent on whether their team won or lost the match. To test this, the correlation between a team’s win/loss and the given rating was computed using Pearson, Spearman and Kendall correlation (Table 4.1). The results of each playthrough (win/loss) were also plotted against the given rating in a bar graph which can be found in Figure 4.1. The calculated coefficients, as well as an observation of the bar graph, indicate a low correlation between these features. This suggests that other factors must be involved in determining the player’s judgement of a match.

In this chapter the methods by which we achieve our goal of level evaluation are described. As mentioned earlier, creating a mapping between gameplay features and feedback is left to machine learning algorithms. Therefore the features must first be represented in a way which makes it easier for these algorithms to interpret them (Section 4.1). In our case we represent the playthroughs as a series of “moments”. The next step is to select those moments which have the strongest influence on player feedback. We call this *moment detection* and employ two different methods to carry it out — weakly supervised learning (Section 4.2) and a probabilistic regression ensemble (Section 4.3). The selected moments and feedback are then used to train a neural network which can predict feedback, given a set of level features. Finally these predictions are visualised as heat maps (Section 4.4) which are compared to user-generated maps in order to assess the accuracy of our system (Section 4.5).

| Correlation | Value |
|---|---|
| Pearson | 0.16 |
| Spearman | 0.16 |
| Kendall | 0.15 |

Table 4.1: Results of computing the correlation between player win/loss and the rating given at the end of the match.

## 4.1 Feature Representation

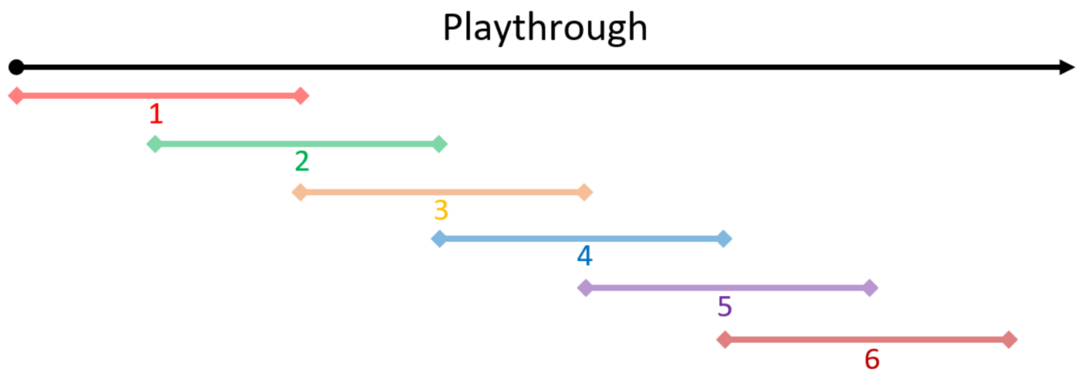



Figure 4.2: Illustration of how a playthrough is divided into chunks with a 50% overlap.

The large amount of data per playthrough leads to a common problem to which game data is subjected: the *curse of dimensionality*. Coined by Richard E. Bellman [6], it refers to the increasing difficulty of finding valid correlations within data as the number of dimensions increases. However this problem can be alleviated via the use of feature engineering and dimensionality reduction techniques. For example when interpreting the motion of a player, their long term motion appears to strongly depend on their location in the map. However by focusing on short sequences of their motion, this becomes less important and can be disregarded, thereby reducing the dimensionality. This need for a more compact representation of data is the same motivation for Siva and Xiang's use of STIP features [56]. They divided their videos into action cuboids described by bag-of-words histograms, derived from clustering STIP features. Similarly, we divide our playthroughs into sequences described by histograms, derived from clustering features. The technique used in this thesis for dimensionality reduction is principal component analysis (PCA) [20]. This allows one to transform data to a lower-dimensional space while keeping most of the original information. The sequences into which we divide the playthroughs contain information about the player's motion, their bounty level (no. of equipped feats), their actions and the geometry which surrounds them. Dimensionality reduction is carried out in a two-fold PCA process: in the first step, events which are tracked regularly are considered (motion, bounties and geometry); in the second step, events which are tracked at their moment of occurrence are considered (actions such as kills, deaths, dodges e.t.c.). Also the time period over which events are considered in the 2nd step is longer than that of the first step. The reason for carrying out a 2-fold process is in order to obtain statistical strength. In terms of player motion, this refers to the fact that when we look at the overall player path in a playthrough, most of these paths are dissimilar to each other. However when we divide a path into shorter motion sequences, we begin to see more of these motions occurring across multiple playthroughs. Clustering the spatial information from the first step and feeding this into the second step means the system does not have to learn those features independently, making it easier for the algorithm to discard irrelevant information. This method also adds non-linearity to the model.

### 4.1.1 Player Motion

Consider a single playthrough $\mathbf{P_x}$ containing all of the player's positions which have been tracked every 3 seconds:

$$\mathbf{P_x} = [\mathbf{x}_0,\ \mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_3,\ \mathbf{x}_4 \dots \mathbf{x}_n], \tag{4.1}$$

where $\mathbf{x}_i = (x_i,\ y_i,\ z_i)$ is the $i$th tracked position of the player during the match and $n$ is the total number of positions in the playthrough. As mentioned in the previous chapter, these are used to estimate the orientation $\phi_i = (\phi_{xi},\ \phi_{yi},\ \phi_{zi})$ of the player at each of these points in time:

$$\mathbf{P}_\phi = [\phi_0,\ \phi_1,\ \phi_2,\ \phi_3,\ \phi_4 \dots \phi_n], \tag{4.2}$$

where

$$\phi_i = \begin{cases} \dfrac{1}{|\mathbf{x}_{i+1} - \mathbf{x}_i|}(\mathbf{x}_{i+1} - \mathbf{x}_i) \text{ for } i \neq n, \\ \phi_{n-1} \text{ for } i = n. \end{cases}$$

Since each position event also contains the player's bounty level $0 \leq b_i \leq 4$, the playthrough in terms of the players bounties at each point in time can be expressed as

$$\mathbf{P}_b = [b_0,\ b_1,\ b_2,\ b_3,\ b_4 \dots b_n]. \tag{4.3}$$

### 4.1.2 Local Geometry

For the geometry surrounding the player, the results of the ray-casting per position $i$ are encoded in a vector $\mathbf{G}_i$ of size $R$ where $R$ is the number of rays projected outward from the $(x, y, z)$ coordinate. Therefore the playthrough in terms of the local geometry is

$$\mathbf{P}_\mathbf{G} = [\mathbf{G}_0,\ \mathbf{G}_1,\ \mathbf{G}_2,\ \mathbf{G}_3,\ \mathbf{G}_4 \dots \mathbf{G}_n], \tag{4.4}$$

where the values of the elements of $\mathbf{G}_i$ (here denoted as $g_r$ where $r$ is the ray index) are dependent on the manner in which the geometry was captured. Several different methods were considered:

1. *Map proximity* — This is based on the idea of lower distances being more important since the game mainly involves close combat. Each ray is projected outward, and at 3 separate distances (Figure 4.3a), the intersection is computed in the following way:

$$g_r = \begin{cases} 0 \text{ if intersection with the map,} \\ d \text{ if no intersection with the map,} \end{cases}$$

   where $d$ is the shortest distance between the ray target and the map's object mesh.

2. *Nested spheres* — In order to better capture the surrounding geometry and compensate for the reduction of ray density with distance, three spheres of increasing radii are used, with the outermost sphere possessing the highest resolution for its polygon (Figure 4.3b). For each ray, its intersection with the map's structure at a fixed distance is computed:

$$g_r = \begin{cases} 1 \text{ if intersection with the map,} \\ 0 \text{ if no intersection with the map.} \end{cases}$$

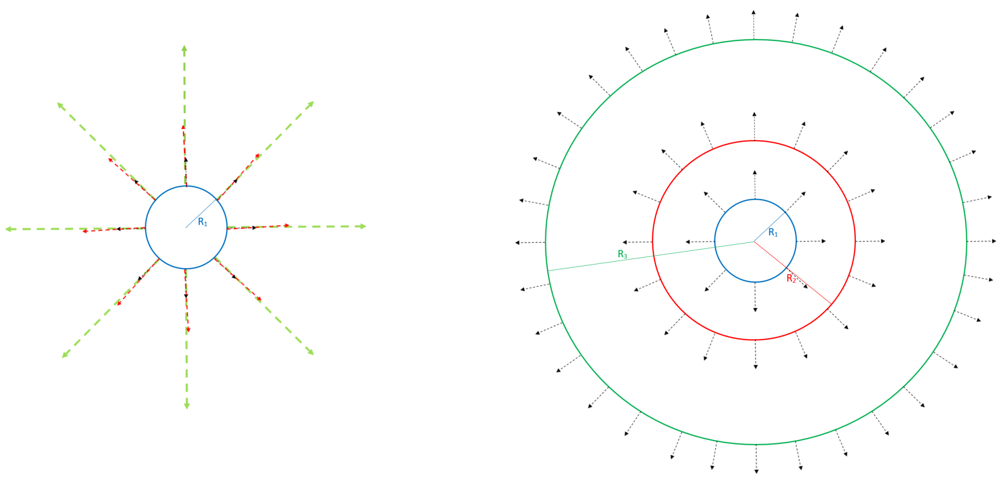
3. *Intersection distance* — Each ray is projected out to a distance $d_{lim} = 200$ from the sphere, and the distance the ray travels before it intersects the map $d_{int}$ is

computed:

$$g_r = \begin{cases} d_{int} \text{ if intersection with the map,} \\ d_{lim} \text{ if no intersection with the map.} \end{cases}$$

4. *Log-distance* — This may provide a better representation of For Honor player behaviour as players are more likely to place importance on events the closer they are to them, especially considering that there is no ranged combat in For Honor. It is equivalent to the intersection distance method but computing the logarithm of the distance instead:

$$g_r = \begin{cases} \log(d_{int}) \text{ if intersection with the map,} \\ \log(d_{lim}) \text{ if no intersection with the map.} \end{cases}$$



(a) Map Proximity. (b) Nested Spheres.

Figure 4.3: Illustration of two ray casting methods to capture the geometry surrounding the player.

The total number of positions across all playthroughs for the Riverfort map is $\sim 10^4$. Due to the run-time costs incurred by computing the geometry vector for every single player position, an approximation of the geometry around the player is used instead. First the positions for a given map are clustered via k-means and the geometry vector around each cluster centre is calculated. Then these geometry vectors are cached for later use. When the geometry surrounding a given position is required, the nearest cluster to that position is computed and its corresponding geometry vector is used. The number of clusters $N_{mc}$ to use for each map is determined by starting with a standard number of $N_{mc} = 1000$ for River Fort, before extrapolating this to the other maps via calculation of their surface areas. This ensures that the number of clusters is proportional to the size of each map. The initial value of 1000 is the result of a compromise between computation time and the distance between a cluster centre and the player's actual position.

### 4.1.3 Zero-meaning & Clustering

Considering the playthrough in terms of all of its features which have been discussed so far, a window of variable size $w$ is used to divide the playthrough into chunks corresponding to a specific period of time. The following illustrates the first chunk for a window of size $w = 4$ (corresponding to a period of 12 seconds, considering the sampling rate of position events):

$$\begin{aligned}
&[\mathbf{x}_0,\ \mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_3,\ \mathbf{x}_4,\ \mathbf{x}_5,\ \mathbf{x}_6,\ \mathbf{x}_7 \ldots \mathbf{x}_n] \\
&[\phi_0,\ \phi_1,\ \phi_2,\ \phi_3,\ \phi_4,\ \phi_5,\ \phi_6,\ \phi_7 \ldots \phi_n] \\
&[b_0,\ b_1,\ b_2,\ b_3,\ b_4,\ b_5,\ b_6,\ b_7 \ldots b_n] \\
&[\underbrace{\mathbf{G}_0,\ \mathbf{G}_1,\ \mathbf{G}_2,\ \mathbf{G}_3}_{\mathbf{C}_{1,1}},\ \mathbf{G}_4,\ \mathbf{G}_5,\ \mathbf{G}_6,\ \mathbf{G}_7 \ldots \mathbf{G}_n]
\end{aligned} \tag{4.5}$$

$$\mathbf{C}_{1,1} = [\mathbf{x}_0,\ \mathbf{x}_1,\ \mathbf{x}_2,\ \mathbf{x}_3,\ \phi_0,\ \phi_1,\ \phi_2,\ \phi_3,\ b_0,\ b_1,\ b_2,\ b_3,\ \mathbf{G}_0,\ \mathbf{G}_1,\ \mathbf{G}_2,\ \mathbf{G}_3],$$

where $\mathbf{C}_{j,k}$ is the $k$th chunk of the $j$th playthrough. Since only the player's motion needs to be captured, the positions in the chunk are zero-meaned — each position is subtracted by the mean of all the positions in the chunk $\mu_{j,k}$, therefore $\mathbf{C}_{1,1}$ can be re-written as

$$\mathbf{C}_{1,1} = [\mathbf{x}_0-\mu_{1,1},\ \mathbf{x}_1-\mu_{1,1},\ \mathbf{x}_2-\mu_{1,1},\ \mathbf{x}_3-\mu_{1,1},\ \phi_0,\ \phi_1,\ \phi_2,\ \phi_3,\ b_0,\ b_1,\ b_2,\ b_3,\ \mathbf{G}_0,\ \mathbf{G}_1,\ \mathbf{G}_2,\ \mathbf{G}_3]. \tag{4.6}$$

This offset is performed in order to create an invariance to the player's location within the level (see Figure 4.4). The chunks in a playthrough are taken in a manner illustrated in Figure 4.2, where consecutive chunks are shifted forward by $\frac{w}{2}$ each time. For example $\mathbf{C}_{1,2}$ would encompass the range $i = 2, 3, 4, 5$ in (4.5). This is continued until the end of the playthrough; the whole process is repeated for the other playthroughs and the chunks are stacked in a matrix $\mathbf{X}$, whose covariance $\Sigma$ is calculated. After this the eigenvalues and eigenvectors of $\Sigma$ are found; each eigenvalue represents the amount of variance along the direction of its corresponding eigenvector i.e. the energy along that dimension. We seek a projection which maximises the variance. Therefore the eigenvectors are stacked in descending order of their eigenvalues to produce a rotation matrix $\mathbf{R}$ where most of the energy is pushed into the first dimension, and the dimensions with little to no energy are discarded (here we keep 99.9% of the energy). **R** is then applied to the original data matrix to transform it to a lower dimensional space and the first two columns of this newly represented data matrix **Z** (corresponding to the two highest-energy features) are plotted against each other (Figure 4.5a).

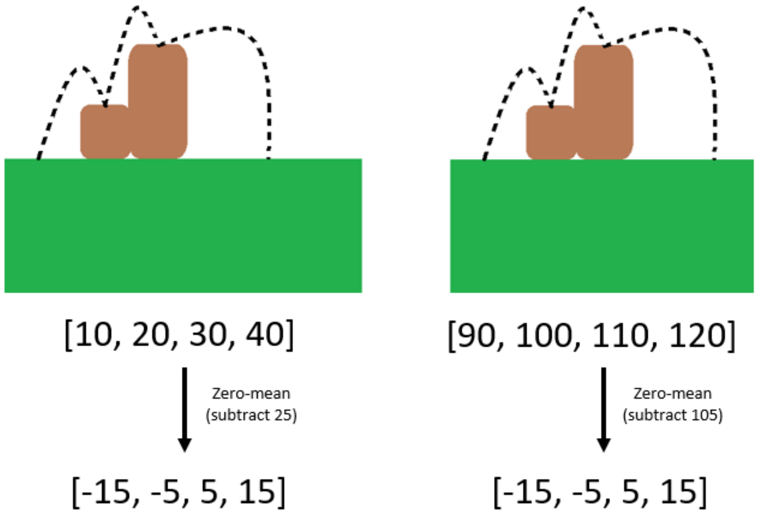
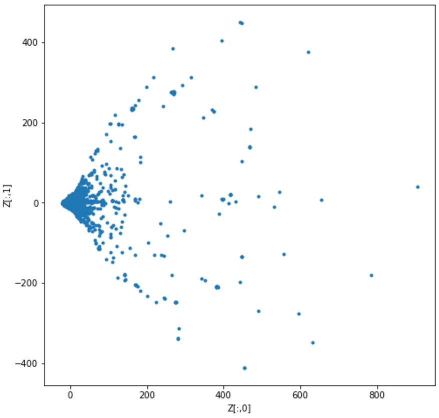
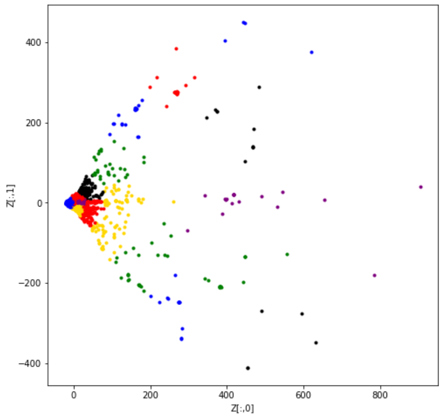



Figure 4.4: Illustration of zero-meaning two x-position sequences in two different locations of a 1-D platformer. After zero-meaning, the resultant arrays are identical, allowing the PCA algorithm to interpret these as two identical motions

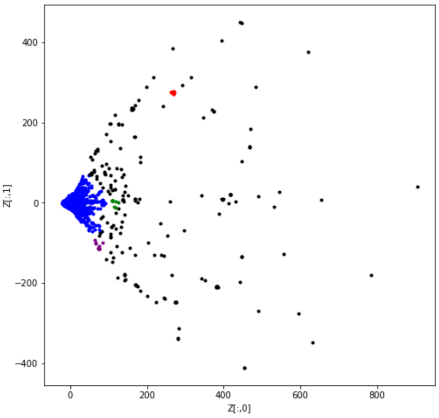
Clustering is then applied to the newly represented data to isolate the various motion/geometry sequences. Three different clustering algorithms were considered for this task: k-means, DBSCAN and mean-shift. DBSCAN and mean-shift were considered because the number of clusters is dynamically determined from the data. Also they are both density-based methods, and hence were thought to have been ideal for isolating the dense structures observed in the PCA results. Figures 4.5b-d feature the results of clustering for all three algorithms. It was difficult to search for a set of parameters which produced a sufficient number of clusters for DBSCAN; mean-shift began to produce similar results to k-means as the bandwidth of the kernel was increased, but required significantly more computation time. Therefore k-means was used throughout the pipeline for the sake of simplicity and speed. We denote the number of clusters derived at this stage as $\kappa_1$.



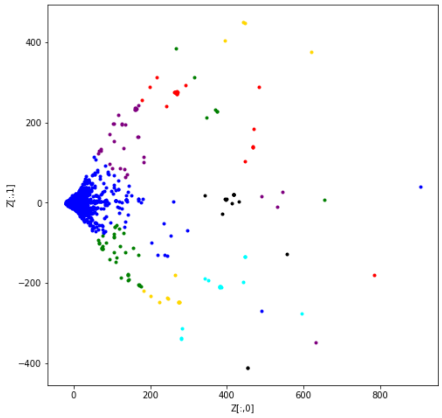

(a) Original.




(b) K-means.




(c) DBSCAN.




(d) Meanshift.

Figure 4.5: Results of PCA (a) and applying different clustering algorithms to them (b-d). The distribution of the points is centered on (0,0) due to zero-meaning and possibly a high frequency of occurrences of the players standing still. The apparent drifting of points to the right is likely due to the presence of bounties in the data, as these are always equal to or greater than zero.

One can infer the in-game motions to which the clusters correspond by extracting their centres, before processing them backwards through PCA. Figure 4.6 shows the results of this. One would expect the existence of these kind of motions in an action game like For Honor.

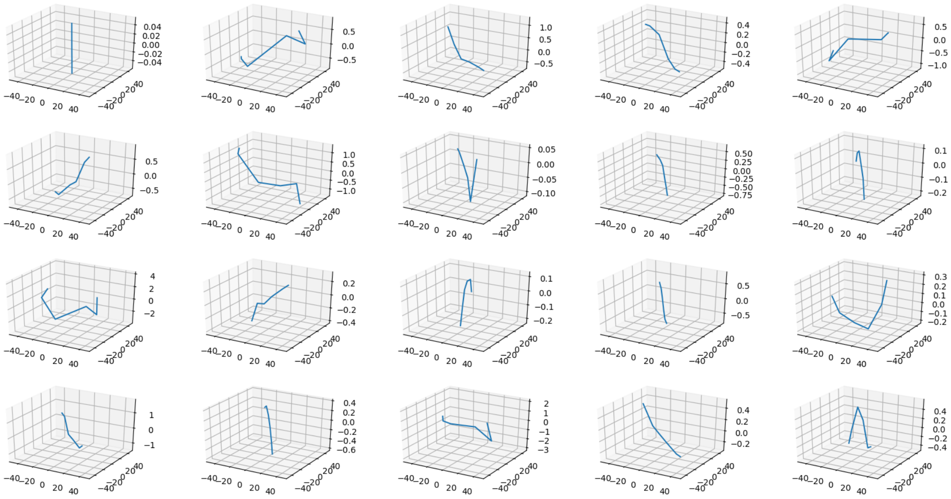


Figure 4.6: Inferred motions from using windows with $w = 6$ and $\kappa_1 = 20$.

### 4.1.4 Player Actions & Clustering

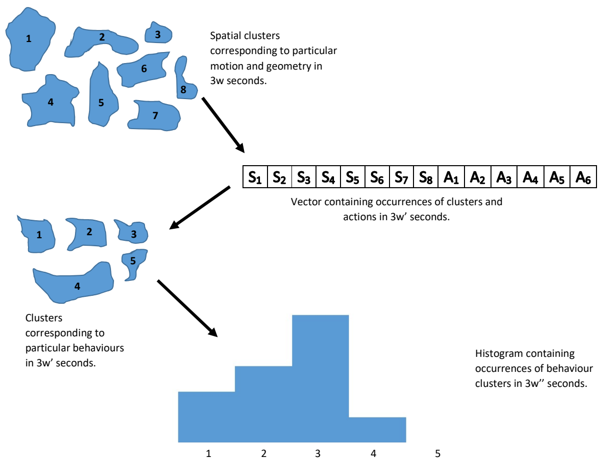



Figure 4.7: The 2-fold PCA and clustering method used for processing the data. In this example $\kappa_1 = 8$ and $\kappa_2 = 5$.

Figure 4.7 illustrates the remainder of the process for clarity. Once specific spatial clusters are found, we repeat the process of dividing playthroughs into chunks as before. However this time we use a window size of $w' > w$. Also instead of filling a vector with the features already discussed, we use the results of 4.1.3 to count the number of times these clusters occur within that window. Therefore the vector for a given chunk may resemble the following:

$$\mathbf{C'}_{j,k} = [S_1,\ S_2,\ S_3 \dots S_{\kappa_1}], \tag{4.7}$$

where $S_i$ is the number of times the ith cluster occurs in the chunk. Actions which occur in this same interval are concatenated on to this vector:

$$\mathbf{C'}_{j,k} = [S_1,\ S_2,\ S_3 \dots S_{\kappa_1},\ A_1,\ A_2,\ A_3 \dots A_{N_A}], \tag{4.8}$$

where $A_l$ is the number of occurrences of the $l$th action in the chunk and $N_A$ is the number of possible actions. PCA and clustering are carried out as before, giving rise to $\kappa_2$ clusters which essentially correspond to player behaviours. Finally, chunks of size $w'' > w'$ are taken per level playthrough; these behaviour clusters are then searched for in these intervals to produce the final bag-of-words histograms containing $\kappa_2$ bins. Each histogram represents a moment in a playthrough.

## 4.2 Moment Detection — Weakly Supervised Learning

The first method of moment detection uses weakly supervised learning (WSL) to select one moment per playthrough which is considered to be that which most strongly influenced the rating given to that playthrough. These moments and their corresponding playthrough ratings are used to train two model: one which can map an unseen playthrough's moments to a score representing the predicted feedback; the other which can map a level's geometry to a predicted feedback score.

### 4.2.1 Multiple Instance Learning

WSL is applied via a MIL approach closely following the method outlined in Section 2.3 for action detection in videos. Here the goal is moment detection in playthroughs. Unlike that method, the histograms are not root-normalised since the chunks they represent are all the same size. Therefore conventional normalisation can be used without worrying about biases towards histograms of a particular size. Also the concepts of objectness and saliency do not translate well to gameplay moment detection, hence we do not introduce these steps during WSL. When searching for “good” moments, the positive bags correspond to playthroughs that received an overall rating of 5 stars, while the rest of the playthroughs correspond to negative bags. The main objective here is: given a set of playthroughs that have been rated 5 stars and a set that have not, select one moment per playthrough which had the strongest influence on that 5 star rating. In other words, select one instance from each positive bag such that the selection minimises equation (2.13). This function is used instead of equation (2.14), since the lack of an abundance of negative instances compared to positive ones invalidates the utilisation of negative mining. Given the number of positive bags, and the number of instances per bag, brute force is not a feasible approach (the number of possible selections can be as high as $\approx 10^{323}$). Solving this involves the implementation of a genetic algorithm, which searches for optimal solutions amongst a population of random candidates. The implementation of the algorithm can be found in Appendix C. A constant mutation rate is used, and new random candidates are added during the optimisation in order to diversify the population and reduce the chance of falling into a local optimum.

In order to locate “bad” moments, WSL is carried out again; this time treating playthroughs with 1 or 2 stars as positive bags, and the rest as negative. An illustration of WSL being carried out on playthroughs to extract “good” and “bad” moments can be found in Figure 4.8.

For both “good” and “bad” moments, WSL is run a second time but with the selected instances removed from all of the positive bags. The reason for doing this is because WSL assumes that there is a single moment per playthrough which most strongly influences the feedback given for that session. However unlike objects-of-interest which occupy a finite discrete number of pixels in an image, moments in a playthrough are continuous — the point at which one moment ends and another begins is ambiguous. Also during a game, consecutive moments may be causally connected. Therefore selecting the second strongest moments is a better reflection of the psychology of player enjoyment. It also allows for slight expansion of the data set used for the next step in the pipeline.

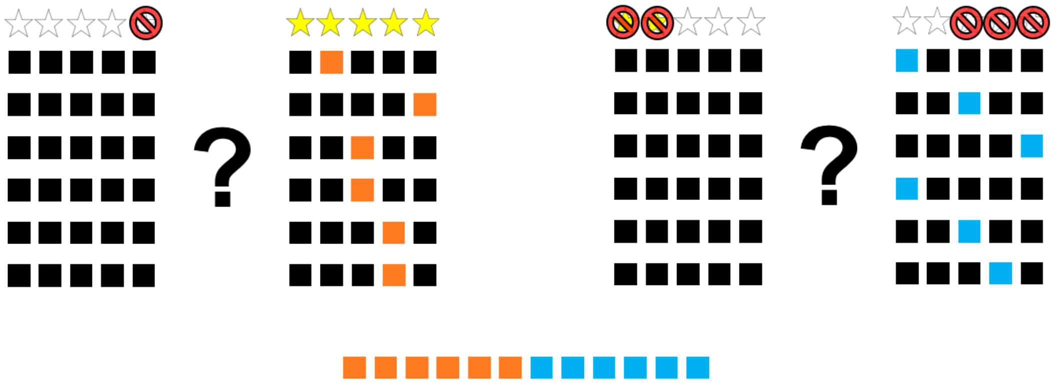
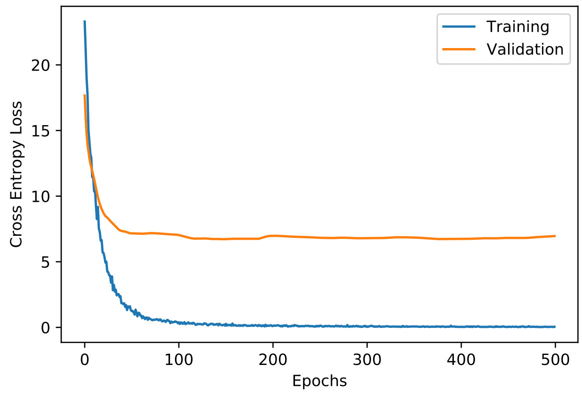


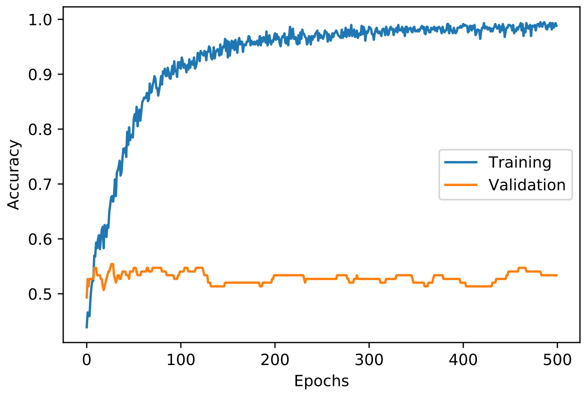
Figure 4.8: Diagram of WSL being carried out on a pair of playthrough sets to extract "good" moments (orange), and on another pair to extract "bad" moments (blue). The star ratings of the playthroughs in each set is shown on top.

### 4.2.2 Model Training

The "good" and "bad" moments obtained from WSL are used to train a model which can serve as either a player experience model for playthroughs, or an evaluation function for levels. Training the model using these moments directly would result in a system that only achieves the former — to take playthroughs as input for predicting their feedback. Therefore in order to build a system which can take in a level i.e. map geometry as input, the training data must also consist solely of geometry. This is achieved by separating the geometry from the other information within each moment. More specifically each WSL-derived moment is located within its playthrough, before only extracting the information associated with the geometry covered within that moment. In the WSL examples covered in Section 2.3, the specific image patches/video segments were used to train classifiers. Here regression is more appropriate because classification will only be able to give a binary output i.e. good/bad; we require a continuous output such as a predicted rating ranging between 0 and 5. Since the input world state of each moment is binary, a neural network is trained using TensorFlow. Cross entropy is used for the cost function, and optimisation is carried out using Adam. The set of moments is split into a training and validation set according to a ratio of 80:20; Figure 4.9 features a visualisation of the optimisation's performance during a typical run. The validation loss/accuracy appears to slightly improve before dropping and staying roughly constant throughout the process — this may be attributed to overfitting, as the optimiser starts to model noise within the training set. Attempts to improve the validation accuracy via regularisation and implementing an ensemble model were unsuccessful.




(a) Cross entropy loss.




(b) Accuracy.

Figure 4.9: Graphical visualisation of optimisation during regression model training for WSL method.

### 4.2.3 Predicting Playthrough Feedback

The trained model is applied to a set of playthroughs from a map which is not used in the original training data. The purpose of this is to assess the system's ability to predict the feedback of an individual playthrough, by comparing its performance to the ground truth (the given rating in this case). These playthroughs contain the same type of data as those used in training. The system evaluates a playthrough by splitting it into chunks of length $w''$, before transforming the information in each chunk into a moment. Then each moment is inputted into the model, returning a rating out of five stars for each interval. At this point, a function is needed which can convert all of these individual ratings into one overall rating for the playthrough. In other words for a given playthrough with rating $R$ containing $n$ chunks, find the function $f$ such that:

$$R = f(r_1,\ r_2,\ r_3,\ \dots\ r_n), \tag{4.9}$$

where $r_i$ is the rating of the $i$th chunk in the playthrough. The way in which a player decides what rating to give would no doubt vary from person to person [17] — this is why several different metrics were attempted for calculating the overall rating:

- Mean — computing the mean is based on the assumption that all moments contribute to the overall rating with equal weight.

- Mode — this is linked to the idea that the player will base their feedback on the most memorable moments. For example if a playthrough's feedback consisted of $[5, 5, 1, 5, 5, 1, 5]$, the 1-star moments would be easily forgotten. In Masthoff's paper this is referred to as "avoiding misery" i.e. ignoring a low rating due to it being overshadowed by the numerous higher ratings [40].

- Weighted average — moderate values are discarded before computing the mean. This is based on the idea that more extreme moments will be more memorable, and that people have an affinity for extremes when it comes to giving feedback. This is apparent in Figure 4.1, as 5 stars and 1 star are the two most frequent ratings given in the training set of playthroughs. In fact this may be why some companies have opted for binary feedback systems, rather than 5-star ratings [25].

- Most extreme — the single most extreme value is taken to be the overall rating. This is simply the motivation of the weighted average metric taken to its greatest extent, and is also linked to the assumption of using WSL.

## 4.3 Moment Detection — Probabilistic Regression Ensemble

The second method of moment detection applies PCA and clustering to the moments. This is combined with the playthrough ratings to form a probability distribution over the rating for each cluster. These moment clusters and their corresponding rating distributions are, through various metrics, used to predict the rating of unseen playthroughs. They are also used to train a model which can map level geometry to a rating probability distribution representing predicted feedback.

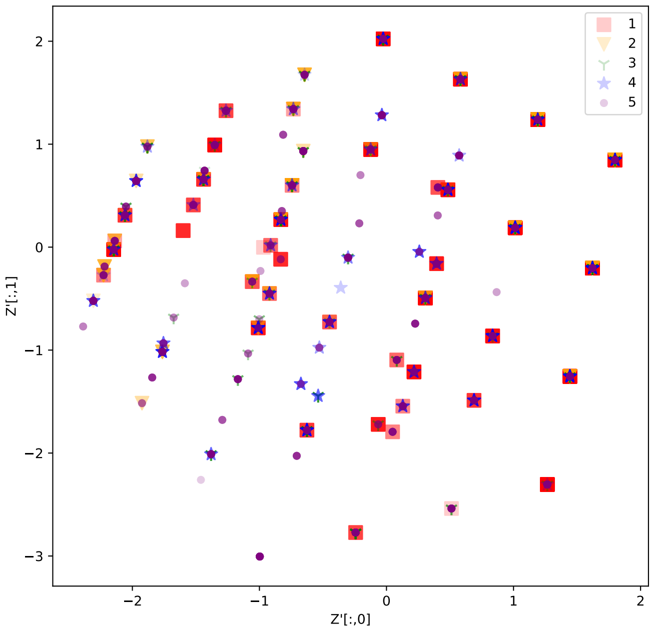



Figure 4.10: Result of applying PCA to the moment histograms, where each moment (datapoint) is colour coded according to the rating of its playthrough.

### 4.3.1 Further Clustering

PCA is applied to the bag-of-words histograms, the results of which are featured in Figure 4.10. Each histogram has been colour coded according to the rating assigned to its playthrough. K-means clustering is then applied to these results, in which each cluster corresponds to a probability distribution over the rating. Figure 4.11 illustrates this entire process for clarity. One can think of this collection of clusters as an ensemble of probabilistic regressors. A playthrough consists of a sequence of clusters (moments). In order to determine the extent to which each cluster influences the player's feedback for that playthrough, one must observe the difference between consecutive clusters. For example a transition between two clusters which are very different to each other could indicate that something important has occurred within the session; something which could significantly affect the overall feedback. Visualising this in 2-D space involves applying Multidimensional Scaling (MDS) to the matrix of Jaccard distances between the clusters. Figure 4.12 illustrates the results of this, as well as the transitions between moments during playthroughs. The jumping between distant clusters indicates the sudden occurrence of something significant, and possibly influential.

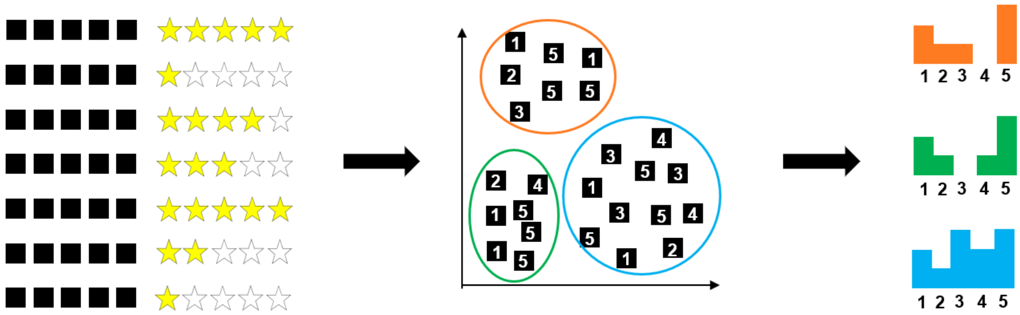



Figure 4.11: Diagram showing how the PRE method involves taking playthroughs with their corresponding ratings (left), applying clustering to their moments (middle), resulting in probability distributions over the ratings (right).

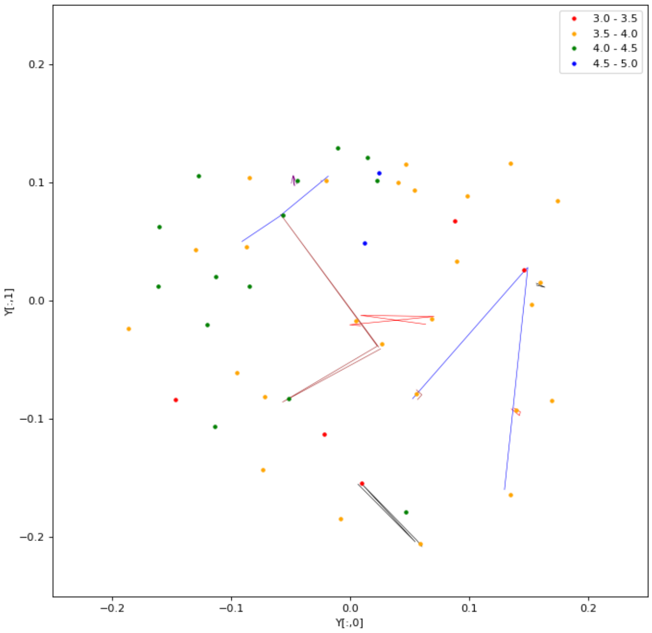



Figure 4.12: Frame from animation of playthroughs in terms of clustered moments — each point represents a specific type of moment and the lines represent the transitions between those moments as the playthrough occurs. The colour of the cluster represents its 5-star rating.

### 4.3.2 Cluster Voting for Predicting Playthrough Feedback

Each cluster corresponds to a probability distribution over the rating, represented as a 5 bin histogram where each bin corresponds to a star. Now that each playthrough consists of a series of distributions, the next step is to formulate a method of combining these distributions so they would correspond to the overall rating of that playthrough. We refer to this method as cluster voting since each cluster (moment) gets a "vote" on how influential it is towards the overall rating. This influence is determined by the weight assigned to it — how do we calculate the weight for each cluster such that for each playthrough:

$$R = \sum_{i=1}^{N} \omega_i c_i, \tag{4.10}$$

where $R$ is the overall rating of the playthrough, $\omega_i$ is the weight assigned to the $i$th cluster and $N$ is the number of clusters in the playthrough? The range of values for $R$ depends on whether our model is binary or multinomial. In a binary model we are only concerned if the feedback is good or bad, corresponding to a value of 1 or 0 respectively. However in a multinomial model it takes a value from 1 to 5. Four metrics were created for the purposes of determining a suitable way of modelling a user's overall feedback as a function of the moments they experienced:

1. Equal weighting — every playthrough is assigned the same weight. This is based on the assumption that every moment contributes to the overall rating equally.

2. Lowest entropy — every moment is assigned a weight of zero except the one with the lowest entropy. This is due to the fact that the distribution with the lowest entropy will be most dissimilar to the others i.e. the moment in which the most significant change occurs during gameplay. This is inspired by WSL as we are choosing one key moment of interest from each playthrough, based on its dissimilarity from all of the other moments.

3. Log entropy weighting — this is an existing entropic method for weight determination of criteria:

$$\omega_i = 1 + \frac{1}{\log(n)} \sum_{j=1}^{n} p_{ij} \log(p_{ij}), \tag{4.11}$$

   where $n$ is the number of possible outcomes (5 for multinomial, 2 for binary) and $p_{ij}$ is the probability value in the $j$th bin of the $i$th moment. This formula was taken from [30].

4. Our formula — this was developed by introducing the following boundary conditions:

$$S = 0, \qquad \omega = \infty \tag{4.12}$$

$$S = -\log(\frac{1}{5}), \qquad \omega = 0, \tag{4.13}$$

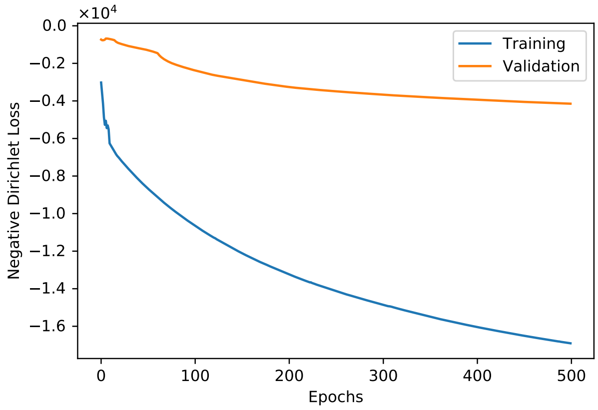
where $S$ is the Shannon entropy of the cluster. The first condition is based on the fact that $S = 0$ implies every moment belonging to that cluster has the same rating; there is maximum certainty of the rating. The following equation satisfied these two conditions:

$$\omega_i = \frac{n-1}{(\frac{1}{b})^{-S_i} - 1} - 1, \tag{4.14}$$

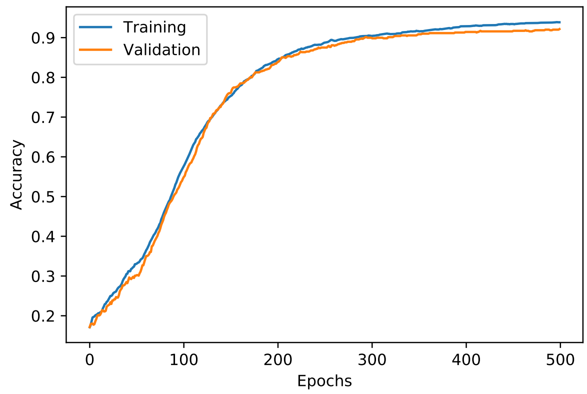
where $b$ is the logarithm base (2 in our case).

### 4.3.3 Model Training for Level Evaluation

In order to build a system which can evaluate levels, the geometry for each moment in the clusters is separated in a similar manner to that described in Section 4.2.2. However unlike the previous method, this system would output a probability distribution over the rating, rather than a single scalar value. Due to this desired output, a simple neural network with a Dirichlet output layer is trained using Tensorflow and Keras. Optimisation is carried out using Adam, and the cost function corresponds to the negative log-likelihood of the Dirichlet distribution with the inclusion of a prior. Appendix D gives this in more mathematical detail. The training and validation sets are split according to a ratio of 80:20 and shuffled according to a random seed. The performance of the model on these sets during optimisation for a typical run is visualised in Figure 4.13 - convergence to an optimum solution appears to be achieved, given that the validation accuracy exceeds 0.9 by the end of the optimisation process.




(a) Negative Dirichlet Loss.




(b) Accuracy.

Figure 4.13: Graphical visualisation of optimisation during regression model training for the PRE method.

## 4.4 Heat Map Generation

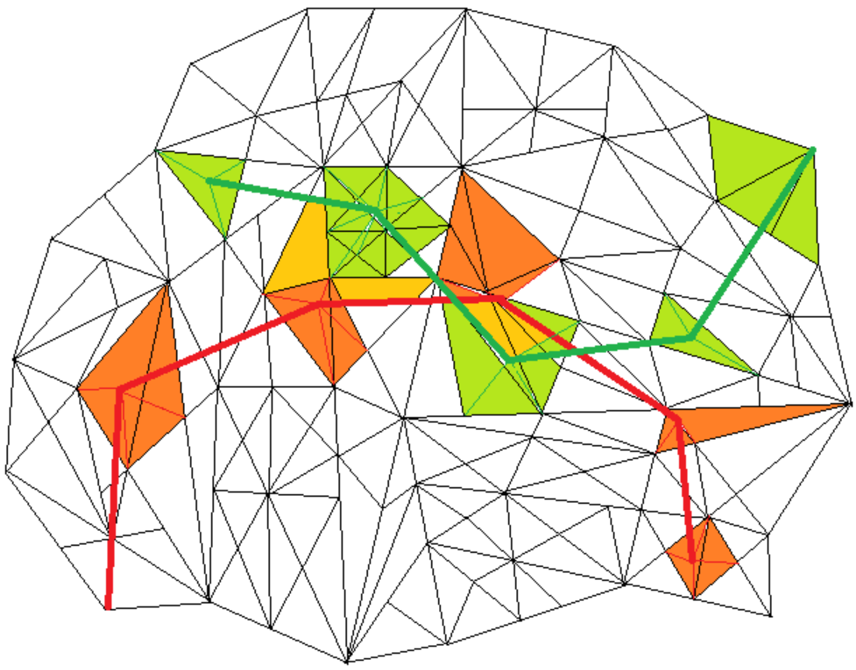


Figure 4.14: Illustration of heat map construction for map evaluation: two random paths have been plotted on a mesh, with ray casting at each point hitting the closest vertices, and the colouring of the triangles corresponding to the predicted feedback of those paths. Green represents positive feedback, red negative, with yellow indicating an even mixture of the two.

After training a model which can take level geometry as input and output a measure of the rating, we now obtain the fundamental tool for evaluating levels. This evaluation is presented visually as a "heat map of enjoyment" — regions of the level are colour-coded according to the outputted ratings given to them, via the input of these regions' geometry into the trained model. The creation of such heat maps first involves plotting many random paths (we use $10^4$) throughout the level, before computing the geometry for each path via ray casting (Figure 4.14). In order to ensure that these paths reflect actual player movement as accurately as possible, a Markov model is trained using a method loosely inspired by Wang, Yang and Shi [67]. The points along these random paths are taken from the position cluster centres derived in Section 4.1 — the number of transitions between clusters is then computed and stored in a matrix, which is finally converted to a transition probability matrix through normalisation of rows:

$$\mathbf{M_T} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N_{mc}} \\ p_{21} & p_{22} & \cdots & p_{2N_{mc}} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N_{mc}1} & p_{N_{mc}2} & \cdots & p_{N_{mc}N_{mc}} \end{pmatrix}, \qquad p_{ij} = \frac{1}{\sum_{j=1}^{N_{mc}} T_{ij}} T_{ij}, \tag{4.15}$$

where $T_{ij}$ is the number of times in the playthrough data that a transition takes place from position cluster $i$ to $j$. This helps to avoid plotting paths through non-traversable areas of the map. The geometry for each path is then inputted into the trained model which outputs a score. This score is then assigned to every vertex which is within a certain distance of the path. Once all scores from all paths have been assigned, the mean score for each vertex is calculated and then converted into an RGB colour. Figure 4.15 features an example of a heat map generated this way.

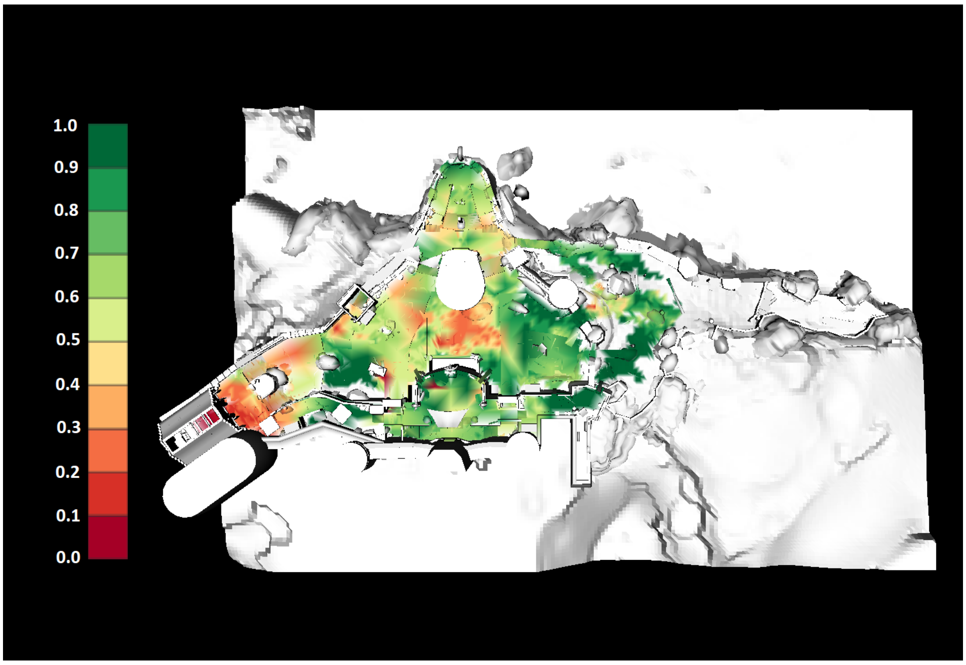



Figure 4.15: An example of a rendered heat map, where the vertices have been coloured according to the predicted normalised scores of the random paths plotted within their vicinity.

## 4.5 Algorithm Performance

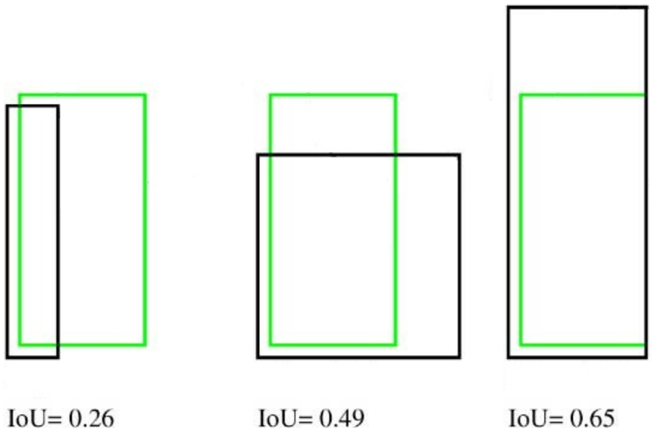



Figure 4.16: Illustration of Jaccard index for bounding boxes in computer vision, taken from [49]. An index of greater than 0.5 starts to indicate strong agreement.

In order to assess the performance of the evaluation tool, the algorithmic heat maps are compared to those produced by the human participants in Section 3.2.2. Quantification of this comparison, the heat map accuracy (HMA), is achieved through the calculation of a metric related to the Jaccard index. This is defined as the Intersection over Union (IoU) and, in the context of computer vision, is a measurement of the area of overlap between two bounding boxes (see Figure 4.16). In the context of our project, image masking is used to compute the HMA. For both types of heat map, the good, bad and neutral areas are isolated separately and their corresponding overlaps are calculated before combining them into one final accuracy score. However our system outputs a continuous heat map; we require a method of introducing thresholds for the boundaries between good, bad and neutral on the score scale in Figure 4.15. Additionally we assume that every user will have their own individual thresholds for these boundaries. For these reasons, cumulative distribution functions (CDFs) are introduced. The scores for each vertex are sorted before putting them through a CDF transform (Figure 4.17). Then they are put through the inverse CDF transform of the user map (Figure 4.18). Since the CDFs of each user will always differ (see Figure 4.19), HMA calculations are carried out on a per-user basis. As an example, Figure 4.20 shows the image masks for the heat map featured in Figure 4.19d.

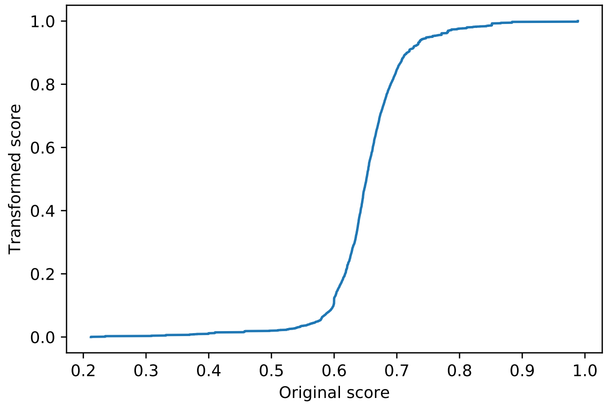



Figure 4.17: The CDF of an algorithmic heat map.

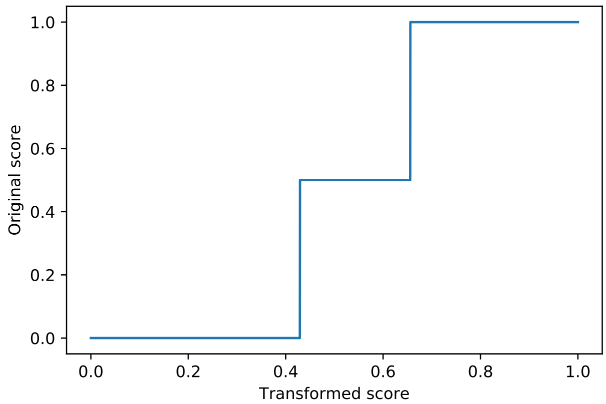
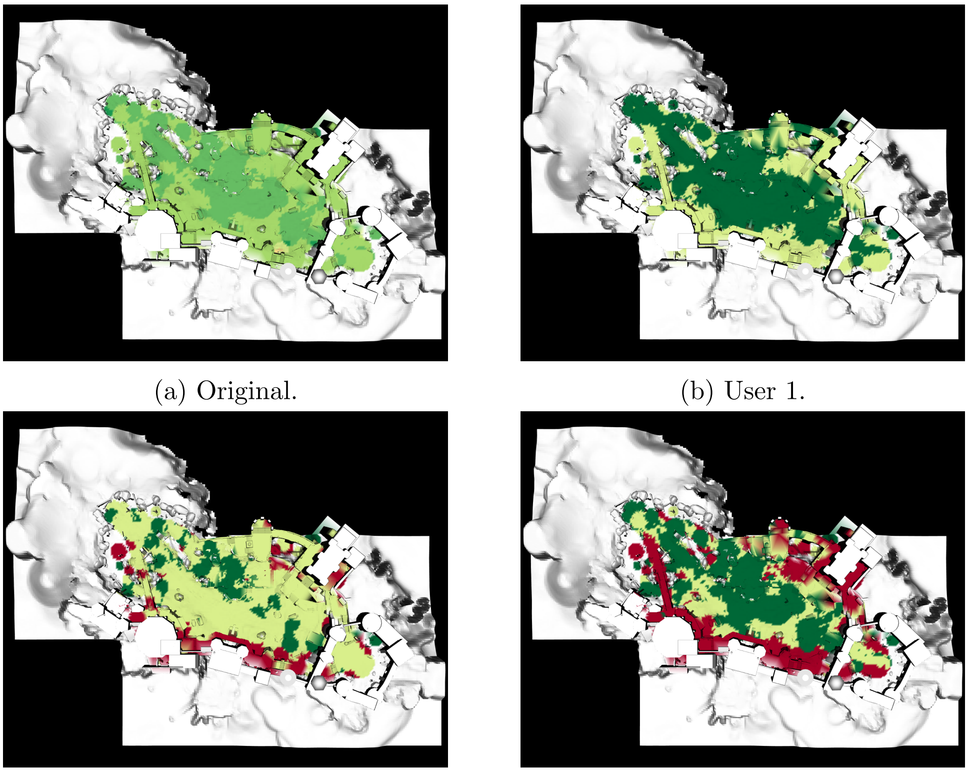



Figure 4.18: The inverse CDF of a user map.

Calculation of the overlap for a specific area is performed on a per-pixel basis:

$$HMA = \frac{P_{good} + P_{bad} + P_{neutral}}{P_{full}}, \tag{4.16}$$

where $P_{good}$, $P_{bad}$ and $P_{neutral}$ are the number of matching pixels between the algorithmic and user maps for the good, bad and neutral regions, respectively. $P_{full}$ is the number of pixels in the full mask (the traversable region of the map).



(a) Original.

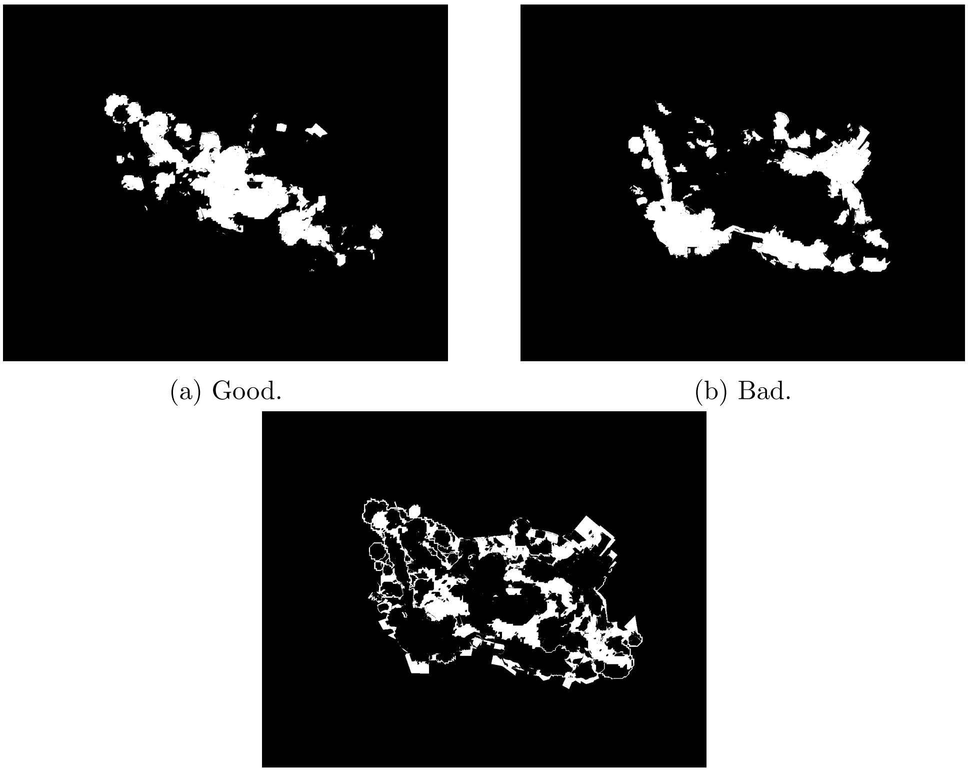
(b) User 1.

(c) User 8.

(d) User 10.

Figure 4.19: An example of an algorithmic heat map on Citadel Gate, and the resultant heat map from transforming it via the CDF of a given user's map.

Since in practice our system is intended to be applied to maps which are still under development, its performance must be assessed by testing it on "unseen" maps. Therefore for both map evaluation and playthrough predictions, the system is trained using the data from five maps and tested on data from the sixth — this is carried out across all six maps used in the project.



(a) Good.

(b) Bad.

(c) Neutral.

Figure 4.20: Image masks for the heat map in Figure 4.19d.

# Chapter 5

# Results and Discussion

The pipeline is dependent on the configuration of hyperparameters associated with feature representation, namely the window sizes into which the chunks are split as well as the number of clusters used in the two steps ($w,\ w'',\ w''',\ \kappa_1,\ \kappa_2$). Additionally there are multiple stages within the pipeline where data is shuffled according to a seed value. Therefore for a given configuration, the pipeline is run multiple times and the results are loaded into histograms. The average values of the histograms for heat map accuracy (HMA) are displayed in tables which can be found in Appendix F. Searching for the optimal set of results is carried out by first observing the "Mean" column to find the user with the highest "worst" score among all of the maps — this is known as the "best-matched user" (BMU). Then the median for each of these user's sets across all configurations is computed, and the highest is selected. The optimum configuration when using weakly supervised learning (WSL) is (2, 20, 50, 22, 7) and that for using a probabilistic regression ensemble (PRE) is (6, 16, 50, 12, 3). The results featured and discussed within this chapter are be those produced under these configurations, unless stated otherwise.

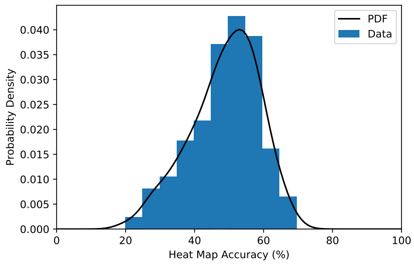
## 5.1 Evaluating Maps

In order to determine which method of capturing geometry around the player would be used, a set of HMAs is computed for random configurations of these hyperparameters using the four methods described in Section 4.1.2. These can be found in Appendix E and they indicate that using the intersection distance gives rise to high accuracies across all six maps for at least one of the users. Therefore this is the method of local geometry capture used to produce our main results.

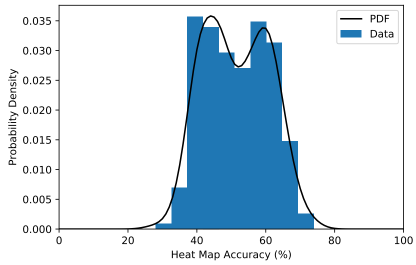
Figures 5.1 and 5.2 feature the histograms of HMAs for all users under the optima of WSL and PRE respectively. The wide range of values is due to the fact that the algorithmic maps produced by the system are a prediction of the average player feedback, since it has been trained using the in-game activity of hundreds of players. The user maps are a reflection of that individual user's feedback, therefore discrepancies between the algorithmic and user maps are to be expected. Some users are better representatives of the average For Honor player than others — this becomes clearer when we only display the results for the BMU and the worst-matched user (WMU), featured in Figures 5.3 and 5.5 for WSL, and Figures 5.4 and Figure 5.6 for PRE. We can see that in most cases, the distribution of values is greater in the BMU and WMU histograms for WSL compared to PRE. This is most likely due to the increased source of noise introduced by WSL, since this process is influenced by a random seed in the genetic algorithm in addition to that of the neural network. The only notable exception to this is Sanctuary Bridge, where the variance appears to be equal across both methods. However this may be attributed to Sanctuary Bridge's significantly small size compared to the other maps. It is important to note that the BMU and WMU are the same for both methods — these are User 6 and User 10 respectively. This is easier to observe in Figure 5.7, which features scatter plots of accuracies across all users. In fact, the ranking of users in order of lowest accuracy value is almost identical in both methods, with the exception of two users. Focusing on the BMU, their accuracies score greater than 50% across all maps for both methods, indicating good agreement between their feedback and that predicted by our system. However PRE gives rise to a smaller range of values, with a slightly higher average.

Figure 5.7b indicates that the WMU can be considered an outlier in comparison to the other users, given the distribution of their values. This is further exemplified when we look at the qualitative feedback across all users (Appendix B). Most users reacted positively to the points where minions battle, due to the intense gameplay, open space and the ability to easily "farm up" their rank. By contrast the WMU saw these areas as "clickfests", with unreasonable amounts of space. Given their low values, combined with their qualitative feedback, it is plausible that the WMU represents a minority opinion among the For Honor community — a dislike of intense gameplay, preferring locations which are more isolated from combat. This may explain why they coloured most of Sanctuary Bridge in red, since the map's smaller size compresses the area of combat and concentrates the intense gameplay.

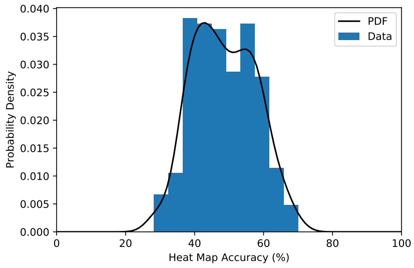
Both scatter plots indicate that there is no strong correlation between map evaluation accuracy and the field in which a given user specialises. The BMU is the only artist within the set of users, however data from other artists would need to be gathered before drawing any links between this and the feedback of general players.



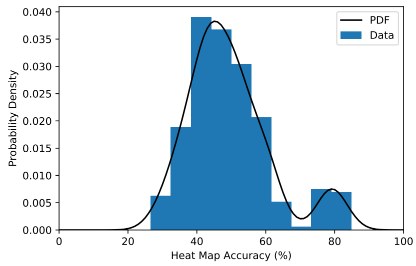

(a) Citadel Gate.



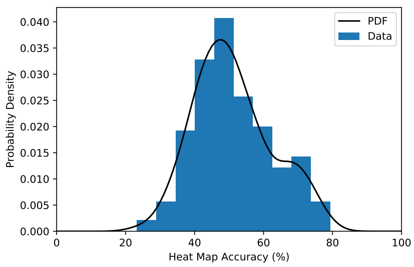

(b) Overwatch.



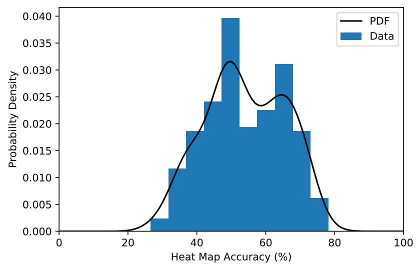

(c) Sanctuary Bridge.




(d) River Fort.

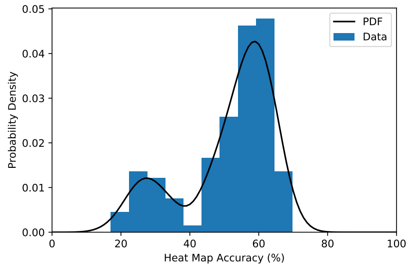



(e) High Fort.

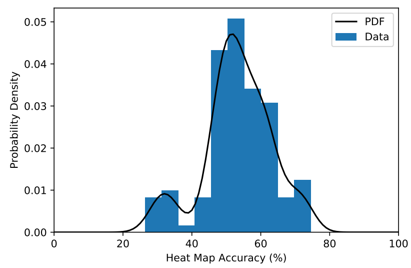



(f) The Shard.

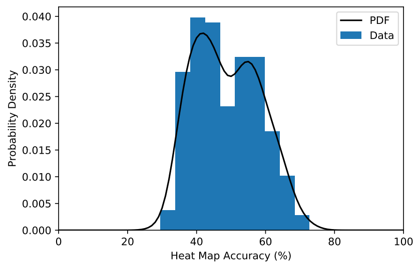
Figure 5.1: Histograms of accuracies, expressed as percentages, for each map across all users (WSL).



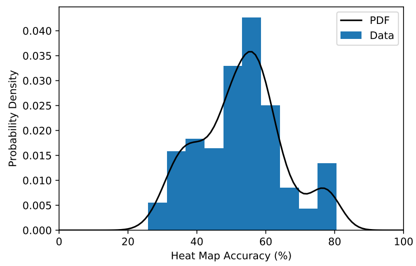

(a) Citadel Gate.



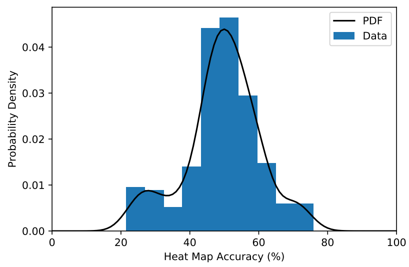

(b) Overwatch.



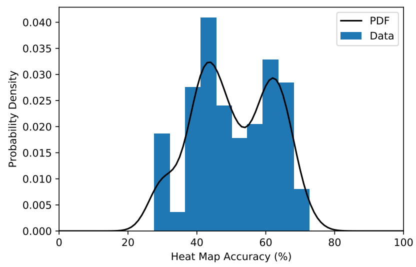

(c) Sanctuary Bridge.




(d) Riverfort.

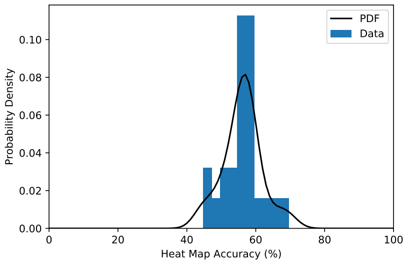



(e) Highfort.

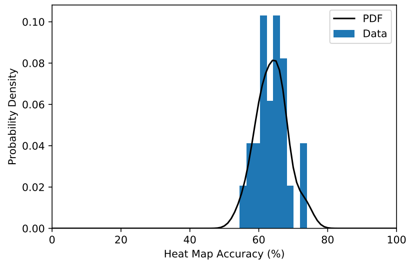



(f) The Shard.

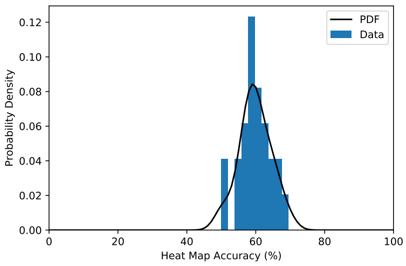
Figure 5.2: Histograms of accuracies, expressed as percentages, for each map across all users (PRE).



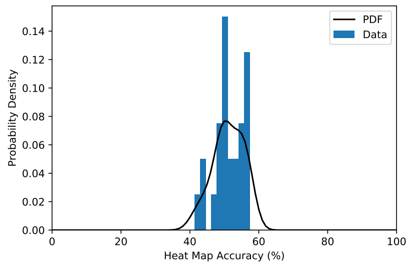

(a) Citadel Gate.



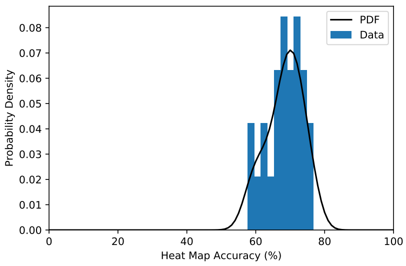

(b) Overwatch.



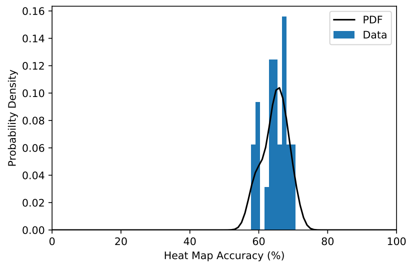

(c) Sanctuary Bridge.




(d) River Fort.

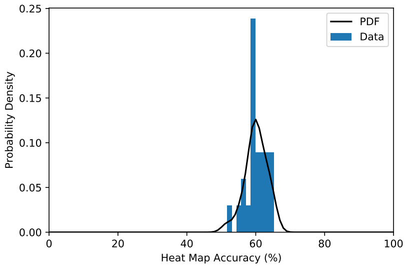



(e) High Fort.

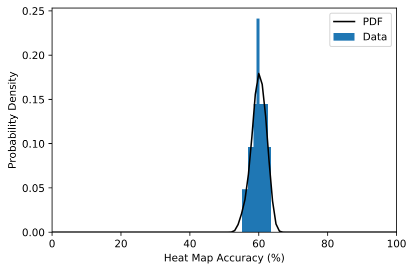



(f) The Shard.

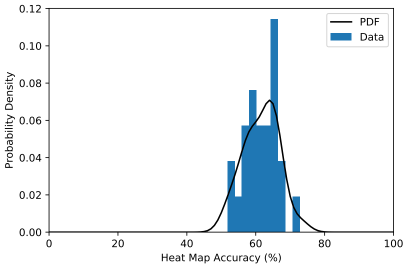
Figure 5.3: Histograms of accuracies, expressed as percentages, for the BMU across all maps (WSL).



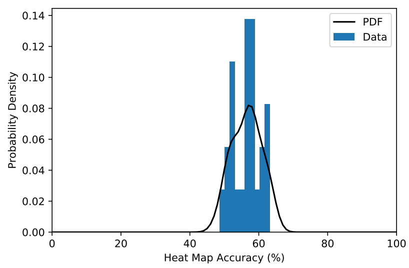

(a) Citadel Gate.



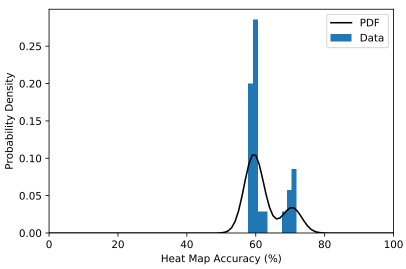

(b) Overwatch.



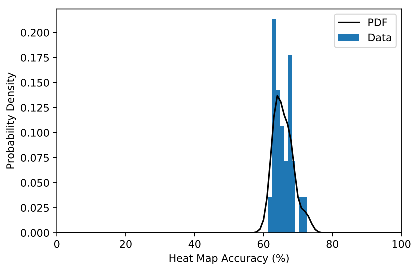

(c) Sanctuary Bridge.




(d) Riverfort.

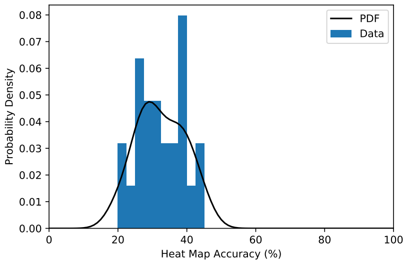



(e) Highfort.

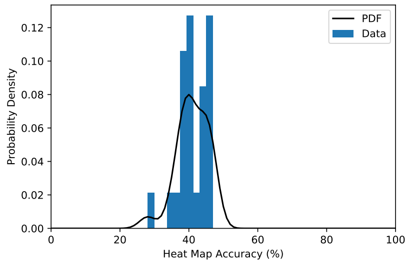



(f) The Shard.

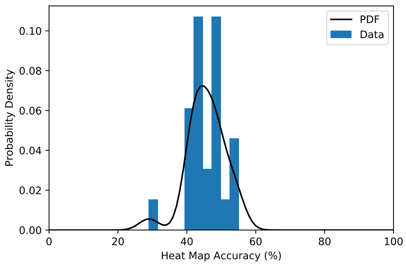
Figure 5.4: Histograms of accuracies, expressed as percentages, for the BMU across all maps (PRE).



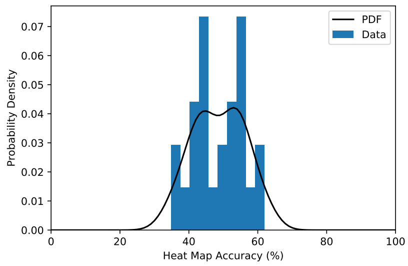

(a) Citadel Gate.



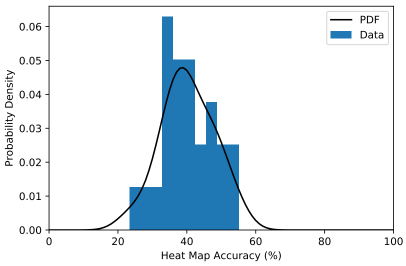

(b) Overwatch.



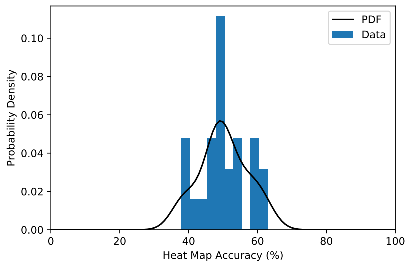

(c) Sanctuary Bridge.




(d) River Fort.

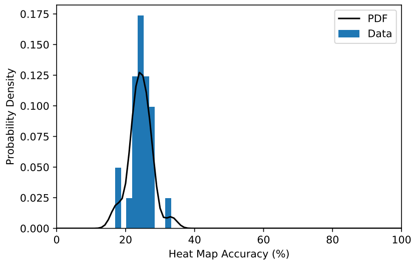



(e) High Fort.

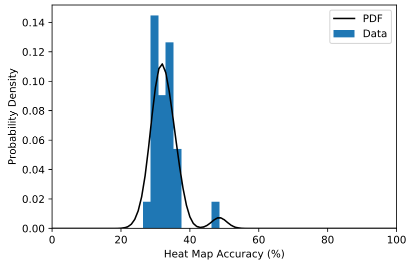



(f) The Shard.

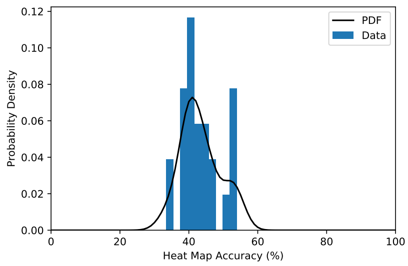
Figure 5.5: Histograms of accuracies, expressed as percentages, for the WMU across all maps (WSL).



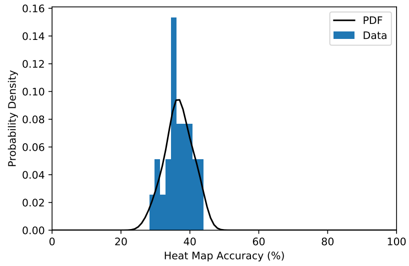

(a) Citadel Gate.



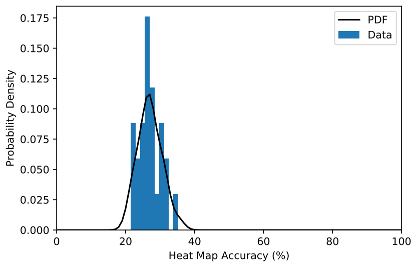

(b) Overwatch.



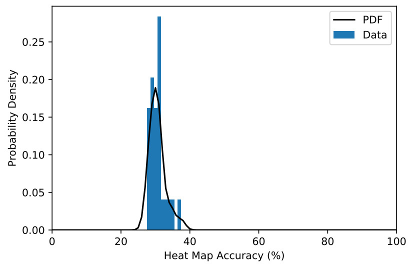

(c) Sanctuary Bridge.




(d) Riverfort.

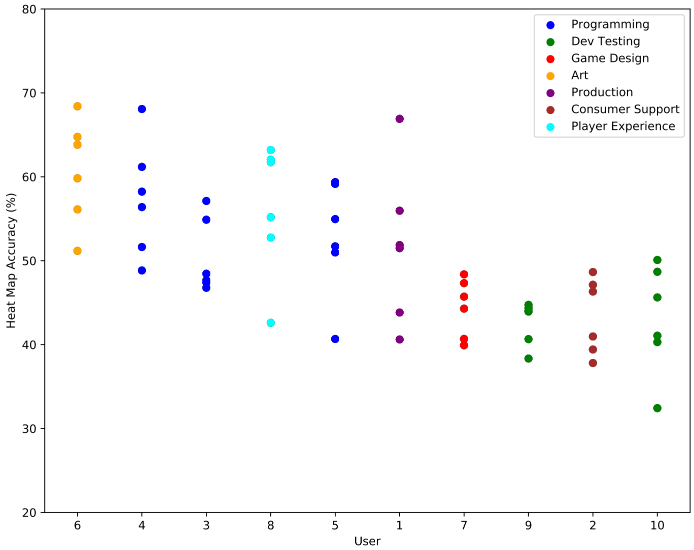



(e) Highfort.

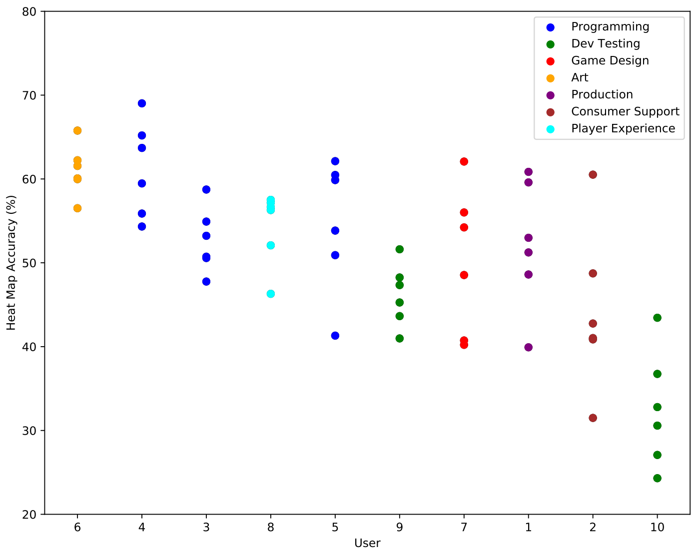



(f) The Shard.

Figure 5.6: Histograms of accuracies, expressed as percentages, for the WMU across all maps (PRE).




Method 1.




Method 2.

Figure 5.7: Scatter plots of heat map accuracies for all users and their maps under the optimal hyperparameter configurations, where the users have been ordered by their lowest accuracy, and colour coded according to their role within the industry.

## 5.2 Predicting Playthrough Feedback

With regards to predicting the ratings of playthroughs, the accuracy of the predictions would have to be compared to the baseline. Table 5.1 features the baseline accuracy for the playthrough training and test sets for all six maps under evaluation. This is found by computing the percentage of playthroughs within each set which are rated 5 stars — the metrics must be able to perform better than simply guessing a blanket rating for all playthroughs.

Note: In this context, the training set refers to the set of playthroughs associated with the five maps involved in training the system. The test set refers to the set of "unseen" playthroughs i.e. those associated with the sixth map not used in training.

| Test Map | Baseline Accuracy (%) | |
|---|---|---|
| | Training | Test |
| **Citadel Gate** | 58.1 | 52.8 |
| **Overwatch** | 54.9 | 59.5 |
| **Sanctuary Bridge** | 53.2 | 66.7 |
| **River Fort** | 55.0 | 60.3 |
| **High Fort** | 54.3 | 72.1 |
| **The Shard** | 57.2 | 46.1 |

Table 5.1: Baseline accuracies for training and testing playthrough sets.

Tables 5.2 and 5.3 respectively show the training and test accuracies of the four metrics discussed in Section 4.2.3. They clearly show that Metric 4 (using the single most extreme rating in a playthrough) gives the most accurate results. However none of the metrics are able to produce an accuracy meeting the baseline across any of the maps, let alone surpassing it. Also Metric 3 (weighted average) produces the exact same results as Metric 1 (mean) in all cases, indicating that using WSL produces a model which will only predict more extreme values from given moments. This makes sense, as the corresponding ratings for the moments outputted using WSL are either 0 or 1.

| Test Map | Test Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 48.1 | 48.3 | 48.1 | 48.9 |
| **Overwatch** | 36.7 | 37.8 | 36.7 | 40.3 |
| **Sanctuary Bridge** | 34.6 | 35.7 | 34.6 | 39.2 |
| **Riverfort** | 39.6 | 40.4 | 39.6 | 42.8 |
| **Highfort** | 30.6 | 32.3 | 30.6 | 35.9 |
| **The Shard** | 45.6 | 46.6 | 45.6 | 49.8 |

Table 5.2: Training accuracies for predicting playthroughs using WSL.

| Test Map | Test Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 41.2 | 41.2 | 41.2 | 41.7 |
| **Overwatch** | 36.1 | 38.7 | 36.1 | 43.7 |
| **Sanctuary Bridge** | 43.2 | 44.2 | 43.2 | 46.9 |
| **Riverfort** | 36.6 | 37.1 | 36.6 | 41.6 |
| **Highfort** | 36.8 | 38.0 | 36.8 | 40.1 |
| **The Shard** | 34.5 | 34.6 | 34.5 | 39.0 |

Table 5.3: Test accuracies for predicting playthroughs using WSL.

Tables 5.4 and 5.5 respectively feature the training and test prediction accuracies of the four metrics discussed in Section 4.3.2 for a multinomial model. The training accuracies achieved for metrics 1–3 across all maps either meet or slightly exceed the baseline, while metric 4 falls slightly below the baseline. All of the metrics display poor performance when looking at their test accuracies which fall far below the baseline. When switching to a binary model (see Tables 5.6 and 5.7) the accuracies display some improvement, with all of the metrics exceeding the training baseline. However despite the increase in performance, their test accuracies fail to meet the respective baselines for all maps except The Shard. Ultimately neither method produces a playthrough feedback predictor which is able to significantly exceed its respective baseline, therefore our system is not effective for this purpose.

| Test Map | Training Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 58.3 | 58.3 | 58.3 | 56.6 |
| **Overwatch** | 55.0 | 55.0 | 55.9 | 54.4 |
| **Sanctuary Bridge** | 53.8 | 53.8 | 53.8 | 52.9 |
| **Riverfort** | 55.3 | 55.3 | 55.3 | 54.0 |
| **Highfort** | 54.6 | 54.3 | 54.6 | 53.6 |
| **The Shard** | 57.9 | 57.4 | 57.9 | 56.7 |

Table 5.4: Training accuracies for predicting playthroughs using PRE via a multinomial model.

| Test Map | Test Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 23.2 | 23.5 | 23.2 | 21.4 |
| **Overwatch** | 24.2 | 24.8 | 24.2 | 17.0 |
| **Sanctuary Bridge** | 30.1 | 30.1 | 30.1 | 24.4 |
| **Riverfort** | 55.6 | 58.7 | 55.6 | 48.9 |
| **Highfort** | 50.0 | 48.5 | 50.0 | 36.8 |
| **The Shard** | 30.5 | 30.5 | 30.5 | 24.8 |

Table 5.5: Test accuracies for predicting playthroughs using PRE via a multinomial model.

| Test Map | Training Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 62.5 | 62.8 | 62.3 | 60.1 |
| **Overwatch** | 59.4 | 60.6 | 59.4 | 58.4 |
| **Sanctuary Bridge** | 58.8 | 58.0 | 58.8 | 57.8 |
| **Riverfort** | 59.1 | 59.4 | 63.1 | 59.7 |
| **Highfort** | 60.9 | 59.7 | 60.9 | 59.0 |
| **The Shard** | 60.1 | 59.8 | 60.2 | 58.8 |

Table 5.6: Training accuracies for predicting playthroughs using PRE via a binary model.

| Test Map | Test Accuracy (%) | | | |
|---|---|---|---|---|
| | Metric 1 | Metric 2 | Metric 3 | Metric 4 |
| **Citadel Gate** | 47.7 | 48.2 | 47.7 | 46.4 |
| **Overwatch** | 43.8 | 41.8 | 43.8 | 39.2 |
| **Sanctuary Bridge** | 42.3 | 41.7 | 42.3 | 39.1 |
| **Riverfort** | 54.0 | 54.0 | 54.0 | 46.0 |
| **Highfort** | 39.7 | 41.2 | 39.7 | 39.7 |
| **The Shard** | 58.9 | 57.4 | 58.9 | 51.1 |

Table 5.7: Test accuracies for predicting playthroughs using PRE via a binary model.

# Chapter 6

# Conclusion

The main goal of this research was to create a system which, given a game level as input, can predict the feedback the level will elicit via the application of machine learning to gameplay, geometry and feedback data. We have achieved this by visualising the feedback in the form of heat maps of enjoyment, whose accuracies have been determined via comparison to coloured maps produced from user study participants. Two distinct methods were used to train the system — one employing WSL, and the other involving a PRE. Both of the methods have produced heat maps which are in good agreement with users, however using WSL led to outputs which were more susceptible to noise and took significantly longer to run. Also this method suffered from overfitting during the training process. Therefore PRE is the preferred method to use. The features extracted from the data were also used to train a system which can predict the ratings of playthroughs, however neither method was able to produce an effective predictor.

## 6.1 Limitations

Since it requires existing telemetry data for training, the presented system may be more suited for DLCs e.g. developing additional maps for a title which has already been released, and using gameplay from this title as training data. Sequels may also be an option, as they generally contain the same gameplay features as their predecessors. However one should not completely dismiss the system as a tool for designers working on a completely new title — games within the same genre will share a vast range of common features, regardless of whether they have been developed by different teams.

Searching for an appropriate data set was one of the most difficult aspects of the

project. Training the system is heavily dependent on the features which are available for extraction. For example the relatively low sampling rate of 1 position every 3 seconds constrains the size of the chunks into which the playthroughs could be split for carrying out PCA. The lack of regularly sampled player positions was the reason that data sets from other games were rejected for use in the project. However this limitation is mainly associated with the telemetry and storage capabilities of game development studios.
Feedback data is also essential for training the system, however this does not necessarily have to be limited to a 5-star rating — a binary like/dislike system could also be used. Additionally if there are no explicit feedback metrics available, one could infer a "rating-by-proxy". For example quitting in the middle of a match could be interpreted as negative feedback. Alternatively if a game's multiplayer modes rotate between random maps in the lobby with the ability to vote to skip, a map's unpopularity may be reflected in the number of times it has been skipped.

There were limitations to the manner in which the heat map accuracies (HMAs) were calculated. The algorithmic maps were produced by automatically colouring the vertices of the map, whereas the users were required to colour a top-down image of the map. This meant that comparisons between the two could only be achieved through pixel matching; this method cannot take into account areas with multiple levels e.g. bridges, walkways and multi-storey buildings. One way of resolving this would have been to ask participants to navigate the 3-D map and colour the regions using graphics software e.g. Blender. However participants may have found this process tedious and awkward; it may also have hindered or at least slowed down gathering of detailed user feedback.

## 6.2 Extensions

Since the system has been trained using data collected from hundreds of playthroughs, the heat maps it produces are a prediction of the average feedback across a vast number of players. However one way of improving the system could be to take into account different types of players i.e. for each game map, produced multiple heat maps, each displaying the predicted feedback of a specific type of player. This could be achieved by collecting data from the players' profiles such as:

- Total play time.
- Current level for each hero type.
- Game count for every game mode.

- Percentage completion of the campaign or trials.
- Total kill/death and win/loss counts.

This information could be fed into clustering algorithms to derive specific player types (similar to the work of Bauckhage et al. [5], Melhart et al. [41] and Ferguson et al. [15]) and combined with the existing system to diversify its feedback. Augmenting the system with this capability would be useful for a designer aiming to tailor their map to a particular subset of their audience.

In Figure 1.1 it was illustrated that player experience modelling (PEM) and content quality evaluation comprise half of the experience-driven procedural content generation (EDPCG) framework. A natural extension of the project would be to complete the framework by combining the system with PCG algorithms, resulting in a tool which can automatically generate multiple levels in which the designer will be able observe the predicted player reception.

## 6.3 Impact

While the system has not been integrated into a level designer toolbox or used in the development of any games at the time of writing, the project has attracted a great deal of interest from other teams across various studios. For example after presenting at UDS 2021 , the leader of another team was interested to see a research project other than theirs which placed an emphasis on geometry. Additionally at the La Forge Open House 2021, there was discussion of potential collaboration with another team working on a PCG tool — we discussed the idea of using our level evaluation tool to filter out unwanted maps amongst a population of candidates produced by their system. This links to the completion of the EDPCG framework mentioned in the previous section, as observing the predicted feedback in the candidates may help designers choose the candidate they wish to be the final design. Overall this shows that there is enormous potential for our system to be utilised in industry, fulfilling the mission statement of the CDE.

## 6.4 Outlook

We have presented a method for building a tool which can potentially provide designers with invaluable insight into how their creations will be received by their target audiences.

The presented system is meant to assist the designer without taking away their control or compromising their workflow. At the beginning of this thesis, the evolution of computer games was described in terms of graphics, play time and required manpower. The introduction of automation in the development lifecycle was also discussed, and the combination of our system with PCG software would be an extremely important tool for the game industry as a whole. However a more significant area undergoing rapid growth in recent years is data collection. Data is underpinning more and more aspects of our society, and harnessing the data of players fuels systems such as the one presented in this thesis. Our results and conclusions would be non-existent without the ability to collect, process and analyse data. As companies are able to gather a wider variety of player data in more efficient ways, PEM and level evaluation tools will continue to improve, and designers will be able to deliver truly personalised gaming experiences.

# Bibliography


[1] Abbott, T., 2010. *Mda framework - unconnected connectivity* [Online]. Gamasutra. Available from: `http://www.gamasutra.com/blogs/TuckerAbbott/20101212/88611/MDA_Framework_Unconnected_Connectivity.php` [Accessed 26/04/2018].

[2] Alexe, B., Deselaers, T. and Ferrari, V., 2010. What is an object? *2010 ieee computer society conference on computer vision and pattern recognition*, 13-18 June 2010 San Fransisco. IEEE, pp.73–80.

[3] Amores, J., 2013. Multiple instance classification: Review, taxonomy and comparative study. *Artificial intelligence*, 201, pp.81–105.

[4] Anon., 2014. Introduction to level design (internal ubisoft document).

[5] Bauckhage, C., Drachen, A. and Sifa, R., 2015. Clustering game behaviour data. *Ieee transactions on computational intelligence and ai in games*, 7(3), pp.266–278.

[6] Bellman, R., 1957. *Dynamic programming.* Princeton: Princeton University Press.

[7] Björk, S. and Holopainen, J., 2004. *Patterns in game design.* USA: Charles River Media Inc.

[8] Caicendo, J.C., C., M., Goodman, A., Singh, S. and Carpenter, A.E., 2018. Weakly supervised learning of single-cell feature embeddings. *The ieee conference on computer vision and pattern recognition (cvpr)*, 18-22 June 2018 Salt Lake City. IEEE, pp.9309–9318.

[9] Cao, L., Liu, Z. and Huang, T., 2010. Cross-dataset action detection. *Proceedings of the ieee computer society conference on computer vision and pattern recognition*, 13-18 June San Francisco, California. IEEE, pp.1998–2005.

[10] CDE, 2021. *Centre for digital entertainment official website* [Online]. Available from: http://www.digital-entertainment.org/ [Accessed 12/07/2021].

[11] Csikszentmihalyi, M., 1990. *Flow: The psychology of optimal experience*. New York: Harper Perennial.

[12] Csurka, G., Dance, C.R., Fan, L., Willamowski, J. and Bray, C., 2004. Visual categorization with bags of keypoints. *In workshop on statistical learning in computer vision, eccv*. pp.1–22.

[13] Egenfeldt-Nielsen, S., Smith, J.H. and Tosca, S.P., 2008. *Understanding video games: The essential introduction*. New York: Taylor and Francis.

[14] Fenlon, W., 2015. *For honor: light strategy and heavy swords in a medieval dueler* [Online]. Available from: https://www.pcgamer.com/for-honor-light-strategy-and-heavy-swords-in-a-medieval-dueler/ [Accessed 21/10/2020].

[15] Ferguson, M., Devlin, S., Kudenko, D. and Walker, J.A., 2020. Player style clustering without game variables. *Proceedings of the international conference on the foundations of digital games (fdg) 2020*, 15-18 September Bugibba Malta. ACM, pp.1–4.

[16] GamesTM, 2017. *The evolution of motion capture* [Online]. GamesTM. Available from: https://www.gamestm.co.uk/features/the-evolution-of-motion-capture/ [Accessed 02/01/2017].

[17] Glenski, M., Stoddard, G., Resnick, P. and Weninger, T., 2018. Guessthekarma: A game to assess social rating systems. *Proceedings of the acm on human-computer interaction*, 2(CSCW), pp.59:1–15.

[18] Green, M.C., Brock, T.C. and Kaufmann, G.F., 2004. Understanding media enjoyment: The role of transportation into narrative worlds. *Communication theory*, 14(4), pp.311–327.

[19] Guckelsberger, C., Salge, C., Gow, J. and Cairns, P., 2017. Predicting player experience without the player.: An exploratory study. *Proceedings of the annual symposium on computer-human interaction in play*, 15-18 October Amsterdam. ACM, pp.305–315.

[20] Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24, pp.417–441.

[21] Hunicke, R., LeBlanc, M. and Zubek, R., 2004. Mda: A formal approach to game design and game research. *Proceedings of the challenges in games ai workshop, nineteenth national conference of artificial intelligence*, 25–26 July 2004 California. California: Press, pp.1–5.

[22] Iida, H., Takeshita, N. and Yoshimura, J., 2002. A metric for entertainment of boardgames: it's implication for evolution of chess variants. In: R. Nakatsu and J. Hoshino, eds. *Ifip first international workshop on entertainment computing*, 14–17 May 2002, Makuhari. London: Kluwer Academic, pp.65–72.

[23] IndigoEX, 2017. *For honor map overviews* [Online]. Available from: `https://imgur.com/gallery/J9LuU/comment/962465585` [Accessed 20/02/2017].

[24] Ize, T., Wald, I. and Parker, S.G., 2009. Ray tracing with the bsp tree. *2008 ieee symposium on interactive ray tracing*, 9-10 August Los Angeles. IEEE, pp.159–166.

[25] Khanna, H., 2017. *The psychology of rating systems* [Online]. Available from: `https://hackernoon.com/the-psychology-of-rating-systems-3103e26fddd8` [Accessed 05/04/2020].

[26] Koster, R., 2014. *A theory of fun for game design.* 2nd ed. Sebastopol CA: O'Reilly Media.

[27] Lankveld, G., Spronck, P. and Rauterberg, M., 2008. Difficulty scaling through incongruity. *Proceedings of the 4th international artificial intelligence and interactive digital entertainment conference*, 22–24 October 2008, California. California: Stanford University, pp.228–229.

[28] Laptev, I., 2005. On space-time interest points. *International journal of computer vision*, 64, pp.107–123.

[29] Lazzaro, N., 2004. *Why we play games: Four keys to more emotion in player experiences.* (510-658-8077). Oakland CA: XEODesign Inc.

[30] Li, C.H., 2015. *Log entropy weighting* [Online]. Available from: `https://www.youtube.com/watch?v=8K23Sjw_Hd4` [Accessed 03/10/2020].

[31] Liapis, A., 2013. *Sentient sketchbook demonstration (with captions)* [Online]. Available from: `https://www.youtube.com/watch?v=Eop1AuFcujE` [Accessed 02/04/2020].

[32] Liapis, A., Yannakakis, G.N. and Togelius, J., 2011. Towards a generic method of evaluating game levels. *7th aaai conference on artificial intelligence and interactive digital entertainment (aiide-11)*, 11-14 October 2011 California. AAAI Press, pp.30–36.

[33] Liapis, A., Yannakakis, G.N. and Togelius, J., 2013. Sentient sketchbook: computer-aided game level authoring. *Proceedings of the 8th international conference on the foundations of digital games (fdg 2013)*, 14-17 May 2013 Crete. Society for the Advancement of the Science of Digital Games, pp.213–220.

[34] Liapis, A., Yannakakis, G.N. and Togelius, J., 2014. Designer modeling for sentient sketchbook. *Ieee conference on computatonal intelligence and games*, 26-29 August 2014 Dortmund. IEEE.

[35] Luenendonk, M., 2015. *The gaming industry - an introduction* [Online]. Cleverism. Available from: `https://www.cleverism.com/gaming-industry-introduction/` [Accessed 13/04/2018].

[36] Malone, T.W., 1980. What makes things fun to learn? heuristics for designing instructional computer games. *Sigsmall '80 proceedings of the 3rd acm sigsmall symposium and the first sigpc symposium on small systems*. pp.162–169.

[37] Mandryk, R.L. and Atkins, M.S., 2007. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International journal of human-computer studies*, 65(4), pp.329–347.

[38] Mandryk, R.L., Inkpen, K.M. and Calvert, T.W., 2006. Using psychophysiological techniques to measure user experience with entertainment technologies. *Behaviour and information technology*, 25(6), pp.141–158.

[39] Martyn, C., 2019. Email exchange with senior technical designer.

[40] Masthoff, J., 2003. Modeling the multiple people that are me. *Lecture notes in artificial intelligence (subseries of lecture notes in computer science)*, 2702, pp.258–262.

[41] Melhart, D., Azadvar, A., Canossa, A., Liapis, A. and Yannakakis, G.N., 2019. Your gameplay says it all: Modelling motivation in tom clancys the division. *Ieee conference on games*, 20–23 August 2019 London. London: IEEE, pp.61–68.

[42] Nabi, R.L. and Krcmar, M., 2004. Conceptualizing media enjoyment as attitude: Implications for mass media effects research. *Communication theory*, 14(4), pp.288–310.

[43] Nesterov, Y., 2005. Smooth minimization of non-smooth functions. *Mathematical programming*, 103, pp.127–152.

[44] Pedersen, C., Togelius, J. and Yannakakis, G.N., 2009. Modeling player experience in super mario bros. *2009 ieee symposium on computational intelligence and games*, 7-10 September 2009 Milan. IEEE, pp.132–139.

[45] Pedersen, C., Togelius, J. and Yannakakis, G.N., 2009. Optimization of platform game levels for player experience. *Proceedings of the 5th artificial intelligence and interactive digital entertainment conference*, 14-16 October 2009 California. California: AAAI, pp.191–192.

[46] Pedersen, C., Togelius, J. and Yannakakis, G.N., 2010. Modeling player experience for content creation. *Ieee transactions on computational intelligence and ai in games*, 2(1), pp.54–67.

[47] Preuss, M., Liapis, A. and Togelius, J., 2014. Searching for good and diverse game levels. *Ieee conference on computatonal intelligence and games*, 26-29 August 2014 Dortmund. IEEE.

[48] Raney, A.A., 2004. Expanding disposition theory: Reconsidering character liking, moral evaluations, and enjoyment. *Communication theory*, 14(4), pp.348–369.

[49] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I. and Savarese, S., 2019. Generalized intersection over union. *The ieee conference on computer vision and pattern recognition (cvpr)*, 16-20 June Long Beach. IEEE, pp.658–666.

[50] Rigby, S. and Ryan, R.M., 2006. *The player experience of need satisfaction.* Florida: Immersyve Inc.

[51] Rigby, S. and Ryan, R.M., 2011. *Glued to games: How video games draw us in and hold us spellbound.* California: Library of Congress.

[52] Rigby, S., Ryan, R.M. and Przybylski, A., 2006. The motivational pull of video games: A self-determination theory approach. *Motivation and emotion*, 30(4), pp.344–360.

[53] Schell, J., 2008. *The art of game design.* Burlington MA: Morgan Kaufmann Publishers.

[54] Shaker, N., Togelius, J. and Nelson, M.J., 2016. *Procedural content generation in games.* Switzerland: Springer International Publishing.

[55] Siva, P., Russell, C. and Xiang, T., 2012. In defence of negative mining for annotating weakly labelled data. In: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato and C. Schmid, eds. *12th european conference on computer vision*, 7-13 October 2012 Florence. Springer, pp.594–608.

[56] Siva, P. and Xiang, T., 2011. Weakly supervised action detection. *British machine vision conference*, 2, pp.1–11.

[57] Smith, G., 2014. The future of procedural content generation in games. *Experimental ai in games workshop at aiide*, 4 October Raleigh, North Carolina. Persson, pp.53–57.

[58] Smith, P., 2016. *Going underground: How reflections tamed randomness* [Online]. Guildford: Ubisoft. Available from: `http://blog.ubi.com/en-GB/divisions-procedural-underground/` [Accessed 02/08/2018].

[59] Song, H.O., Girshick, R., Jegelka, S., Mairal, J., Harchaoui, Z. and Darrell, T., 2014. On learning to localize objects with minimal supervision. *31st international conference on machine learning, icml 2014*, 21-26 June Beijing, China. JMLR, pp.3582–3590.

[60] Sundström, P., 2005. *Exploring the affective loop.* Ph.D. thesis. Stockholm University.

[61] Sweetser, P. and Wyeth, P., 2005. Gameflow: A model for evaluating player enjoyment in games. *Computers in entertainment*, 3(3), p.3.

[62] Taylor, T., 2018. *How to take your workout outdoors: Escape to sun valley for a mountain biking adventure* [Online]. Available from: `https://www.si.com/edge/2017/05/05/`

take-your-workout-outdoors-escape-sun-valley-mountain-biking-adventure [Accessed 14/06/2018].

[63] Togelius, J., Kastbjerg, E., Schedl, D. and Yannakakis, G.N., 2011. What is procedural content generation? mario on the borderline. *Proceedings of the 2nd international workshop on procedural content generation in games*. Bordeaux: Association for Computing Machinery.

[64] Togelius, J., Yannakakis, G.N., Stanley, K. and Browne, C., 2011. Search-based procedural content generation. *Ieee transactions on computational intelligence and ai in games*, 3(3), pp.172–186.

[65] *Ubisoft reflections official website*, 2021. [Online]. Available from: https://reflections.ubisoft.com/ [Accessed 12/07/2021].

[66] Uijlings, J.R., Van De Sande, K.E., Gevers, T. and Smeulders, A.W., 2013. Selective search for object recognition. *International journal of computer vision*, 104(2), pp.154–171.

[67] Wang, H., Yang, Z. and Shi, Y., 2019. Next location prediction based on an adaboost-markov model of mobile users. *Sensors (switzerland)*, 19(6), pp.1–19.

[68] Yannakakis, G.N. and Hallam, J., 2004. Evolving opponents for interesting interactive computer games. *Proceedings of the 8th international conference on simulation of adaptive behavior*, Los Angeles. Cambridge: MIT Press, pp.499–508.

[69] Yannakakis, G.N. and Hallam, J., 2005. A generic approach for obtaining higher entertainment in predator/prey games. *Journal of game development*, 1(3), pp.23–30.

[70] Yannakakis, G.N. and Hallam, J., 2006. Towards capturing and enhancing entertainment in computer games. *Advances in artificial intelligence*, 18-20 May Crete. Springer Berlin Heidelberg, pp.432–442.

[71] Yannakakis, G.N. and Hallam, J., 2007. Feature selection for capturing the experience of fun. *Proceedings of the aiide07 workshop on optimizing player satisfaction*, 6-8 June 2007 Stanford. California: AAAI, pp.37–42.

[72] Yannakakis, G.N. and Hallam, J., 2007. Towards optimizing entertainment in computer games. *Applied artificial intelligence*, 21(10), pp.933–971.

[73] Yannakakis, G.N., Hallam, J. and Lund, H.H., 2006. Capturing entertainment through heart rate dynamics in the playware playground. In: R. Harper, M. Rauterberg and M. Combetto, eds. *Entertainment computing - icec 2006*, 20-22 September 2006 Cambridge. Springer-Verlag, pp.314–317.

[74] Yannakakis, G.N., Hallam, J. and Lund, H.H., 2006. Comparative fun analysis in the innovative playware game platform. *Proceedings of the 1st world conference for fun 'n games*, 26-28 June 2006, Preston. Preston: University of Central Lancashire, pp.64–70.

[75] Yannakakis, G.N., Lund, H.H. and Hallam, J., 2006. Modeling children's entertainment in the playware playground. In: S. Louis and G. Kendall, eds. *2006 ieee symposium on computational intelligence and games*, 22-24 May 2006 Reno. AAAI, pp.134–141.

[76] Yannakakis, G.N. and Togelius, J., 2011. Experience-driven procedural content generation. *Ieee transactions on affective computing*, 2(3), pp.147–161.

[77] Yuan, J., Liu, Z. and Wu, Y., 2011. Discriminative video pattern search for efficient action detection. *Ieee transactions on pattern analysis and machine intelligence*, 33(9), pp.1728–1743.

[78] Zhou, Z.H., 2018. A brief introduction to weakly supervised learning. *National science review*, 5(1), pp.44–53.

# Appendix A — User Study Instructions

Azeem Khan
azeem.khan@ubisoft.com

Supervisor: Tom Haines
tsfh20@bath.ac.uk

## FH QC Test Instructions

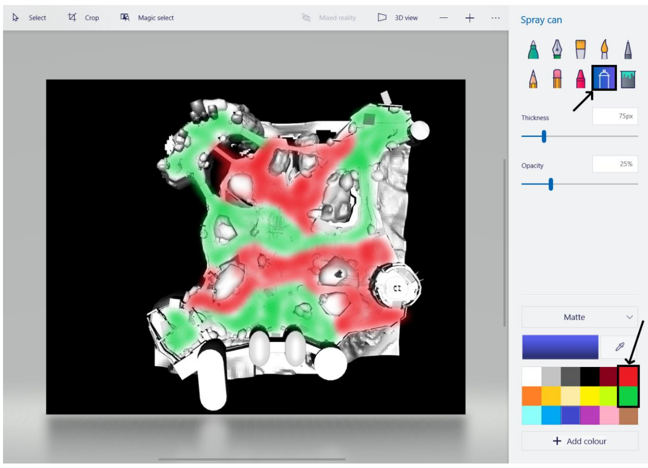
We are conducting a user study with the University of Bath to compare the performance of a system which has been trained to evaluate game levels, with the feedback of a human.

By participating in this, you give consent for your feedback to be used for the purposes of this study. You are free to quit at any time.

**For each map:**

Play several custom matches of Dominion with bots (any weather condition) until you feel confident that you understand the layout of the map. Play as any hero you wish.

Open the map's image in Paint 3D and, using the spray can, highlight in green (red) the areas in which you experienced the most (least) enjoyment. If you felt neutral about a certain area, just leave it uncoloured. In a separate text file, write a few lines explaining your colouring (e.g. "the area around B was most enjoyable because…….") Then save the image and text file and send to azeem.khan@ubisoft.com. An example is shown below:



# Appendix B — Summarised User Feedback

## Citadel Gate

### Capture Point A

Most disputed point in terms of feedback. Ramp leading up to it feels like wasted potential as it is far removed from other regions of map and rarely visited, but defenders can use it to ambush. Pillars/columns result in an enclosed space in which it is difficult to manoeuvre/dodge.

### Capture Point B

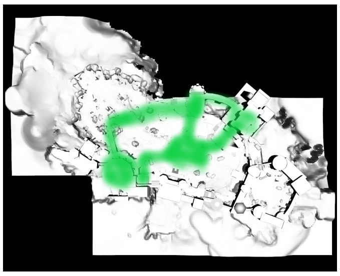
All users except one marked this green  wide open area with lots of action, and users never felt overwhelmed/swarmed by minions unlike with some other maps. One user even mentioned that there could have been more environmental dangers here to make things more exciting. The one user who marked this red was vague in their feedback — "clickfest with no real fun to it or... unreasonable amounts of space".

### Capture Point C

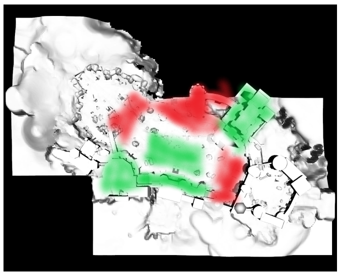
Consistently good across all users, mainly due to having plenty of space to fight e.g. circular prop in the middle can be used to put distance between you and the enemy. Also many entrances/exits. Two users complained about the ladders under this area however.

### Other

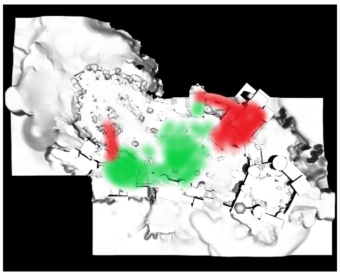
Spawn near C was noted by two users to be visually impressive during rush into battle, particularly the view of the citadel from here. Paths leading up to C and A were deemed by some users to be bad due to narrowness. Areas connecting points are empty and unused for combat — seen as boring by some, but others find this makes it easy to quickly move between the points (one user even doing a lap) and circumvent the minions in B if one does not wish to go through them to get from point to point. NOTE: User 9 marked many sections red which he considered to be the most fun, but they were not visited very often (see: wasted potential in Capture Point A section).



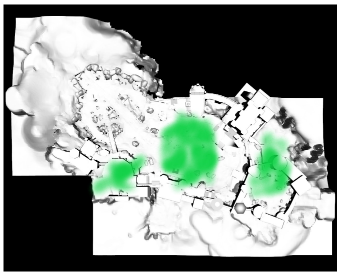
User 1.



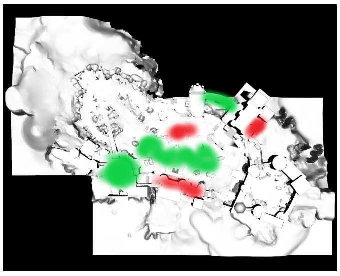
User 2.



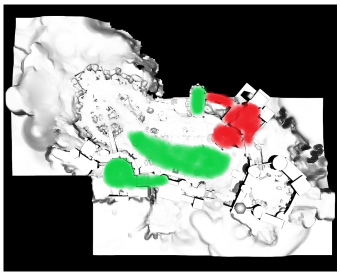
User 3.



User 4.



User 5.

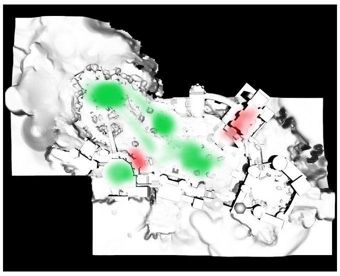


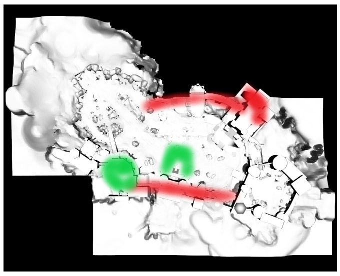
User 6 (BMU).

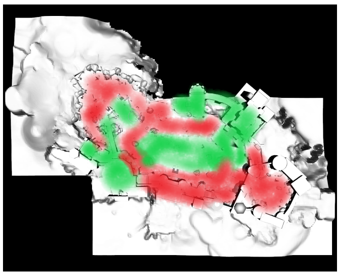
## Overwatch

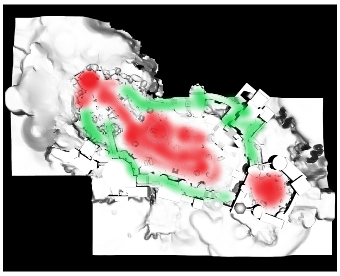
### Capture Point A

Many users liked this point due to its close connection to both bridges, large fighting area and spread out ledges. However one user said the capture point itself might be too


User 7.


User 8.


User 9.


User 10 (WMU).

large. It also appears to be rarely visited by enemies which makes it easy to capture but also quite boring, which is why some users marked it red.

### Capture Point B

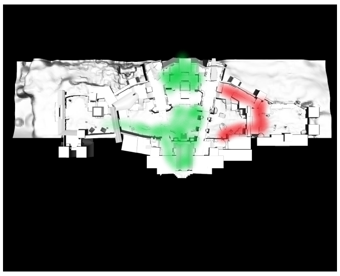
Minions are fought here, which is fun. But one can get aerial attacked by enemies from the bridge above.

### Capture Point C

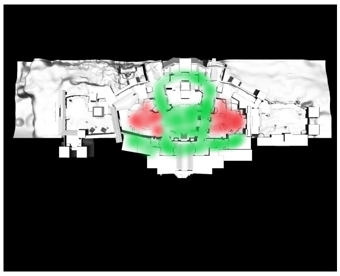
Feedback mostly positive — multiple entrances and the ability to activate floor panels which drop into spike pits makes for very intense and fun gameplay. However being on the receiving end of these traps can be extremely frustrating, as well as the narrow corridors leading up to this point.

### Other

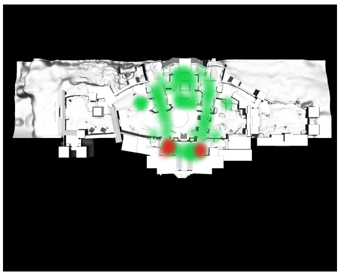
Bridges — mixed feedback — disagreement over whether bridges were wide enough or not. Opportunity for jumping down and ambushing enemies but this can damage your character and doesn't seem worth trying as most of the time there wont be an enemy directly underneath you. Also falling through holes can be annoying, but some people appreciate that these environmental hazards make the game more interesting. Underneath bridges — negative — minions end up being funnelled into two corridors and enclosed space makes it difficult to wield certain weapons. Two users reported erratic AI behaviour which dragged out the game.



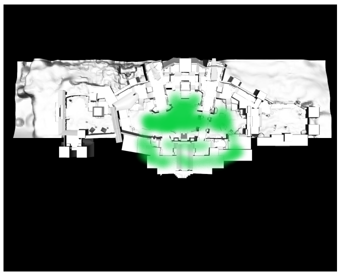
User 1.



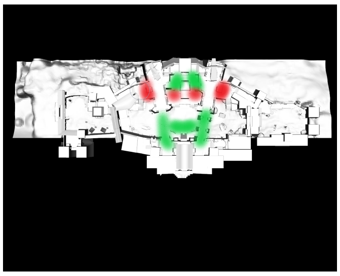
User 2.



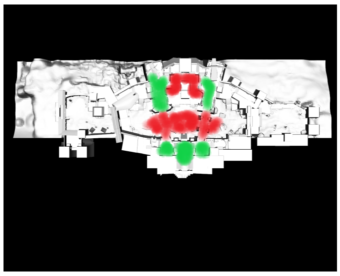
User 3.



User 4.



User 5.

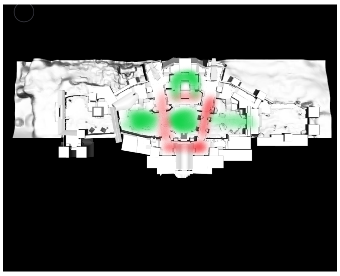


User 6 (BMU).

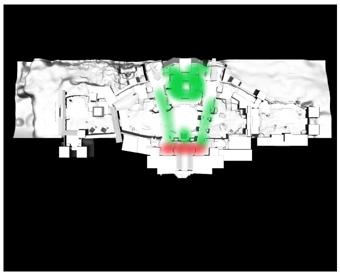
## Sanctuary Bridge

### Capture Points A & C

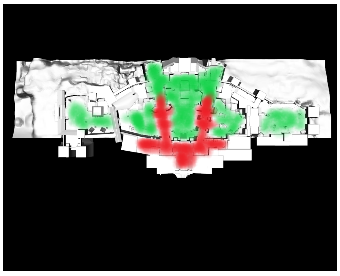
Very similar — map in general is symmetric. Isolation and proximity to their respective spawn points makes game more competitive i.e. teams compete solely for point B and every second counts. Almost all users marked these green.



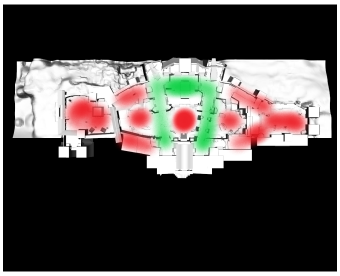
User 7.



User 8.



User 9.



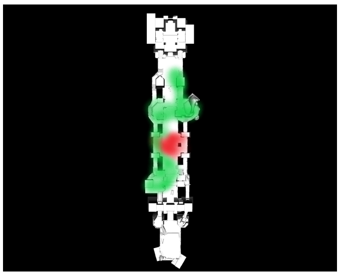
User 10 (WMU).

### Capture Point B

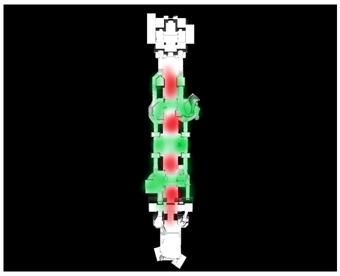
Mostly positive, as the narrow structure of whole map ensures that most enemies will be encountered here, producing numerous intense fights. Also nearby hole/well can be used to throw enemies down. However one user felt it was too compact and an easy place to get killed. Walkways surrounding B can be fun if you can activate traps successfully or knock off opponents, but they are not often visited and hence can be boring or frustrating if you are knocked off.

### Other

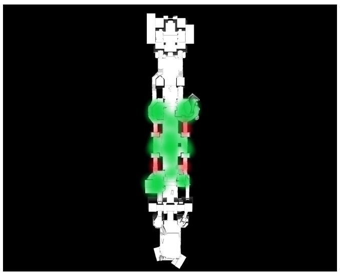
Structure of the whole map “compresses fun” so apart from in spawn points, there weren’t many uneventful moments/areas.



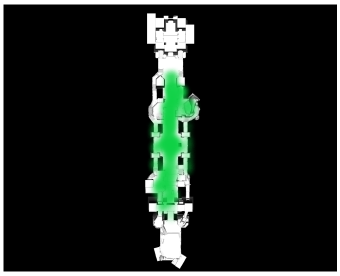
User 1.



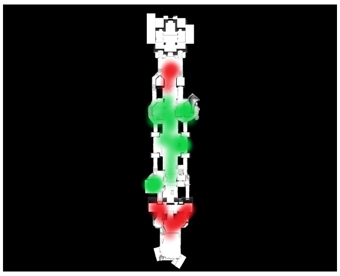
User 2.



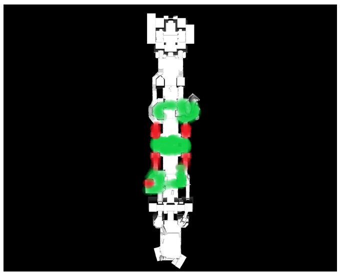
User 3.



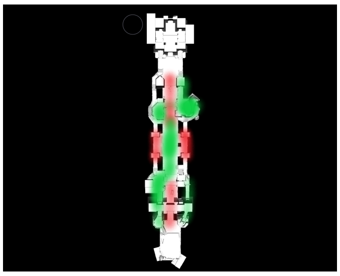
User 4.



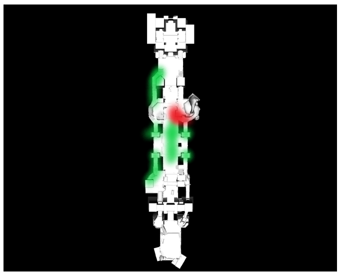
User 5.



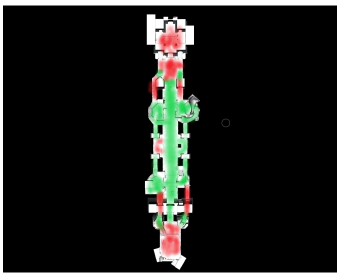
User 6 (BMU).



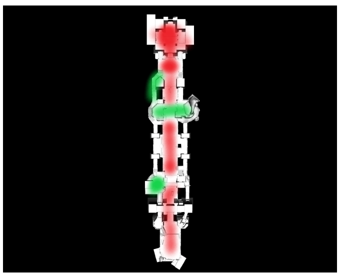
User 7.



User 8.



User 9.



User 10 (WMU).

## Riverfort

### Capture Point A

Majority opinion — good — elevated position makes for good vantage point from which to observe battlefield. Very fun and easy to defend due to opportunities for 1v1 and throwing/kicking players off. However these same things make it very difficult for attackers due to narrow passages either side of A serving as bottlenecks making it hard to manoeuvre. Also players can get stuck on/kicked off ladders.

### Capture Point B

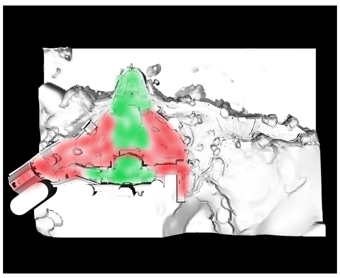
Players generally enjoyed fighting on an open battlefield where they can easily kill minions and use them to farm up rank, but admitted it can be annoying when harassed/swarmed by minions, which also provide cover for enemies to attack. Ladders leading from here to A are bad (mentioned above).

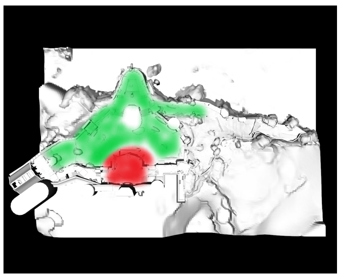
### Capture Point C

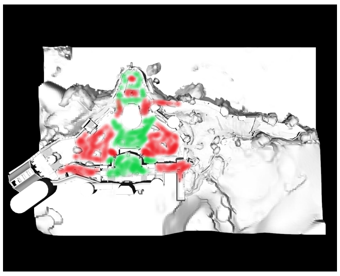
Mixed opinions — geometry of C provides interesting and challenging combat situations. Area is secluded from other points. Upper area generally good, lower area generally bad due to enclosed space and being prone to ambush. Similar to A, bottlenecks make defending easy and attacking hard.

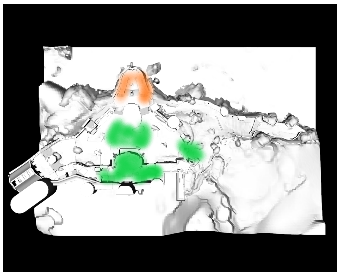
### Other

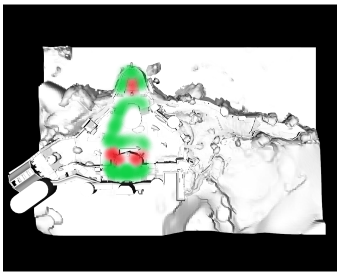
Spawn points/outskirts of map — mostly neutral to bad. Most players see these as boring or just sprint zones. Teammates not waiting for each other can create sense of discord. However opening experience of rushing into battle with teammates is exciting.

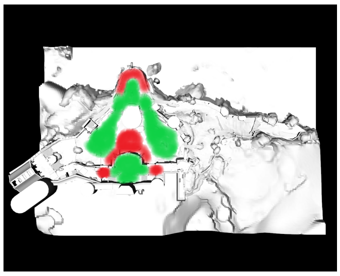

User 1.

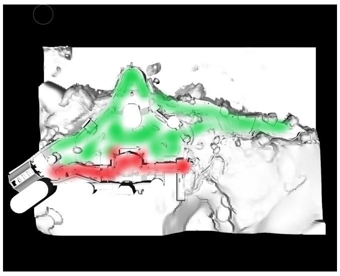

User 2.

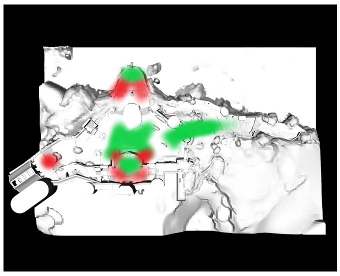

User 3.

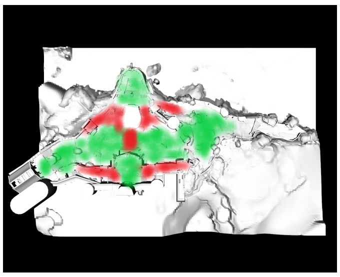

User 4.

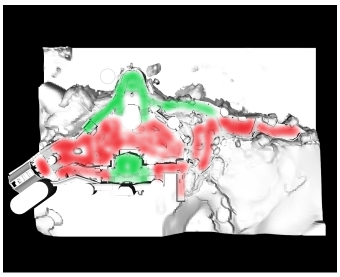

User 5.


User 6 (BMU).


User 7.


User 8.


User 9.


User 10 (WMU).

## Highfort

### Capture Point A

Almost unanimously positive. Epic feeling from having to climb up to reach it. Good view of battlefield. Ladder provides quick escape but User 4 complained that one can get magnetised on to the ladder (this person had a problem with ladders in general). Circular shape and middle pillar give room to fight and obstacle to play with, respectively. Good fights here.

### Capture Point B

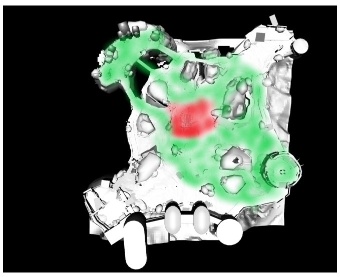
Mixed feedback. Open area with lots of minions but tends to get slightly more negative feedback overall compared to minion zones of other maps. Low amount of environmental hazards except for a fire pit which provides fun. Can get easily swarmed/interrupted by minions, making it easier to die at the hands of enemy players, who use cover points to ambush. Cart acts as a blocker into which the player can be pushed, breaking the flow of combat.

### Capture Point C

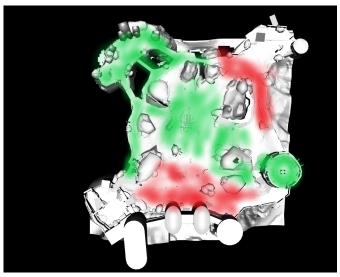
Almost unanimously positive. Plenty of space to fight. Cliffs and bridges provide interesting gameplay with opportunities to throw people off. Risk of falling/getting knocked off bridges was exciting for most, but an issue for User 3. Risky nature of these bridges make it easier to defend C, and attackers feel like they are working towards the point. View from this area also pleasing.

### Other

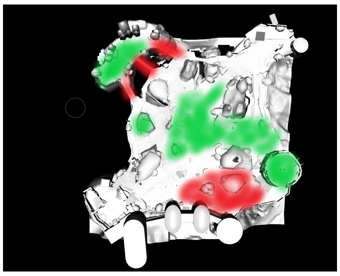
Map is very large — results in underused areas between spawn points and control zones, however these regions have good visibility.

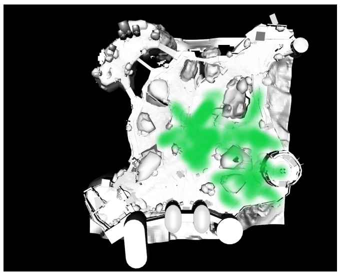


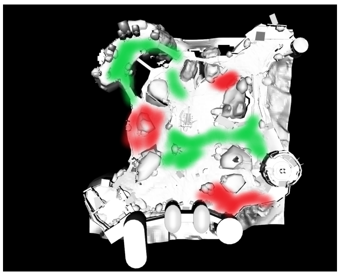
User 1.

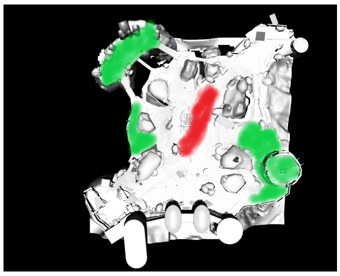


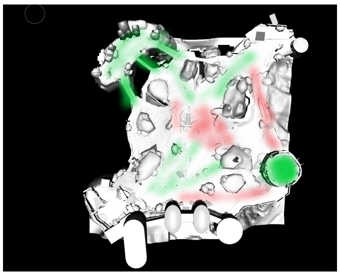
User 2.

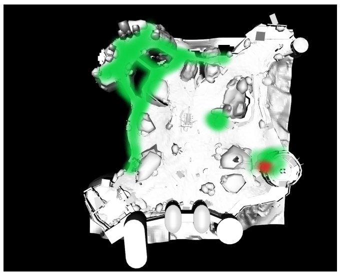

User 3.

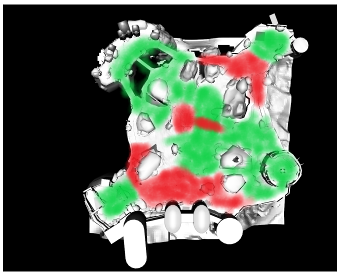

User 4.

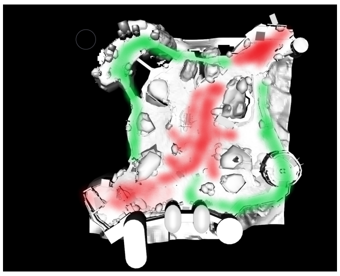

User 5.


User 6 (BMU).


User 7.


User 8.


User 9.


User 10 (WMU).

## The Shard

### Capture Point A

Majority of users gave positive feedback for the point itself, and negative feedback for the corridors leading up to it. Elevation makes for good vantage point, spike wall is fun to push enemies into, and contains separate spaces for fair fights.

### Capture Point B

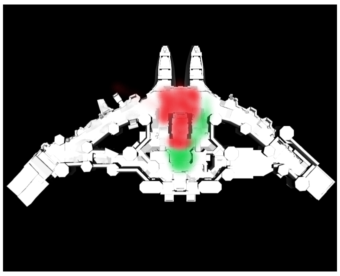
Fighting pit with many environmental hazards and possible ambushes from enemies jumping/climbing down from above — chaotic and intense gameplay. Most users gave positive feedback for these features, but User 8 felt it was too compact and marked it red. One user mentioned the insta-death pit may be excessive.

### Capture Point C

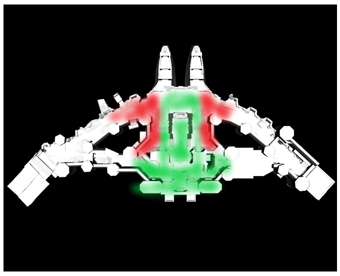
This is where the minions fight, unlike in the other maps. User 8 marked this red because they felt it was too compact and impeded player movement, but User 10 marked it red because they felt it was large and boring, preferring the more "interior" points. Most users liked this point, mainly because they either like killing minions or because one can avoid the minions and easily move to the other points via the two large areas on either side of the point. Also there is good line of sight between this point and the other points.

### Other

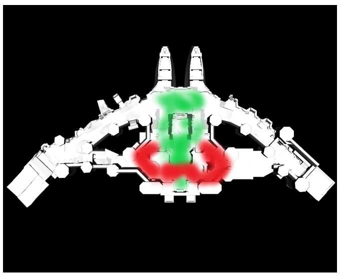
Routes/corridors between points can feel cramped/awkward.



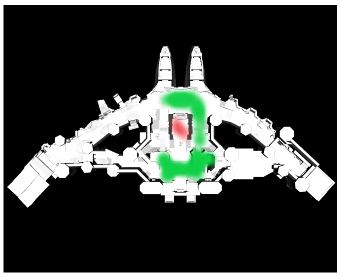
User 1.



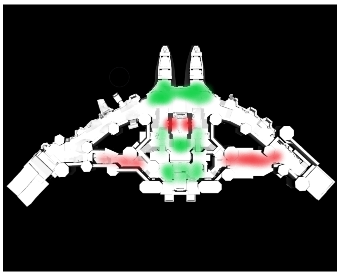
User 2.



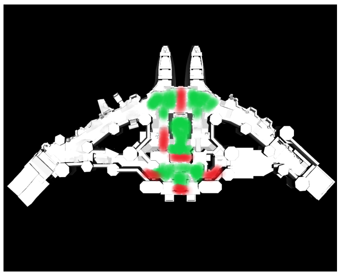
User 3.



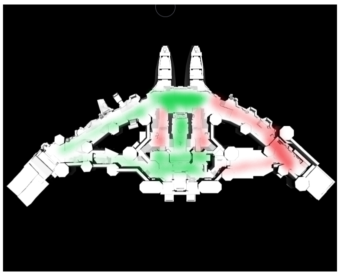
User 4.



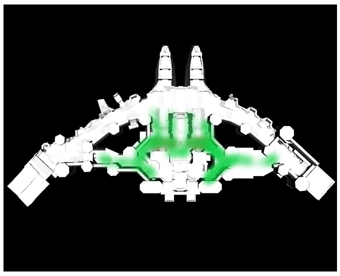
User 5.



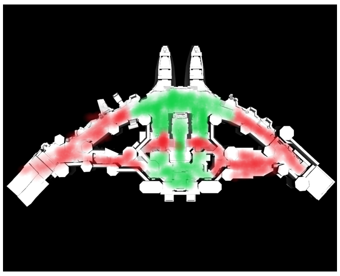
User 6 (BMU).



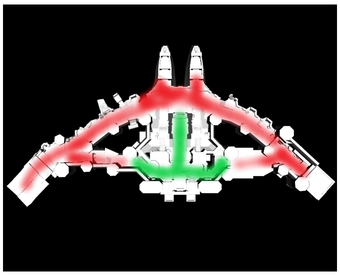
User 7.



User 8.



User 9.



User 10 (WMU).

# Appendix C — Genetic Algorithm Implementation

1. A population of random candidate solutions is initialised:

$$\{G_1,\ G_2,\ G_3, \ldots G_P\},$$

where $P$ is the size of the population.

2. The cost for each candidate is computed using equation 2.13 and the two with the lowest values are chosen e.g:

$$G_I = \{1, 0, 0, 2, 1, 3, 1, 0\} \qquad G_{II} = \{2, 2, 1, 1, 0, 1, 1, 0\}.$$

3. Crossover — each of these solutions are split in half, swapped and recombined:

$$G_I = \{1, 0, 0, 2,\ 0, 1, 1, 0\} \qquad G_{II} = \{2, 2, 1, 1,\ 1, 3, 1, 0\}.$$

4. Mutation — with a certain probability, one of the elements of the resulting "offspring" are changed:

$$G_I = \{1, 0, 0, 2,\ 0, 1, 1, 1\} \qquad G_{II} = \{2, 2, 1, 1,\ 1, 3, 1, 0\}.$$

5. These two new candidates are added to the population and the two which give the highest cost are deleted.

6. Steps 2 to 5 are repeated for a set number of generations.

# Appendix D — Negative Dirichlet Loss

The expression for the log-likelihood of the Dirichlet distribution is:

$$NLL \ = -\log\left(\frac{1}{B(\mathbf{c})}\prod_{i}^{5}\theta_i^{c_i-1}\right),$$

where $\mathbf{c} = \{c_1,\ c_2,\ c_3,\ c_4,\ c_5\}$ are the counts outputted by the neural network, $\boldsymbol{\theta} = \{\theta_1,\ \theta_2,\ \theta_3,\ \theta_4,\ \theta_5\}$ are the parameters to be trained and

$$B(\mathbf{c}) = \frac{\Gamma(c_1)\Gamma(c_2)\Gamma(c_3)\Gamma(c_4)\Gamma(c_5)}{\Gamma(c_1+c_2+c_3+c_4+c_5)}$$

is the Beta function. We then substitute this into the loss and include a prior via $\boldsymbol{\theta} \longrightarrow \boldsymbol{\theta} + \boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = \{0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2\}$. This is equivalent to observing one of each star rating at the start before updating as more data is observed. After expansion, the loss now becomes

$$\begin{aligned} NLL \ =& \log\Gamma(c_1) + \log\Gamma(c_2) + \log\Gamma(c_3) + \log\Gamma(c_4) + \log\Gamma(c_5) - \log\Gamma(c_1+c_2+c_3+c_4+c_5) \\ & - (c_1-1)\log(\theta_1+\alpha_1) - (c_2-1)\log(\theta_2+\alpha_2) - (c_3-1)\log(\theta_3+\alpha_3) \\ & - (c_4-1)\log(\theta_4+\alpha_4) - (c_5-1)\log(\theta_5+\alpha_5), \end{aligned}$$

where $\log\Gamma$ is the log-Gamma function.

# Appendix E — Preliminary Map Evaluation Results by Geometry Capture Metric

| | Citadel Gate | Overwatch | Sanctuary Bridge | Riverfort | Highfort | The Shard | Mean |
|---|---|---|---|---|---|---|---|
| User 1 | 50 | 45.6 | 48.6 | 41.4 | 47.1 | 65.4 | 49.7 |
| User 2 | 42.9 | 48.5 | 39.5 | 31.9 | 38.7 | 43.6 | 40.9 |
| User 3 | 46.8 | 55.2 | 50.5 | 54.8 | 43.9 | 51.7 | 50.5 |
| User 4 | 52.2 | 57.2 | 53.1 | 59.5 | 51.6 | 68.0 | 56.9 |
| User 5 | 62.0 | 61.2 | 38.8 | 60.6 | 52.1 | 50.2 | 54.2 |
| User 6 | 58.3 | 64.5 | 55.6 | 52.7 | 70.6 | 66.6 | 61.4 |
| User 7 | 43.9 | 45.8 | 37.9 | 30.8 | 48.0 | 37.7 | 40.7 |
| User 8 | 53.7 | 60.3 | 54.6 | 40.9 | 70.9 | 53.2 | 55.6 |
| User 9 | 40.5 | 37.8 | 37.5 | 44.9 | 43.0 | 41.1 | 40.8 |
| User 10 | 50.6 | 40.7 | 40.4 | 48.7 | 44.8 | 40.4 | 44.3 |
| Mean | 50.1 | 51.7 | 45.7 | 46.6 | 51.1 | 51.8 | |

Table 1: Map Proximity.

| | Citadel Gate | Overwatch | Sanctuary Bridge | Riverfort | Highfort | The Shard | Mean |
|---|---|---|---|---|---|---|---|
| User 1 | 48.3 | 47.1 | 51.0 | 42.8 | 44.8 | 60.5 | 49.1 |
| User 2 | 40.7 | 44.2 | 43.3 | 62.4 | 31.3 | 40.8 | 43.8 |
| User 3 | 46.6 | 49.1 | 46.7 | 48.2 | 41.1 | 58.7 | 48.4 |
| User 4 | 55.2 | 61.8 | 42.6 | 52.0 | 55.3 | 66.6 | 55.6 |
| User 5 | 60.5 | 64.5 | 34.1 | 64.0 | 46.0 | 51.1 | 53.4 |
| User 6 | 56.7 | 59.9 | 57.1 | 54.5 | 75.0 | 66.4 | 61.6 |
| User 7 | 42.5 | 55.5 | 43.3 | 62.9 | 45.6 | 42.4 | 48.7 |
| User 8 | 56.8 | 57.7 | 48.2 | 49.9 | 68.6 | 50.8 | 55.3 |
| User 9 | 36.9 | 49.6 | 30.7 | 49.7 | 41.2 | 42.2 | 41.7 |
| User 10 | 48.9 | 30.8 | 48.4 | 47.4 | 47.9 | 33.7 | 42.9 |
| Mean | 49.3 | 52.0 | 44.5 | 53.4 | 49.7 | 51.3 | |

Table 2: Nested Spheres.

| | Citadel Gate | Overwatch | Sanctuary Bridge | Riverfort | Highfort | The Shard | Mean |
|---|---|---|---|---|---|---|---|
| User 1 | 58.7 | 51.3 | 53.1 | 40.8 | 47.6 | 60.5 | 52.0 |
| User 2 | 32.1 | 50.4 | 41.1 | 60.0 | 47.5 | 41.5 | 45.4 |
| User 3 | 51.2 | 52.0 | 54.9 | 47.6 | 52.4 | 48.6 | 52.8 |
| User 4 | 62.5 | 70.9 | 56.7 | 53.9 | 53.4 | 65.6 | 60.5 |
| User 5 | 57.9 | 62.9 | 41.7 | 60.5 | 54.3 | 51.0 | 54.7 |
| User 6 | 57.0 | 60.1 | 60.4 | 56.4 | 68.8 | 65.8 | 61.4 |
| User 7 | 54.9 | 52.1 | 40.3 | 63.0 | - | 40.5 | 50.2 |
| User 8 | 58 | 57.7 | 56.1 | 51.4 | 68.9 | 46.5 | 56.4 |
| User 9 | 46.4 | 45.6 | 41.6 | 51.5 | 44.6 | 45.2 | 45.8 |
| User 10 | 25 | 34.6 | 42.8 | 37.3 | 31.1 | 31.0 | 33.6 |
| Mean | 50.4 | 53.8 | 48.9 | 52.2 | 52.1 | 50.6 | |

Table 3: Intersection Distance.

| | Citadel Gate | Overwatch | Sanctuary Bridge | Riverfort | Highfort | The Shard | Mean |
|---|---|---|---|---|---|---|---|
| User 1 | 50.1 | 43.0 | 49.8 | 44.3 | 45.6 | 63.4 | 49.4 |
| User 2 | 34.3 | 51.9 | 42.8 | 32.6 | 43.2 | 39.0 | 40.6 |
| User 3 | 36.5 | 59.5 | 56.3 | 48.8 | 46.2 | 58.5 | 51.0 |
| User 4 | 49.6 | 49.7 | 56.4 | 54.3 | 46.3 | 66.1 | 53.7 |
| User 5 | 50.8 | 60.3 | 39.3 | 57.7 | 55.1 | 50.5 | 52.3 |
| User 6 | 52.6 | 65.3 | 58.5 | 47.4 | 69.3 | 65.0 | 59.7 |
| User 7 | 38.1 | 47.5 | 40.5 | 35.4 | - | 40.3 | 40.4 |
| User 8 | 48.1 | 64.2 | 58.4 | 41.5 | 74.3 | 41.0 | 54.6 |
| User 9 | 41.3 | 40.5 | 43.3 | 44.5 | 42.8 | 44.0 | 42.7 |
| User 10 | 52.0 | 44.0 | 43.2 | 48.0 | 39.5 | 28 | 42.5 |
| Mean | 45.3 | 52.6 | 48.9 | 45.5 | 51.4 | 49.6 | |

Table 4: Log-distance.

Note: Some entries have no values as that particular user's map was not available at the time these values were computed.

# Appendix F — Full Results

Figure .13: Heat Map Accuracy Results (method 1).

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) |
|---|---|---|---|---|---|---|---|---|---|
| **User 1** | | | | | | | | | |
| Citadel Gate | 55.42680034 | 56.01291875 | 55.18873834 | 55.26633184 | 55.12244152 | 55.95727602 | 55.75135223 | 54.70622369 | 56.29879666 |
| Overwatch | 42.71783726 | 42.59956987 | 43.98915892 | 43.29935269 | 44.28291858 | 43.8285241 | 41.41719112 | 43.05172231 | 43.74793028 |
| Sanctuary Bridge | 51.52747143 | 53.03269288 | 51.45744787 | 50.18268922 | 50.2935542 | 51.86601172 | 50.90444128 | 50.50121884 | 51.48640357 |
| Riverfort | 37.44669781 | 41.38919032 | 40.33176282 | 40.59906935 | 42.62831332 | 40.61575724 | 40.80136986 | 39.79560999 | 40.72851518 |
| Highfort | 48.86906517 | 51.18608241 | 51.98232943 | 50.71091066 | 50.9883761 | 51.49619273 | 51.19321854 | 48.7056585 | 50.57240259 |
| Shard | 68.06422885 | 67.02412067 | 65.87857244 | 68.14078675 | 67.333833 | 66.90703389 | 67.28544181 | 68.8499622 | 66.38143199 |
| **User 2** | | | | | | | | | |
| Citadel Gate | 36.98682127 | 37.10787269 | 37.90862809 | 37.8709523 | 37.79725659 | 37.80823145 | 36.70009448 | 37.14415278 | 35.33512137 |
| Overwatch | 46.58766082 | 46.31711302 | 47.64128942 | 46.07782329 | 45.0351062 | 47.13093433 | 45.38449857 | 47.15680785 | 47.21235754 |
| Sanctuary Bridge | 41.64210564 | 41.18007911 | 41.43128206 | 42.41523191 | 42.49505668 | 40.97144754 | 42.23489881 | 42.29638155 | 41.63075313 |
| Riverfort | 47.21014893 | 42.55975241 | 44.54360484 | 42.67849388 | 43.42133016 | 46.32383337 | 43.58688714 | 46.0199834 | 43.74679715 |
| Highfort | 41.03692229 | 41.25648485 | 38.89771339 | 38.02699266 | 37.96946099 | 39.41921491 | 38.94481526 | 40.91802908 | 37.73433606 |
| Shard | 48.71622114 | 46.73308674 | 47.92613546 | 50.36292172 | 47.64882758 | 48.64826524 | 49.54493283 | 48.12022753 | 48.9793851 |
| **User 3** | | | | | | | | | |
| Citadel Gate | 47.46569875 | 46.88717843 | 47.99258837 | 48.04922348 | 48.68487636 | 46.7744579 | 47.70668069 | 46.15922432 | 47.9891516 |
| Overwatch | 56.24067528 | 55.6820954 | 57.28321343 | 56.1198069 | 54.93754103 | 57.12066015 | 55.19760116 | 56.12361204 | 57.44137802 |
| Sanctuary Bridge | 53.40170909 | 56.46413809 | 53.90079027 | 54.55136558 | 54.37130611 | 54.8843553 | 54.40238653 | 54.0299227 | 55.45254566 |
| Riverfort | 45.95190086 | 46.60824023 | 45.80271312 | 46.48996485 | 46.98710469 | 47.66994092 | 46.53010343 | 46.13095526 | 46.26691003 |
| Highfort | 46.30852336 | 47.40497133 | 46.75811012 | 47.09521789 | 46.82609944 | 47.41180483 | 47.8239863 | 47.27594217 | 45.19853103 |
| Shard | 48.40814415 | 47.39795487 | 49.45179029 | 47.71796118 | 45.79669262 | 48.44716371 | 48.03435174 | 48.19535008 | 46.66018583 |
| **User 4** | | | | | | | | | |
| Citadel Gate | 60.16250188 | 59.2801761 | 59.64086577 | 56.9899242 | 58.98689174 | 58.23078555 | 59.37577506 | 59.10343415 | 58.29801599 |
| Overwatch | 50.10825609 | 49.06856592 | 51.63865149 | 47.22831334 | 55.0828801 | 48.84089365 | 46.07782259 | 52.27857632 | 49.77218639 |
| Sanctuary Bridge | 55.04026244 | 58.79472551 | 55.70161307 | 56.28886948 | 53.45731348 | 56.39159502 | 55.99423527 | 53.66588431 | 57.94592586 |
| Riverfort | 49.6759449 | 51.74471357 | 50.8416034 | 52.44407151 | 51.91130353 | 51.63361 | 52.20774993 | 50.16728616 | 51.99913353 |
| Highfort | 55.20811812 | 60.15385628 | 61.59494904 | 65.10508372 | 63.06212795 | 61.18050387 | 62.03114186 | 56.02979717 | 60.1820709 |
| Shard | 68.92816526 | 68.09987633 | 68.5467411 | 67.59271031 | 67.33503421 | 68.07782563 | 68.43678819 | 68.15311496 | 67.42451842 |
| **User 5** | | | | | | | | | |
| Citadel Gate | 54.30631418 | 55.57365078 | 55.55253287 | 55.79450457 | 54.35691928 | 54.96039489 | 55.60159197 | 53.41189328 | 54.44320978 |
| Overwatch | 58.66798945 | 59.3978794 | 59.71775835 | 59.02762946 | 55.76977935 | 59.15315878 | 59.45868748 | 59.65440146 | 59.70190492 |
| Sanctuary Bridge | 40.04712236 | 42.07925446 | 40.63850976 | 40.19884872 | 39.32024177 | 40.67312587 | 40.48379911 | 40.15086406 | 41.55207356 |
| Riverfort | 58.68307201 | 58.54007481 | 58.31907417 | 58.31026813 | 58.34186576 | 59.36834568 | 59.17021771 | 59.24532262 | 58.39818803 |
| Highfort | 53.18968691 | 51.97202436 | 50.60528484 | 52.32782036 | 52.59518001 | 50.98477309 | 52.28644658 | 52.78158195 | 50.7909345 |
| Shard | 52.20504269 | 51.13522818 | 51.89711368 | 51.25329748 | 49.52323385 | 51.71394046 | 51.61957439 | 51.60626771 | 50.54249607 |
| **User 6** | | | | | | | | | |
| Citadel Gate | 55.60188616 | 57.013267 | 57.25028647 | 57.59098896 | 56.36240477 | 56.11587905 | 57.05641066 | 55.60080968 | 56.2367208 |
| Overwatch | 62.78371846 | 62.88483123 | 62.13270515 | 63.58941633 | 59.98323186 | 63.82394785 | 63.65527611 | 63.51622836 | 64.17952996 |
| Sanctuary Bridge | 57.5651512 | 58.85615108 | 58.13481102 | 58.82039065 | 57.91708276 | 59.82494545 | 58.31060128 | 57.59425832 | 59.13552211 |
| Riverfort | 50.13892538 | 50.18682385 | 50.54306398 | 51.03656788 | 50.50856331 | 51.17247163 | 50.67363329 | 50.08153145 | 49.78777071 |
| Highfort | 68.23816525 | 67.78790014 | 69.71566082 | 66.06688043 | 67.23832637 | 68.40526866 | 67.77032573 | 68.56529017 | 66.14946279 |
| Shard | 64.84543832 | 64.91899088 | 65.13263407 | 65.29763441 | 64.19424263 | 64.7518226 | 66.00489548 | 65.44175522 | 64.36396711 |
| **User 7** | | | | | | | | | |
| Citadel Gate | 48.90796385 | 50.00703665 | 49.27189416 | 50.05316503 | 48.73690669 | 48.37511169 | 50.70653925 | 47.71392947 | 51.84935444 |
| Overwatch | 46.21836487 | 48.11650689 | 48.34575207 | 46.16213795 | 46.6620457 | 47.32652313 | 47.41646012 | 47.34528193 | 47.37952631 |
| Sanctuary Bridge | 40.07024857 | 39.6617841 | 40.11530124 | 40.39803285 | 39.96056221 | 39.9247705 | 41.00192407 | 40.9573452 | 40.46466276 |
| Riverfort | 45.55115582 | 41.55175496 | 43.70416504 | 41.02689529 | 43.67712391 | 45.71667069 | 41.92435583 | 43.30631743 | 41.62925701 |
| Highfort | 46.30430543 | 45.09109886 | 44.67746635 | 42.84391293 | 43.53869363 | 44.28928356 | 43.81662151 | 46.15895038 | 43.46629211 |
| Shard | 37.26056583 | 41.48850835 | 37.57346818 | 41.07689761 | 40.9272519 | 40.68163545 | 40.95317617 | 37.80143054 | 40.78035389 |
| **User 8** | | | | | | | | | |
| Citadel Gate | 53.25751919 | 54.3340931 | 54.26432339 | 53.27815101 | 52.70702992 | 52.75904051 | 53.77050192 | 52.73689008 | 52.72456136 |
| Overwatch | 60.70866492 | 61.53333144 | 62.37758513 | 60.80443607 | 59.81499341 | 62.05398483 | 61.36995199 | 61.24904587 | 61.85319247 |
| Sanctuary Bridge | 54.52970974 | 55.68008838 | 55.1077754 | 55.0356729 | 55.33292856 | 55.18493458 | 54.65375031 | 53.70938188 | 54.51905284 |
| Riverfort | 41.11082411 | 41.13440099 | 42.52418496 | 41.1979352 | 42.07700192 | 42.59522037 | 41.80085033 | 41.0921398 | 41.65666409 |
| Highfort | 67.71262454 | 63.61049672 | 63.30701591 | 58.8587981 | 61.06177032 | 63.19178971 | 61.37407544 | 66.76914942 | 61.91792861 |
| Shard | 62.88917486 | 60.69198394 | 63.56774393 | 63.81061616 | 61.63497711 | 61.73857261 | 62.4051018 | 59.28604076 | 63.86029765 |
| **User 9** | | | | | | | | | |
| Citadel Gate | 43.86633511 | 43.45357919 | 42.30338385 | 42.56627639 | 44.05046579 | 44.05000726 | 43.32843892 | 44.15371809 | 43.41138807 |
| Overwatch | 43.81369576 | 45.63212636 | 44.53132393 | 43.63375878 | 45.4102172 | 44.38174162 | 44.29499863 | 43.93916888 | 43.31207 |
| Sanctuary Bridge | 40.10751149 | 42.07222858 | 39.96305164 | 40.96775706 | 37.76607371 | 40.65313228 | 39.46160887 | 39.38929416 | 40.72598442 |
| Riverfort | 45.00650618 | 43.53974114 | 45.02873005 | 45.15511903 | 44.87838177 | 43.93925054 | 45.1770914 | 44.09937926 | 43.77584432 |
| Highfort | 44.16092981 | 45.17579639 | 43.9987397 | 43.81195984 | 43.08722224 | 44.73665704 | 43.78391402 | 43.96075945 | 43.39699865 |
| Shard | 38.38295175 | 37.06509619 | 38.69136606 | 37.20333132 | 36.54287172 | 38.33913848 | 38.18630473 | 39.12041125 | 37.89962492 |
| **User 10** | | | | | | | | | |
| Citadel Gate | 32.16654365 | 31.15580312 | 32.31408159 | 31.45883167 | 31.12195066 | 32.43131533 | 30.18790118 | 32.71421514 | 30.81612308 |
| Overwatch | 40.67891452 | 39.70622871 | 41.43470201 | 40.2731495 | 40.64551473 | 41.05960652 | 39.95356115 | 39.8061101 | 40.86113308 |
| Sanctuary Bridge | 44.60711729 | 45.28179952 | 42.90562857 | 45.44601029 | 45.69644136 | 45.62681395 | 46.36452773 | 44.63035224 | 45.6481976 |
| Riverfort | 46.68698593 | 49.58058897 | 48.51753691 | 47.99263373 | 49.80427176 | 48.67864113 | 49.23087401 | 47.84520166 | 48.42325853 |
| Highfort | 38.64532131 | 40.14739887 | 41.14019567 | 36.52325028 | 38.4913845 | 40.3013977 | 38.0226065 | 37.5757466 | 38.29411679 |
| Shard | 49.80098594 | 48.32785268 | 50.12721856 | 51.93769311 | 51.00748694 | 50.08896572 | 51.564154 | 47.56176294 | 51.09209324 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 16, 7) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|---|---|---|---|---|---|
| **User 1** | | | | | | | | | | |
| Citadel Gate | 57.05986242 | 56.16325843 | 57.11946855 | 57.91690017 | 55.3959998 | 55.58018826 | 54.62874807 | 56.22327168 | 57.03205464 | 55.93614619 |
| Overwatch | 42.07769897 | 43.61402867 | 43.98067806 | 42.81727819 | 43.43017241 | 43.57559748 | 43.30682805 | 43.77824929 | 42.91221446 | 43.24594171 |
| Sanctuary Bridge | 50.38212228 | 51.77525693 | 52.0558554 | 52.10785406 | 52.12013034 | 52.06593446 | 52.69378619 | 52.77217388 | 52.46254153 | 51.64931034 |
| Riverfort | 40.04904816 | 41.59187537 | 38.52671905 | 40.13527582 | 39.65169695 | 41.58047176 | 41.23001483 | 40.11731811 | 41.79240955 | 40.50061753 |
| Highfort | 50.2517357 | 50.31850804 | 51.11173025 | 50.13947077 | 51.18542586 | 50.77410606 | 50.63143048 | 52.13556932 | 50.25625441 | 50.69491483 |
| Shard | 66.50586871 | 67.21131421 | 65.84748871 | 66.84726248 | 66.19548928 | 67.04660342 | 66.33540414 | 67.35979746 | 65.50130013 | 66.92866334 |
| **User 2** | | | | | | | | | | |
| Citadel Gate | 35.48496102 | 36.00025325 | 35.71615918 | 36.77763843 | 35.86091742 | 36.01120797 | 37.65429493 | 38.6298417 | 37.14148327 | 36.88532712 |
| Overwatch | 46.34627176 | 47.53046372 | 45.01332767 | 47.32569853 | 46.27037757 | 45.93020003 | 47.43594418 | 47.54968421 | 45.77580698 | 46.54007587 |
| Sanctuary Bridge | 40.42603092 | 41.95613451 | 42.51232212 | 41.73597086 | 42.10466943 | 41.84924605 | 41.98220158 | 41.13817108 | 42.46692826 | 41.8038284 |
| Riverfort | 46.01401531 | 43.52642833 | 47.18389559 | 45.69478899 | 44.10519696 | 45.34195085 | 44.24648756 | 44.30824062 | 45.35643613 | 44.77045954 |
| Highfort | 41.4535983 | 40.55792284 | 38.52951843 | 37.75372359 | 40.3278263 | 39.71730339 | 41.45195345 | 39.57629636 | 39.42631767 | 39.61102388 |
| Shard | 48.42278078 | 49.80731776 | 49.23181522 | 50.05604175 | 49.02320075 | 51.13108202 | 49.04163451 | 49.34901897 | 49.23546231 | 48.99879763 |
| **User 3** | | | | | | | | | | |
| Citadel Gate | 47.79509264 | 49.03777041 | 47.61471374 | 48.78546398 | 46.89125216 | 46.80397152 | 46.04157981 | 47.90503286 | 47.85806526 | 47.58011235 |
| Overwatch | 56.49096995 | 56.78388312 | 52.86092623 | 56.49332861 | 56.41081024 | 55.94111763 | 57.01005027 | 57.09620245 | 55.02401382 | 56.1254381 |
| Sanctuary Bridge | 53.79068664 | 56.18203268 | 55.60019355 | 54.38864767 | 55.65886076 | 55.15126142 | 54.00139555 | 54.31386743 | 56.98548625 | 54.86283063 |
| Riverfort | 46.83743581 | 47.54260939 | 44.88547864 | 47.06633611 | 46.60590505 | 46.4283886 | 47.30947416 | 46.93608349 | 47.01349742 | 46.61461345 |
| Highfort | 47.10327289 | 46.80179849 | 45.510319 | 45.73374599 | 47.12815342 | 47.00438074 | 47.20211127 | 48.06206527 | 46.58347209 | 46.84625031 |
| Shard | 48.07868327 | 48.34816022 | 48.66424174 | 47.50964023 | 49.55861037 | 48.1829424 | 48.49165349 | 47.0870408 | 47.94369017 | 47.99856984 |
| **User 4** | | | | | | | | | | |
| Citadel Gate | 59.26709868 | 58.10812593 | 60.5391743 | 58.51234387 | 57.30071734 | 58.89668813 | 57.87167427 | 59.83185111 | 58.58945271 | 58.8325276 |
| Overwatch | 45.98495648 | 51.11218183 | 58.84386493 | 49.22643195 | 49.00424654 | 47.47793495 | 50.38760662 | 49.78407604 | 49.64142465 | 50.08660388 |
| Sanctuary Bridge | 56.36990254 | 57.4924393 | 56.72195714 | 56.87993731 | 56.29653367 | 56.14553429 | 55.35488862 | 56.29264411 | 58.02975458 | 56.27022311 |
| Riverfort | 50.03324255 | 51.65894779 | 51.29544427 | 50.19891883 | 51.94784369 | 51.50137828 | 50.88946563 | 51.95516393 | 50.16348511 | 51.2371837 |
| Highfort | 58.55569435 | 58.18302073 | 58.32710493 | 59.90819469 | 61.46267397 | 61.26088365 | 59.26348646 | 61.09709919 | 60.48353447 | 60.17163007 |
| Shard | 67.19603247 | 69.14434772 | 67.8085786 | 68.95123101 | 68.10439104 | 69.81335269 | 69.0159003 | 68.66623743 | 68.67355949 | 68.33157806 |
| **User 5** | | | | | | | | | | |
| Citadel Gate | 55.41437144 | 54.75730037 | 54.89722881 | 55.78938392 | 53.76091957 | 53.88205112 | 54.75306603 | 54.72968189 | 54.22934866 | 54.78968686 |
| Overwatch | 59.37702697 | 59.74765945 | 56.37275358 | 59.42365576 | 59.95404166 | 59.11708026 | 59.13498474 | 60.26948255 | 58.45693466 | 59.02237824 |
| Sanctuary Bridge | 40.31827446 | 42.25623813 | 41.75536878 | 41.02341141 | 41.71728496 | 40.40680731 | 39.83822898 | 40.64385605 | 41.53631816 | 40.81331266 |
| Riverfort | 58.29540661 | 58.20543075 | 57.80786402 | 58.85623881 | 58.47457562 | 58.76902024 | 58.45828472 | 58.50920547 | 58.23828698 | 58.55504123 |
| Highfort | 53.78125894 | 51.32925652 | 51.71392752 | 51.58173632 | 51.6589889 | 50.94826073 | 52.23988144 | 52.39488382 | 50.37164235 | 51.86408718 |
| Shard | 51.62510323 | 51.15113465 | 50.96925362 | 50.92910994 | 52.31243225 | 52.98538897 | 51.97645203 | 52.03338669 | 51.0648812 | 51.47462984 |
| **User 6** | | | | | | | | | | |
| Citadel Gate | 56.33017825 | 58.50809667 | 55.27992088 | 57.09814179 | 55.53003734 | 56.18603518 | 56.57852931 | 56.39298281 | 56.32526175 | 56.5032132 |
| Overwatch | 63.50001481 | 63.04522016 | 59.45256934 | 63.13584007 | 62.06925983 | 61.71810163 | 62.49900033 | 63.14326809 | 61.29386649 | 62.57811256 |
| Sanctuary Bridge | 57.02126888 | 59.45205409 | 59.32019351 | 58.04244262 | 58.89020654 | 59.14339568 | 59.56929251 | 59.05713345 | 59.70843567 | 58.68685205 |
| Riverfort | 50.04867153 | 50.82496372 | 50.45816392 | 50.08074839 | 51.65691594 | 51.32834528 | 50.12886907 | 50.73655607 | 50.53885306 | 50.55174658 |
| Highfort | 67.93666249 | 68.47024076 | 67.17117008 | 66.92555988 | 65.75261337 | 66.43783998 | 66.8325187 | 69.44529549 | 65.53864989 | 67.46932395 |
| Shard | 63.5027404 | 65.61251067 | 64.38021927 | 64.92507958 | 64.26104629 | 65.68343055 | 64.4762163 | 65.28252966 | 64.804554 | 64.88220597 |
| **User 7** | | | | | | | | | | |
| Citadel Gate | 49.90411306 | 51.36822594 | 49.7518041 | 51.56631289 | 49.75924267 | 50.69643516 | 49.19581203 | 48.4658349 | 51.09626423 | 49.85699701 |
| Overwatch | 47.45315059 | 48.96416033 | 45.43224946 | 45.92348533 | 48.07391412 | 48.73000405 | 47.26426713 | 47.22283381 | 48.39307151 | 47.35720752 |
| Sanctuary Bridge | 39.90594342 | 40.93499109 | 39.3767704 | 40.09466462 | 40.32481507 | 39.72016819 | 40.48463966 | 40.47758611 | 39.7970476 | 40.20395876 |
| Riverfort | 45.23828694 | 43.18186802 | 45.02409739 | 44.23826738 | 43.63262699 | 45.13060172 | 43.10114583 | 43.24444226 | 45.54242678 | 43.69008107 |
| Highfort | 44.99119061 | 44.86599822 | 44.90300716 | 43.891728 | 44.04416704 | 43.24261084 | 44.93710883 | 45.23941897 | 44.53996818 | 44.49121237 |
| Shard | 40.10469413 | 40.06718651 | 40.33184944 | 37.98222601 | 38.45167909 | 38.65580227 | 38.34216313 | 40.0548133 | 39.38739942 | 39.55117229 |
| **User 8** | | | | | | | | | | |
| Citadel Gate | 54.31004737 | 54.04113778 | 53.82098708 | 53.88668726 | 53.91917903 | 53.64846087 | 53.83922337 | 54.64708369 | 52.41478254 | 53.57553886 |
| Overwatch | 62.49272114 | 60.8965038 | 59.42665068 | 60.71963035 | 61.09671391 | 61.81075711 | 61.58251819 | 62.26014121 | 60.06478658 | 61.22864495 |
| Sanctuary Bridge | 54.61900413 | 56.9044745 | 55.63166067 | 54.36523127 | 56.02781557 | 56.11914601 | 54.41264373 | 55.39779339 | 56.87093003 | 55.22788855 |
| Riverfort | 40.37536244 | 43.39600531 | 42.32009494 | 40.22746342 | 42.85543869 | 41.7916437 | 40.93469329 | 41.2710446 | 41.05841004 | 41.6344099 |
| Highfort | 65.45525289 | 65.63853857 | 63.71524001 | 62.51025404 | 61.49772551 | 61.61824775 | 63.34020968 | 64.37077629 | 60.63651121 | 63.14368915 |
| Shard | 63.6964454 | 62.43726526 | 63.80139097 | 64.29061962 | 64.65272312 | 63.85018483 | 64.01443585 | 63.95474675 | 63.25915196 | 62.99119292 |
| **User 9** | | | | | | | | | | |
| Citadel Gate | 44.31029668 | 44.15934085 | 44.4046208 | 43.4722917 | 43.5868657 | 43.07611605 | 43.11783267 | 42.69471059 | 45.00763206 | 43.61184999 |
| Overwatch | 44.85659968 | 45.08705749 | 42.03601695 | 43.74303623 | 44.8798576 | 46.42431333 | 43.86084897 | 44.43018936 | 45.63537205 | 44.43902182 |
| Sanctuary Bridge | 40.51754258 | 41.5399234 | 41.05800176 | 41.83058154 | 41.22977052 | 40.09268523 | 39.07777909 | 41.33711213 | 41.94220261 | 40.54068006 |
| Riverfort | 44.7782099 | 44.66799264 | 44.94566134 | 44.2771203 | 45.98680406 | 44.27303694 | 44.45259271 | 44.79545141 | 44.52408565 | 44.62783326 |
| Highfort | 44.65743524 | 44.52227282 | 43.90463861 | 42.71735733 | 44.30120045 | 43.74548489 | 44.76895716 | 45.25408887 | 43.92150706 | 44.10588442 |
| Shard | 37.36649621 | 37.85825831 | 37.78830697 | 38.30452977 | 38.0984948 | 39.12403935 | 38.01290882 | 37.81475824 | 38.02160311 | 37.99002733 |
| **User 10** | | | | | | | | | | |
| Citadel Gate | 32.69468677 | 30.90081003 | 31.25642992 | 32.52693435 | 31.29813749 | 30.33793763 | 33.00324981 | 33.39426557 | 32.98227786 | 31.82008305 |
| Overwatch | 41.04653388 | 40.65337917 | 37.76001519 | 40.98015272 | 40.49935586 | 39.13304394 | 40.27032715 | 41.21180612 | 39.27919213 | 40.29181814 |
| Sanctuary Bridge | 43.37415836 | 43.17660762 | 46.22577521 | 45.43538346 | 44.65462367 | 45.32648253 | 47.32250895 | 45.62514159 | 45.37814791 | 45.15142877 |
| Riverfort | 49.02145087 | 48.44746338 | 46.97332512 | 49.32968343 | 47.28341271 | 49.22865638 | 48.87308024 | 47.73021561 | 50.51544179 | 48.56459568 |
| Highfort | 39.70939036 | 39.89078219 | 39.85941674 | 39.18278707 | 37.24890941 | 37.35955762 | 38.99801577 | 41.28699289 | 37.68523771 | 38.90902822 |
| Shard | 50.36049819 | 50.95159164 | 50.72389964 | 51.46597853 | 50.49315911 | 53.10563163 | 51.93382871 | 51.65432393 | 50.75882077 | 50.71977474 |

Figure .14: Heat Map Accuracy Results (method 2).

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) |
|---|---|---|---|---|---|---|---|---|---|
| **User 1** | | | | | | | | | |
| Citadel Gate | 59.61290924 | 59.59146789 | 60.25832805 | 58.91597572 | 58.65664841 | 57.36731557 | 58.67477135 | 58.55675112 | 59.32670291 |
| Overwatch | 50.66309486 | 51.23781486 | 50.04052114 | 48.17223236 | 49.05188605 | 50.11641779 | 47.98418293 | 49.52726375 | 46.88428607 |
| Sanctuary Bridge | 50.73128472 | 52.98408728 | 51.85924912 | 53.52226479 | 54.14733852 | 53.012821 | 54.08911125 | 50.85416535 | 54.114018 |
| Riverfort | 38.55519567 | 39.9209262 | 40.57976579 | 40.21633171 | 38.37696711 | 39.88397757 | 38.73677718 | 39.21884529 | 38.01256271 |
| Highfort | 47.5635899 | 48.60464889 | 49.27432653 | 49.68493429 | 49.41948804 | 49.36611617 | 49.77492306 | 47.85895091 | 50.03284165 |
| Shard | 60.90016935 | 60.84803083 | 60.35747982 | 60.84751715 | 60.30723772 | 61.13231976 | 60.84004255 | 61.22033916 | 60.47533838 |
| **User 2** | | | | | | | | | |
| Citadel Gate | 31.2302042 | 31.49650109 | 31.37807207 | 31.57988577 | 31.42424438 | 32.01388814 | 31.55714821 | 31.13652732 | 31.37614894 |
| Overwatch | 49.497472 | 48.73799314 | 47.30029714 | 45.68708714 | 46.08850633 | 47.93993287 | 46.16730438 | 49.24180807 | 45.82046582 |
| Sanctuary Bridge | 39.87827931 | 40.86646368 | 40.5246169 | 41.4949854 | 41.0337156 | 41.53463535 | 41.86560353 | 39.73484608 | 41.70289349 |
| Riverfort | 64.78911149 | 60.51522013 | 60.04957888 | 59.53604695 | 59.18441766 | 57.69920288 | 59.35660762 | 63.18091999 | 59.12559856 |
| Highfort | 47.48261895 | 42.75822043 | 42.66806478 | 40.440077 | 40.05359596 | 43.56100797 | 39.36908228 | 45.693545 | 38.79297437 |
| Shard | 39.48136778 | 41.02094717 | 40.8308995 | 42.2907339 | 42.6103862 | 41.81404409 | 42.27686917 | 39.38124725 | 41.9817723 |
| **User 3** | | | | | | | | | |
| Citadel Gate | 52.89380014 | 53.22105293 | 54.05928617 | 48.8669729 | 49.86095352 | 51.15351127 | 47.87547171 | 50.18587036 | 48.33879983 |
| Overwatch | 51.46629486 | 50.728064 | 49.79306057 | 49.76805592 | 49.73707731 | 49.88922875 | 50.02853473 | 52.22663107 | 50.67605312 |
| Sanctuary Bridge | 51.98260097 | 54.91637669 | 54.09408161 | 54.30774131 | 54.23922251 | 54.50186621 | 54.52337065 | 52.10236427 | 54.4750478 |
| Riverfort | 47.25751415 | 47.76315271 | 47.34720479 | 46.28977614 | 45.91012685 | 45.96918468 | 45.79320269 | 45.3759508 | 45.96140125 |
| Highfort | 52.3896477 | 50.5708893 | 50.48558198 | 49.94634917 | 49.30682759 | 51.04444125 | 49.32745158 | 51.56186992 | 49.08602684 |
| Shard | 59.00237752 | 58.73911556 | 58.66276252 | 57.7943565 | 57.62283042 | 56.74175478 | 57.80556842 | 57.31138527 | 57.26159044 |
| **User 4** | | | | | | | | | |
| Citadel Gate | 63.57544422 | 63.70379129 | 64.1176694 | 59.16953197 | 61.16229406 | 62.46437849 | 59.23816122 | 61.38567543 | 59.38756244 |
| Overwatch | 70.297216 | 69.02083657 | 67.02946743 | 64.95642991 | 65.46339065 | 67.8465942 | 65.45129394 | 69.06222572 | 63.49816004 |
| Sanctuary Bridge | 55.12852625 | 55.87523561 | 56.22687606 | 56.35416993 | 56.82179867 | 55.83642099 | 55.67734003 | 55.32455528 | 55.9945628 |
| Riverfort | 53.81339897 | 54.32797117 | 53.58130652 | 53.63171764 | 53.28411591 | 53.69646704 | 52.7547341 | 53.07805086 | 52.65838189 |
| Highfort | 53.35792408 | 59.47078103 | 59.32895529 | 62.36705909 | 61.85862451 | 59.6836946 | 62.32209168 | 53.67181213 | 62.0948748 |
| Shard | 65.37525548 | 65.19307934 | 65.35267055 | 65.95999917 | 65.718176 | 66.33274521 | 66.33034266 | 64.61564348 | 65.64296231 |
| **User 5** | | | | | | | | | |
| Citadel Gate | 59.27261392 | 59.87019102 | 60.54053396 | 57.10573802 | 57.49095261 | 58.03621093 | 56.40409489 | 58.27623644 | 56.52206109 |
| Overwatch | 62.880768 | 62.12392229 | 61.54673371 | 61.13208168 | 61.50399938 | 61.72009097 | 61.4087067 | 61.98537156 | 60.26557564 |
| Sanctuary Bridge | 40.06680501 | 41.31140351 | 41.28604181 | 42.26361516 | 42.49198435 | 41.24961658 | 42.19460829 | 40.65587081 | 42.23881977 |
| Riverfort | 61.07047994 | 60.47570292 | 60.35717274 | 59.41658437 | 58.94140869 | 59.37658761 | 58.49923468 | 60.7667182 | 58.36513372 |
| Highfort | 54.34195375 | 53.841961 | 53.49923063 | 53.44225602 | 53.34970506 | 53.5689936 | 53.02325876 | 54.72506645 | 53.13594533 |
| Shard | 50.26334537 | 50.91186506 | 50.39409372 | 50.96741955 | 51.22075667 | 50.76959269 | 51.34033244 | 50.31482368 | 51.00092181 |
| **User 6** | | | | | | | | | |
| Citadel Gate | 58.86939915 | 60.06925439 | 60.15185229 | 56.41297665 | 56.35520234 | 59.19318349 | 55.21698465 | 58.04051067 | 55.06092805 |
| Overwatch | 59.35605029 | 59.97950171 | 59.846656 | 59.56814826 | 59.60939196 | 59.71035395 | 59.88251482 | 59.83230325 | 59.89061736 |
| Sanctuary Bridge | 57.62985383 | 61.5577425 | 59.81883263 | 60.8080398 | 60.70971293 | 60.60570149 | 61.38755022 | 56.89449121 | 60.80561119 |
| Riverfort | 58.28329806 | 56.5108884 | 55.52545643 | 54.39729209 | 55.41955078 | 54.82606993 | 54.43064326 | 55.25567509 | 54.37319296 |
| Highfort | 68.76308365 | 62.22992811 | 62.38505451 | 60.40807793 | 60.35517634 | 61.88092748 | 60.02959408 | 68.61324164 | 60.26119491 |
| Shard | 65.41584844 | 65.77080744 | 65.64953597 | 65.25830361 | 65.13505562 | 65.66009719 | 66.01065299 | 64.61967602 | 65.20564767 |
| **User 7** | | | | | | | | | |
| Citadel Gate | 54.0107995 | 56.00654993 | 56.50586376 | 54.0492608 | 54.48701834 | 53.58085472 | 54.14738418 | 53.71953549 | 54.45974336 |
| Overwatch | 52.24290743 | 54.22442057 | 54.58565486 | 56.60841779 | 56.50144924 | 54.30695099 | 56.35282124 | 52.12147089 | 54.4667779 |
| Sanctuary Bridge | 38.46022238 | 40.21770335 | 39.90945256 | 40.88778534 | 40.56602109 | 39.7540179 | 41.06017249 | 38.34103905 | 41.62941885 |
| Riverfort | 66.72555255 | 62.07806962 | 62.98418418 | 62.9134683 | 62.03334433 | 59.67162672 | 62.28138308 | 64.45254354 | 61.81011827 |
| Highfort | 49.14709332 | 48.54254333 | 48.66269505 | 47.28022287 | 47.51161821 | 48.34842228 | 46.86653889 | 48.56810788 | 47.08761225 |
| Shard | 39.181417 | 40.73442727 | 40.13691214 | 39.44419279 | 39.96214162 | 40.23031137 | 39.77224134 | 40.1694497 | 39.51907235 |
| **User 8** | | | | | | | | | |
| Citadel Gate | 57.23009671 | 57.16128662 | 57.65425118 | 56.32324086 | 56.31309346 | 56.21266487 | 56.44527047 | 56.68967741 | 55.8606044 |
| Overwatch | 57.10999771 | 56.655872 | 56.13359543 | 57.5969207 | 57.42839594 | 55.97014843 | 57.66536382 | 57.78000658 | 57.84171333 |
| Sanctuary Bridge | 54.4366612 | 56.29657822 | 55.60018486 | 55.38501538 | 56.38792528 | 55.20229622 | 56.50957821 | 54.14805484 | 56.61161204 |
| Riverfort | 51.2988826 | 52.08462629 | 51.81133246 | 50.66224019 | 50.98072881 | 49.89752571 | 50.90263058 | 49.7215131 | 49.87387593 |
| Highfort | 68.93817442 | 57.50800339 | 58.33343494 | 54.60477475 | 54.67235422 | 57.68428915 | 54.08695822 | 69.69334992 | 53.71934269 |
| Shard | 42.68919607 | 46.3022699 | 45.46916098 | 47.44459321 | 48.00853355 | 47.50090304 | 47.93851801 | 43.7722054 | 47.49748274 |
| **User 9** | | | | | | | | | |
| Citadel Gate | 47.49810633 | 47.34371719 | 47.47421413 | 45.90656643 | 45.57666096 | 45.54328937 | 46.00727168 | 46.14677144 | 46.55096015 |
| Overwatch | 46.14356114 | 48.24980114 | 49.62018743 | 52.29849464 | 51.98302939 | 48.21845816 | 52.0058514 | 46.68896284 | 51.2616763 |
| Sanctuary Bridge | 40.85976566 | 40.97952008 | 40.75232667 | 40.96872012 | 41.41496165 | 40.99624215 | 40.77574594 | 40.8179355 | 41.37129704 |
| Riverfort | 52.90553778 | 51.61311727 | 51.48353688 | 51.75911206 | 51.22342274 | 50.85834891 | 50.84319708 | 51.93793444 | 50.75101316 |
| Highfort | 44.56831832 | 43.64008623 | 43.71545832 | 43.00705666 | 42.75070839 | 44.30806176 | 42.78164855 | 44.34284878 | 42.57414499 |
| Shard | 43.82302937 | 45.26622342 | 44.84614903 | 44.63637615 | 44.84902906 | 45.74987591 | 44.68042295 | 43.66661403 | 44.47173455 |
| **User 10** | | | | | | | | | |
| Citadel Gate | 25.04249929 | 24.30651374 | 24.01926281 | 28.3083064 | 26.81506059 | 24.64981887 | 28.30327513 | 27.67935447 | 27.54077025 |
| Overwatch | 33.67074743 | 32.78813257 | 31.80732343 | 30.45400478 | 30.63552887 | 31.81212788 | 30.60550123 | 34.28869222 | 31.65274394 |
| Sanctuary Bridge | 40.33630869 | 43.4446861 | 42.10473852 | 42.93533031 | 42.35221973 | 43.02195082 | 43.644055 | 39.80132486 | 43.46110139 |
| Riverfort | 35.62138321 | 36.74553044 | 36.66492412 | 37.25644749 | 35.07783789 | 38.30678826 | 36.06117555 | 39.61410118 | 36.17407742 |
| Highfort | 31.14895153 | 27.07944869 | 27.95912963 | 27.69775572 | 27.36894287 | 27.3318527 | 27.73175121 | 33.61590948 | 27.9544924 |
| Shard | 28.88296781 | 30.57866242 | 30.1094246 | 31.90244635 | 32.42517618 | 31.6321258 | 32.58268577 | 28.72661734 | 32.11046737 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 16, 7) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|---|---|---|---|---|---|
| **User 1** | | | | | | | | | | |
| Citadel Gate | 57.57013396 | 56.71315396 | 58.21350929 | 58.21350973 | 57.93993988 | 57.38775364 | 57.54797608 | 57.64017672 | 56.97807494 | 58.28694991 |
| Overwatch | 49.94376926 | 49.84833829 | 49.08536112 | 49.19754971 | 49.56688085 | 50.05304589 | 48.76403549 | 49.45651168 | 49.16550781 | 49.37548333 |
| Sanctuary Bridge | 52.9568205 | 53.39329956 | 52.98014517 | 52.64447173 | 53.36564236 | 53.23734272 | 53.13079175 | 52.56565242 | 52.85979411 | 52.91379446 |
| Riverfort | 39.20478464 | 39.94932287 | 37.61351487 | 38.9553826 | 39.03932986 | 39.29442689 | 38.62844935 | 38.80522888 | 39.10016146 | 39.11621948 |
| Highfort | 48.90230903 | 49.37214291 | 49.08052587 | 49.15714786 | 49.5003778 | 48.91765412 | 49.68355594 | 49.25713366 | 49.01544397 | 49.13700614 |
| Shard | 60.66375228 | 61.41383429 | 60.95533171 | 61.18880995 | 61.39311377 | 60.87728568 | 60.52178556 | 60.76608063 | 60.62748337 | 60.85199733 |
| **User 2** | | | | | | | | | | |
| Citadel Gate | 32.3347806 | 33.27917101 | 32.29590773 | 32.29590773 | 32.56856269 | 32.8982862 | 32.83254326 | 32.6542906 | 33.03673765 | 32.07715598 |
| Overwatch | 49.20145108 | 47.51005099 | 47.44918694 | 47.08256914 | 47.43251321 | 48.83452645 | 47.6101285 | 48.16092906 | 47.41813295 | 47.62113085 |
| Sanctuary Bridge | 41.22639689 | 41.36712853 | 41.03701824 | 40.80724234 | 41.10353327 | 41.46786164 | 41.03016738 | 40.8627436 | 41.34841235 | 41.04925242 |
| Riverfort | 57.90409057 | 57.09022197 | 58.24413007 | 58.15882897 | 58.39380057 | 57.83258021 | 58.27815352 | 57.86529531 | 57.48099945 | 59.14915582 |
| Highfort | 43.79000147 | 43.5202791 | 43.14543721 | 43.06401008 | 43.19988988 | 43.39625862 | 43.44605585 | 43.82946035 | 43.43004855 | 42.86892377 |
| Shard | 42.31813163 | 42.03081695 | 41.93492252 | 42.65123918 | 42.43233052 | 42.37203534 | 41.42183811 | 42.07685666 | 41.82885983 | 41.70862767 |
| **User 3** | | | | | | | | | | |
| Citadel Gate | 50.91475482 | 50.84199551 | 51.52294446 | 51.52294446 | 50.99887849 | 51.71009757 | 50.86756459 | 50.96063951 | 51.42721804 | 50.95681979 |
| Overwatch | 52.09839552 | 50.87675797 | 50.97473081 | 50.87189943 | 51.06367711 | 51.17185514 | 51.27802184 | 51.43429161 | 50.62669428 | 50.81718467 |
| Sanctuary Bridge | 55.13669689 | 54.97193501 | 54.650198 | 54.12902466 | 54.23686238 | 54.79377622 | 54.30812589 | 54.32006883 | 54.92047267 | 54.25610181 |
| Riverfort | 46.62602018 | 46.86356443 | 45.70623271 | 46.78129406 | 46.79802513 | 46.47073649 | 46.56814043 | 46.08686852 | 46.48765456 | 46.44755837 |
| Highfort | 50.67971313 | 51.07261078 | 50.32329839 | 50.08485538 | 50.4730471 | 50.65460673 | 51.05413558 | 51.19923524 | 50.66065458 | 50.55118013 |
| Shard | 57.1224389 | 57.81936927 | 57.25027842 | 56.33884248 | 56.54956633 | 57.36646448 | 57.53500788 | 57.63655551 | 57.44845565 | 57.55604002 |
| **User 4** | | | | | | | | | | |
| Citadel Gate | 62.38216218 | 63.51321273 | 63.0261818 | 63.02617218 | 62.596495 | 63.22928692 | 63.46122894 | 62.55077449 | 63.28674571 | 62.2931538 |
| Overwatch | 68.94738822 | 66.99061012 | 67.62093266 | 66.300928 | 67.03625556 | 69.09251488 | 67.0991136 | 67.86367377 | 66.9146381 | 67.24953719 |
| Sanctuary Bridge | 56.8193987 | 56.44956344 | 55.76804922 | 56.11501482 | 55.34735526 | 55.7693179 | 55.53156745 | 56.32000948 | 55.4953603 | 55.93639568 |
| Riverfort | 53.93061285 | 54.10375328 | 53.3289032 | 53.63735831 | 52.88739541 | 53.60685982 | 53.3858846 | 53.78373789 | 53.37628967 | 53.49260773 |
| Highfort | 59.28862989 | 59.60039165 | 59.39625476 | 59.30732342 | 59.87294314 | 59.45590799 | 60.14417297 | 59.7819457 | 59.71298069 | 59.48424263 |
| Shard | 65.42241658 | 66.0785627 | 66.15237023 | 65.96594379 | 65.87415135 | 65.59681442 | 65.34214861 | 65.54557551 | 65.27732382 | 65.65423229 |
| **User 5** | | | | | | | | | | |
| Citadel Gate | 58.41758884 | 58.23371207 | 58.58905646 | 58.58904837 | 58.5522794 | 58.91444603 | 58.38476174 | 58.5862473 | 58.882542 | 58.37046195 |
| Overwatch | 61.70497678 | 61.24963198 | 61.17825829 | 61.54900114 | 61.30728018 | 61.13610723 | 61.91386475 | 61.57560705 | 60.94503797 | 61.50705641 |
| Sanctuary Bridge | 41.85160927 | 41.91855516 | 41.55634294 | 41.13564897 | 41.14790079 | 41.67987603 | 41.63623347 | 41.61055278 | 42.05270407 | 41.57489938 |
| Riverfort | 59.56526704 | 59.44572135 | 58.75505282 | 59.70499793 | 59.38983495 | 59.54737785 | 58.80459159 | 59.46200906 | 59.30532803 | 59.51384464 |
| Highfort | 52.69305975 | 53.69555113 | 53.50832913 | 52.83134538 | 53.19637945 | 52.96037926 | 53.27200942 | 53.23524434 | 52.88158161 | 53.400125 |
| Shard | 51.08134444 | 50.97480767 | 51.22879726 | 51.28137879 | 50.91810231 | 51.16202511 | 50.37601462 | 51.09011658 | 50.88516549 | 50.89893907 |
| **User 6** | | | | | | | | | | |
| Citadel Gate | 59.54824774 | 59.70607015 | 59.61353986 | 59.61353986 | 59.735552 | 59.8440343 | 59.34682733 | 59.40457277 | 60.39966032 | 58.69901867 |
| Overwatch | 59.6650494 | 59.69679211 | 59.04431543 | 59.09664914 | 59.22125574 | 59.50108341 | 59.46872443 | 59.31397479 | 59.17522663 | 59.54770048 |
| Sanctuary Bridge | 61.42589906 | 61.03886689 | 60.44334895 | 60.98101512 | 60.84334533 | 61.45133257 | 61.38434631 | 61.03912442 | 61.21212121 | 60.55760754 |
| Riverfort | 55.56060054 | 54.7220913 | 55.74702935 | 55.55523052 | 55.13291374 | 55.24981049 | 55.4111856 | 55.70666044 | 55.02031976 | 55.39599493 |
| Highfort | 61.35088843 | 61.89263583 | 61.93161605 | 61.77932316 | 61.86312699 | 61.54809383 | 62.22455574 | 61.38287044 | 61.67918588 | 62.25436528 |
| Shard | 65.30222398 | 66.09645459 | 65.66353417 | 65.52551966 | 65.71210649 | 65.47368312 | 65.55005249 | 65.65318985 | 65.33079731 | 65.5018437 |
| **User 7** | | | | | | | | | | |
| Citadel Gate | 54.23201627 | 52.56712065 | 54.86359451 | 54.86359451 | 53.73816767 | 53.84709428 | 53.9145364 | 54.42297205 | 54.41754005 | 54.32409147 |
| Overwatch | 52.98556934 | 52.09348557 | 52.39810589 | 53.13470171 | 52.09067821 | 52.55759438 | 52.79614535 | 52.24189799 | 51.67861175 | 53.52153673 |
| Sanctuary Bridge | 40.88423728 | 40.05773026 | 40.00123243 | 39.66688113 | 39.71364102 | 40.41532927 | 40.26968491 | 39.98832244 | 40.74648398 | 40.1427431 |
| Riverfort | 58.66543933 | 58.93894399 | 58.92943229 | 59.45893802 | 59.90535785 | 58.97163728 | 60.13036416 | 58.66028255 | 58.70617075 | 60.9620476 |
| Highfort | 47.54995177 | 48.14502663 | 47.71476856 | 47.06033979 | 47.67686327 | 48.04181796 | 47.4044915 | 47.9559775 | 47.57853305 | 47.8412569 |
| Shard | 40.95970471 | 40.3030196 | 40.74662662 | 40.49531687 | 40.62024677 | 41.07590993 | 40.44784063 | 40.95161109 | 41.07297337 | 40.32352307 |
| **User 8** | | | | | | | | | | |
| Citadel Gate | 56.4509114 | 56.18170678 | 55.94877583 | 55.94876815 | 55.98307338 | 56.68797185 | 56.21743757 | 56.02661536 | 55.85392369 | 56.39940944 |
| Overwatch | 56.86361478 | 56.14938141 | 55.51892834 | 55.75992686 | 55.72734818 | 56.07047185 | 56.3679834 | 56.00931202 | 55.57387166 | 56.56793625 |
| Sanctuary Bridge | 55.96345591 | 56.15943669 | 55.66314759 | 55.59467517 | 55.48498881 | 56.29520349 | 56.24927731 | 55.6125499 | 56.09181528 | 55.76069202 |
| Riverfort | 49.14792026 | 49.90612786 | 49.51187916 | 49.12827044 | 48.78317138 | 49.00930329 | 49.78617207 | 49.69509381 | 49.38826101 | 50.08830861 |
| Highfort | 57.91270001 | 57.76960028 | 57.58278525 | 58.20972103 | 57.85819315 | 57.68651158 | 57.82946523 | 57.53235966 | 58.06260014 | 58.31581211 |
| Shard | 47.45215382 | 47.6170985 | 47.91249035 | 48.28430144 | 48.07776012 | 48.34308084 | 46.62464442 | 47.79036059 | 47.3179253 | 47.00237102 |
| **User 9** | | | | | | | | | | |
| Citadel Gate | 45.91361562 | 45.34274975 | 46.13792832 | 46.13792832 | 45.96651223 | 45.70114695 | 45.90525535 | 45.89483566 | 45.29284417 | 46.13002078 |
| Overwatch | 47.9706398 | 47.06061153 | 47.26070329 | 47.9136 | 47.17665689 | 47.0733518 | 47.55298539 | 46.96618609 | 46.57862135 | 48.44574326 |
| Sanctuary Bridge | 41.19495609 | 41.79386069 | 40.60526864 | 40.97757157 | 40.83355837 | 40.83134725 | 41.02911618 | 40.81643761 | 41.03632014 | 41.00305285 |
| Riverfort | 50.63643613 | 50.34226913 | 51.15615705 | 50.78466842 | 50.86962085 | 50.33745828 | 51.45375429 | 50.47559543 | 50.69668032 | 51.11821446 |
| Highfort | 44.02366143 | 44.32979534 | 43.95485616 | 43.93171374 | 43.86412149 | 44.11953285 | 43.896055 | 44.19358001 | 44.14790898 | 43.7860865 |
| Shard | 45.12356235 | 45.52953234 | 45.22541867 | 45.56204049 | 45.605419 | 45.26495975 | 45.31518423 | 44.84117708 | 44.79948278 | 44.95867951 |
| **User 10** | | | | | | | | | | |
| Citadel Gate | 25.04683142 | 24.66195846 | 24.87909505 | 24.87909505 | 24.99682178 | 24.56198963 | 24.46561428 | 24.64617843 | 24.59739078 | 25.52221313 |
| Overwatch | 33.00851405 | 32.34354794 | 32.83988378 | 32.202752 | 32.46682463 | 33.10703347 | 33.13328061 | 33.21380289 | 32.29500208 | 32.35141354 |
| Sanctuary Bridge | 42.87297804 | 42.77271615 | 43.2465134 | 42.88613606 | 42.78601916 | 42.8407536 | 43.12004422 | 42.39560887 | 42.53856749 | 42.58672513 |
| Riverfort | 38.14942653 | 38.50518848 | 36.80294879 | 38.17819832 | 38.56668125 | 38.41749609 | 37.27328037 | 37.74732676 | 37.8846381 | 37.39152501 |
| Highfort | 27.06728202 | 27.43791532 | 27.53417368 | 28.4220087 | 28.12750617 | 27.72594455 | 27.96282522 | 27.48387443 | 27.81833035 | 28.19267193 |
| Shard | 32.10742444 | 31.81260342 | 32.65479572 | 33.19829897 | 33.06097351 | 32.32136411 | 31.06353072 | 31.95543598 | 31.94510834 | 31.61500605 |

Figure .15: Playthrough Prediction Training Accuracies (Method 1).

| (t, t', t'', κ₁, κ₂) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.460240964 | 0.39545611 | 0.378037866 | 0.343958692 | 0.367091222 |
| Overwatch | 0.376829268 | 0.324926829 | 0.376878049 | 0.361073171 | 0.300195122 |
| Sanctuary Bridge | 0.307270502 | 0.34374541 | 0.319608323 | 0.313537332 | 0.324504284 |
| Riverfort | 0.345054945 | 0.366637363 | 0.389846154 | 0.317758242 | 0.347296703 |
| Highfort | 0.327779006 | 0.325392265 | 0.325922652 | 0.283668508 | 0.284552486 |
| The Shard | 0.373269231 | 0.379903846 | 0.404951923 | 0.359663462 | 0.377307692 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.464027539 | 0.399380379 | 0.386299484 | 0.351876076 | 0.376798623 |
| Overwatch | 0.386 | 0.352731707 | 0.38897561 | 0.371902439 | 0.311560976 |
| Sanctuary Bridge | 0.318824969 | 0.374296206 | 0.338017136 | 0.330575275 | 0.342129743 |
| Riverfort | 0.354241758 | 0.390461538 | 0.394725275 | 0.328483516 | 0.359912088 |
| Highfort | 0.336353591 | 0.347535912 | 0.329767956 | 0.307933702 | 0.290961326 |
| The Shard | 0.382355769 | 0.403028846 | 0.411875 | 0.383413462 | 0.390432692 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.460240964 | 0.39545611 | 0.378037866 | 0.343958692 | 0.367091222 |
| Overwatch | 0.376829268 | 0.324926829 | 0.376878049 | 0.361073171 | 0.300195122 |
| Sanctuary Bridge | 0.307270502 | 0.34374541 | 0.319608323 | 0.313537332 | 0.324504284 |
| Riverfort | 0.345054945 | 0.366637363 | 0.389846154 | 0.317758242 | 0.347296703 |
| Highfort | 0.327779006 | 0.325392265 | 0.325922652 | 0.283668508 | 0.284552486 |
| The Shard | 0.373269231 | 0.379903846 | 0.404951923 | 0.359663462 | 0.377307692 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.498864028 | 0.436626506 | 0.410464716 | 0.383270224 | 0.393459552 |
| Overwatch | 0.439707317 | 0.429853659 | 0.413268293 | 0.401756098 | 0.334780488 |
| Sanctuary Bridge | 0.394222766 | 0.444014688 | 0.389522644 | 0.383059976 | 0.385312118 |
| Riverfort | 0.420747253 | 0.44778022 | 0.415912088 | 0.34967033 | 0.385450549 |
| Highfort | 0.393723757 | 0.40481768 | 0.350497238 | 0.360928177 | 0.317966851 |
| The Shard | 0.444134615 | 0.457740385 | 0.441730769 | 0.431538462 | 0.422644231 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.481445783 | 0.508364888 | 0.449913941 | 0.422650602 | 0.401858864 |
| Overwatch | 0.36702439 | 0.394146341 | 0.347756098 | 0.318341463 | 0.331365854 |
| Sanctuary Bridge | 0.345948592 | 0.330917993 | 0.365385557 | 0.302227662 | 0.362643819 |
| Riverfort | 0.39578022 | 0.407120879 | 0.316703297 | 0.368131868 | 0.358461538 |
| Highfort | 0.30638674 | 0.335248619 | 0.359160221 | 0.240353591 | 0.317348066 |
| The Shard | 0.455913462 | 0.474326923 | 0.416971154 | 0.369951923 | 0.369423077 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.4832358 | 0.514010327 | 0.459414802 | 0.428709122 | 0.408674699 |
| Overwatch | 0.378487805 | 0.419609756 | 0.349073171 | 0.324878049 | 0.35995122 |
| Sanctuary Bridge | 0.357356181 | 0.360195838 | 0.376891065 | 0.317307222 | 0.384577723 |
| Riverfort | 0.40443956 | 0.425494505 | 0.324659341 | 0.373406593 | 0.383472527 |
| Highfort | 0.322740331 | 0.34559116 | 0.36238674 | 0.252861878 | 0.338828729 |
| The Shard | 0.466634615 | 0.483798077 | 0.422644231 | 0.380721154 | 0.394759615 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.481445783 | 0.508364888 | 0.449913941 | 0.422650602 | 0.401858864 |
| Overwatch | 0.36702439 | 0.394146341 | 0.347756098 | 0.318341463 | 0.331365854 |
| Sanctuary Bridge | 0.345948592 | 0.330917993 | 0.365385557 | 0.302227662 | 0.362643819 |
| Riverfort | 0.39578022 | 0.407120879 | 0.316703297 | 0.368131868 | 0.358461538 |
| Highfort | 0.30638674 | 0.335248619 | 0.359160221 | 0.240353591 | 0.317348066 |
| The Shard | 0.455913462 | 0.474326923 | 0.416971154 | 0.369951923 | 0.369423077 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.489225473 | 0.527160069 | 0.491566265 | 0.442134251 | 0.435731497 |
| Overwatch | 0.402585366 | 0.454634146 | 0.389609756 | 0.340390244 | 0.428682927 |
| Sanctuary Bridge | 0.391627907 | 0.418898409 | 0.425556916 | 0.348151775 | 0.444455324 |
| Riverfort | 0.427604396 | 0.457142857 | 0.365010989 | 0.382769231 | 0.433142857 |
| Highfort | 0.359027624 | 0.386563536 | 0.394342541 | 0.277834254 | 0.396729282 |
| The Shard | 0.497548077 | 0.503605769 | 0.457596154 | 0.405048077 | 0.450384615 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.51524957 | 0.374595525 | 0.350774527 | 0.366678141 | 0.314836489 |
| Overwatch | 0.40004878 | 0.319853659 | 0.339268293 | 0.325756098 | 0.334634146 |
| Sanctuary Bridge | 0.29625459 | 0.30247246 | 0.324700122 | 0.341346389 | 0.322301102 |
| Riverfort | 0.390945055 | 0.330813187 | 0.358241758 | 0.356747253 | 0.334021978 |
| Highfort | 0.299491713 | 0.36 | 0.343027624 | 0.339049724 | 0.316022099 |
| The Shard | 0.400673077 | 0.379086538 | 0.399423077 | 0.416730769 | 0.351875 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.51848537 | 0.38939759 | 0.376179002 | 0.37858864 | 0.334802065 |
| Overwatch | 0.412487805 | 0.337609756 | 0.358390244 | 0.34497561 | 0.354585366 |
| Sanctuary Bridge | 0.31373317 | 0.324700122 | 0.344430845 | 0.366511628 | 0.342031824 |
| Riverfort | 0.406285714 | 0.343296703 | 0.375120879 | 0.375164835 | 0.361802198 |
| Highfort | 0.315270718 | 0.370077348 | 0.362519337 | 0.354209945 | 0.339668508 |
| The Shard | 0.401923077 | 0.389711538 | 0.422451923 | 0.435865385 | 0.368509615 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.51524957 | 0.374595525 | 0.350774527 | 0.366678141 | 0.314836489 |
| Overwatch | 0.40004878 | 0.319853659 | 0.339268293 | 0.325756098 | 0.334634146 |
| Sanctuary Bridge | 0.29625459 | 0.30247246 | 0.324700122 | 0.341346389 | 0.322301102 |
| Riverfort | 0.390945055 | 0.330813187 | 0.358241758 | 0.356747253 | 0.334021978 |
| Highfort | 0.299491713 | 0.36 | 0.343027624 | 0.339049724 | 0.316022099 |
| The Shard | 0.400673077 | 0.379086538 | 0.399423077 | 0.416730769 | 0.351875 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.537142857 | 0.418932874 | 0.426987952 | 0.416179002 | 0.387607573 |
| Overwatch | 0.441073171 | 0.386390244 | 0.42702439 | 0.410926829 | 0.432634146 |
| Sanctuary Bridge | 0.362350061 | 0.383353733 | 0.400979192 | 0.41747858 | 0.410820073 |
| Riverfort | 0.428395604 | 0.397406593 | 0.42021978 | 0.418417582 | 0.421274725 |
| Highfort | 0.34921547 | 0.411668508 | 0.406232044 | 0.402033149 | 0.392265193 |
| The Shard | 0.425432692 | 0.427403846 | 0.465384615 | 0.472355769 | 0.421538462 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Metric 1** | | | | |
| Citadel Gate | 0.377280551 | 0.406609294 | 0.383614458 | 0.405480972 |
| Overwatch | 0.324634146 | 0.286878049 | 0.301268293 | 0.340604336 |
| Sanctuary Bridge | 0.327637699 | 0.338017136 | 0.334932681 | 0.327969536 |
| Riverfort | 0.353846154 | 0.361274725 | 0.363824176 | 0.359028083 |
| Highfort | 0.332552486 | 0.324640884 | 0.326364641 | 0.319275629 |
| The Shard | 0.346826923 | 0.371682692 | 0.393846154 | 0.391212607 |
| **Metric 2** | | | | |
| Citadel Gate | 0.40626506 | 0.420240964 | 0.406471601 | 0.416825397 |
| Overwatch | 0.356780488 | 0.308439024 | 0.330878049 | 0.358184282 |
| Sanctuary Bridge | 0.352117503 | 0.368518972 | 0.360979192 | 0.348510812 |
| Riverfort | 0.385230769 | 0.383032967 | 0.38189011 | 0.375062271 |
| Highfort | 0.367116022 | 0.34439779 | 0.349790055 | 0.335445058 |
| The Shard | 0.376826923 | 0.394423077 | 0.414519231 | 0.406883013 |
| **Metric 3** | | | | |
| Citadel Gate | 0.377280551 | 0.406609294 | 0.383614458 | 0.405480972 |
| Overwatch | 0.324634146 | 0.286878049 | 0.301268293 | 0.340604336 |
| Sanctuary Bridge | 0.327637699 | 0.338017136 | 0.334932681 | 0.327969536 |
| Riverfort | 0.353846154 | 0.361274725 | 0.363824176 | 0.359028083 |
| Highfort | 0.332552486 | 0.324640884 | 0.326364641 | 0.319275629 |
| The Shard | 0.346826923 | 0.371682692 | 0.393846154 | 0.391212607 |
| **Metric 4** | | | | |
| Citadel Gate | 0.459690189 | 0.472289157 | 0.460585198 | 0.449328744 |
| Overwatch | 0.431560976 | 0.389512195 | 0.41497561 | 0.409409214 |
| Sanctuary Bridge | 0.414247246 | 0.435152999 | 0.430991432 | 0.404455324 |
| Riverfort | 0.44843956 | 0.439252747 | 0.439868132 | 0.416583639 |
| Highfort | 0.426298343 | 0.41038674 | 0.406541436 | 0.380392879 |
| The Shard | 0.429903846 | 0.454086538 | 0.467740385 | 0.448656517 |

Figure .16: Playthrough Prediction Test Accuracies (Method 1).

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.356734694 | 0.311734694 | 0.344795918 | 0.264489796 | 0.293877551 |
| Overwatch | 0.316601307 | 0.305620915 | 0.379869281 | 0.264052288 | 0.272156863 |
| Sanctuary Bridge | 0.366923077 | 0.433333333 | 0.326666667 | 0.374358974 | 0.355384615 |
| Riverfort | 0.352380952 | 0.342857143 | 0.346031746 | 0.291428571 | 0.302857143 |
| Highfort | 0.361764706 | 0.354705882 | 0.371176471 | 0.342941176 | 0.292941176 |
| The Shard | 0.278014184 | 0.287092199 | 0.325106383 | 0.247659574 | 0.294184397 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.370204082 | 0.318265306 | 0.341326531 | 0.268367347 | 0.295102041 |
| Overwatch | 0.338039216 | 0.30875817 | 0.387189542 | 0.275294118 | 0.282614379 |
| Sanctuary Bridge | 0.378717949 | 0.461794872 | 0.332564103 | 0.376410256 | 0.372051282 |
| Riverfort | 0.365079365 | 0.378412698 | 0.344761905 | 0.295873016 | 0.316825397 |
| Highfort | 0.374705882 | 0.380588235 | 0.384705882 | 0.361764706 | 0.33 |
| The Shard | 0.285673759 | 0.304397163 | 0.312056738 | 0.267234043 | 0.295602837 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.356734694 | 0.311734694 | 0.344795918 | 0.264489796 | 0.293877551 |
| Overwatch | 0.316601307 | 0.305620915 | 0.379869281 | 0.264052288 | 0.272156863 |
| Sanctuary Bridge | 0.366923077 | 0.433333333 | 0.326666667 | 0.374358974 | 0.355384615 |
| Riverfort | 0.352380952 | 0.342857143 | 0.346031746 | 0.291428571 | 0.302857143 |
| Highfort | 0.361764706 | 0.354705882 | 0.371176471 | 0.342941176 | 0.292941176 |
| The Shard | 0.278014184 | 0.287092199 | 0.325106383 | 0.247659574 | 0.294184397 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.413265306 | 0.366020408 | 0.350204082 | 0.291632653 | 0.319183673 |
| Overwatch | 0.431372549 | 0.407843137 | 0.41254902 | 0.312679739 | 0.320784314 |
| Sanctuary Bridge | 0.422307692 | 0.507692308 | 0.368974359 | 0.403333333 | 0.395384615 |
| Riverfort | 0.42984127 | 0.445714286 | 0.403174603 | 0.311746032 | 0.351111111 |
| Highfort | 0.43 | 0.452941176 | 0.415294118 | 0.394117647 | 0.353529412 |
| The Shard | 0.323971631 | 0.330496454 | 0.352907801 | 0.319148936 | 0.330780142 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.411938776 | 0.441938776 | 0.340918367 | 0.367755102 | 0.317142857 |
| Overwatch | 0.361045752 | 0.380653595 | 0.292810458 | 0.315294118 | 0.339346405 |
| Sanctuary Bridge | 0.432307692 | 0.435897436 | 0.395897436 | 0.346410256 | 0.451282051 |
| Riverfort | 0.366349206 | 0.393015873 | 0.316825397 | 0.380952381 | 0.344126984 |
| Highfort | 0.368235294 | 0.382941176 | 0.384117647 | 0.269411765 | 0.332352941 |
| The Shard | 0.345248227 | 0.39035461 | 0.269787234 | 0.303829787 | 0.278865248 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.411938776 | 0.443571429 | 0.343367347 | 0.370510204 | 0.315408163 |
| Overwatch | 0.386666667 | 0.390588235 | 0.293071895 | 0.324705882 | 0.350588235 |
| Sanctuary Bridge | 0.441794872 | 0.46025641 | 0.402820513 | 0.354102564 | 0.454102564 |
| Riverfort | 0.371428571 | 0.45015873 | 0.322539683 | 0.38984127 | 0.367619048 |
| Highfort | 0.38 | 0.391764706 | 0.404705882 | 0.286470588 | 0.375882353 |
| The Shard | 0.345531915 | 0.400567376 | 0.262411348 | 0.307234043 | 0.298156028 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.411938776 | 0.441938776 | 0.340918367 | 0.367755102 | 0.317142857 |
| Overwatch | 0.361045752 | 0.380653595 | 0.292810458 | 0.315294118 | 0.339346405 |
| Sanctuary Bridge | 0.432307692 | 0.435897436 | 0.395897436 | 0.346410256 | 0.451282051 |
| Riverfort | 0.366349206 | 0.393015873 | 0.316825397 | 0.380952381 | 0.344126984 |
| Highfort | 0.368235294 | 0.382941176 | 0.384117647 | 0.269411765 | 0.332352941 |
| The Shard | 0.345248227 | 0.39035461 | 0.269787234 | 0.303829787 | 0.278865248 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.416632653 | 0.457244898 | 0.378571429 | 0.382142857 | 0.355306122 |
| Overwatch | 0.437385621 | 0.438954248 | 0.34248366 | 0.352941176 | 0.418039216 |
| Sanctuary Bridge | 0.468974359 | 0.496666667 | 0.448461538 | 0.365128205 | 0.497179487 |
| Riverfort | 0.415873016 | 0.487619048 | 0.371428571 | 0.406349206 | 0.441269841 |
| Highfort | 0.401176471 | 0.457647059 | 0.437647059 | 0.302352941 | 0.419411765 |
| The Shard | 0.390070922 | 0.404822695 | 0.299007092 | 0.333333333 | 0.330496454 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Metric 1** | | | | | |
| Citadel Gate | 0.444897959 | 0.304183673 | 0.320306122 | 0.316122449 | 0.240816327 |
| Overwatch | 0.396339869 | 0.271633987 | 0.303006536 | 0.288888889 | 0.304052288 |
| Sanctuary Bridge | 0.358974359 | 0.390512821 | 0.394871795 | 0.417179487 | 0.355897436 |
| Riverfort | 0.355555556 | 0.319365079 | 0.356825397 | 0.365079365 | 0.352380952 |
| Highfort | 0.397647059 | 0.414117647 | 0.44 | 0.405882353 | 0.363529412 |
| The Shard | 0.297021277 | 0.284539007 | 0.313475177 | 0.347801418 | 0.221560284 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.449897959 | 0.315204082 | 0.329081633 | 0.324285714 | 0.256836735 |
| Overwatch | 0.396601307 | 0.279477124 | 0.328366013 | 0.31503268 | 0.334379085 |
| Sanctuary Bridge | 0.37025641 | 0.397179487 | 0.403589744 | 0.425128205 | 0.384871795 |
| Riverfort | 0.380952381 | 0.326984127 | 0.355555556 | 0.365714286 | 0.377777778 |
| Highfort | 0.376470588 | 0.452941176 | 0.470588235 | 0.464117647 | 0.387058824 |
| The Shard | 0.300992908 | 0.270638298 | 0.308652482 | 0.342695035 | 0.227234043 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.444897959 | 0.304183673 | 0.320306122 | 0.316122449 | 0.240816327 |
| Overwatch | 0.396339869 | 0.271633987 | 0.303006536 | 0.288888889 | 0.304052288 |
| Sanctuary Bridge | 0.358974359 | 0.390512821 | 0.394871795 | 0.417179487 | 0.355897436 |
| Riverfort | 0.355555556 | 0.319365079 | 0.356825397 | 0.365079365 | 0.352380952 |
| Highfort | 0.397647059 | 0.414117647 | 0.44 | 0.405882353 | 0.363529412 |
| The Shard | 0.297021277 | 0.284539007 | 0.313475177 | 0.347801418 | 0.221560284 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.467755102 | 0.356836735 | 0.367755102 | 0.356020408 | 0.306530612 |
| Overwatch | 0.432941176 | 0.331764706 | 0.392156863 | 0.362091503 | 0.403660131 |
| Sanctuary Bridge | 0.37025641 | 0.421538462 | 0.434615385 | 0.445897436 | 0.455128205 |
| Riverfort | 0.420952381 | 0.376507937 | 0.418412698 | 0.431746032 | 0.455873016 |
| Highfort | 0.396470588 | 0.469411765 | 0.532352941 | 0.486470588 | 0.443529412 |
| The Shard | 0.31858156 | 0.310921986 | 0.334184397 | 0.366524823 | 0.270921986 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Metric 1** | | | | |
| Citadel Gate | 0.308163265 | 0.335 | 0.316530612 | 0.335408163 |
| Overwatch | 0.303006536 | 0.322875817 | 0.275555556 | 0.316267248 |
| Sanctuary Bridge | 0.328205128 | 0.423076923 | 0.420512821 | 0.389316239 |
| Riverfort | 0.365714286 | 0.346031746 | 0.341587302 | 0.346631393 |
| Highfort | 0.371764706 | 0.398235294 | 0.369411765 | 0.367843137 |
| The Shard | 0.226950355 | 0.299574468 | 0.315460993 | 0.295918046 |
| **Metric 2** | | | | |
| Citadel Gate | 0.329591837 | 0.346122449 | 0.329591837 | 0.342148526 |
| Overwatch | 0.317908497 | 0.329411765 | 0.317647059 | 0.330907771 |
| Sanctuary Bridge | 0.362820513 | 0.442307692 | 0.436410256 | 0.403176638 |
| Riverfort | 0.365079365 | 0.363809524 | 0.375238095 | 0.361869489 |
| Highfort | 0.403529412 | 0.408823529 | 0.405882353 | 0.391111111 |
| The Shard | 0.247375887 | 0.302695035 | 0.310921986 | 0.299448385 |
| **Metric 3** | | | | |
| Citadel Gate | 0.308163265 | 0.335 | 0.316530612 | 0.335408163 |
| Overwatch | 0.303006536 | 0.322875817 | 0.275555556 | 0.316267248 |
| Sanctuary Bridge | 0.328205128 | 0.423076923 | 0.420512821 | 0.389316239 |
| Riverfort | 0.365714286 | 0.346031746 | 0.341587302 | 0.346631393 |
| Highfort | 0.371764706 | 0.398235294 | 0.369411765 | 0.367843137 |
| The Shard | 0.226950355 | 0.299574468 | 0.315460993 | 0.295918046 |
| **Metric 4** | | | | |
| Citadel Gate | 0.375408163 | 0.39244898 | 0.380612245 | 0.374087302 |
| Overwatch | 0.376470588 | 0.409150327 | 0.404444444 | 0.388206245 |
| Sanctuary Bridge | 0.412307692 | 0.480512821 | 0.483333333 | 0.437649573 |
| Riverfort | 0.42031746 | 0.464761905 | 0.476190476 | 0.418271605 |
| Highfort | 0.460588235 | 0.474705882 | 0.449411765 | 0.432058824 |
| The Shard | 0.283404255 | 0.335035461 | 0.342411348 | 0.332056738 |

Figure .17: Playthrough Prediction Training Accuracies (Method 2 — Multinomial).

| (t, t', t'', κ1, κ2) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.581709145 | 0.581709145 |
| Overwatch | 0.54884742 | 0.549342105 | 0.549342105 | 0.549835706 | 0.549835706 |
| Sanctuary Bridge | 0.531701891 | 0.532222222 | 0.532222222 | 0.532150776 | 0.532150776 |
| Riverfort | 0.55 | 0.550440744 | 0.550440744 | 0.55078125 | 0.55078125 |
| Highfort | 0.542120912 | 0.542574257 | 0.542574257 | 0.542574257 | 0.542574257 |
| The Shard | 0.571888412 | 0.572347267 | 0.572347267 | 0.572649573 | 0.572649573 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.59939759 | 0.582831325 | 0.611445783 | 0.625187406 | 0.620689655 |
| Overwatch | 0.568605928 | 0.559210526 | 0.57127193 | 0.579408543 | 0.583789704 |
| Sanctuary Bridge | 0.540600667 | 0.537777778 | 0.563333333 | 0.575388027 | 0.586474501 |
| Riverfort | 0.55 | 0.55337904 | 0.576885406 | 0.583984375 | 0.580078125 |
| Highfort | 0.545094153 | 0.545544554 | 0.57029703 | 0.593287266 | 0.57946693 |
| The Shard | 0.575107296 | 0.578778135 | 0.589496249 | 0.608974359 | 0.590811966 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.596385542 | 0.582831325 | 0.605421687 | 0.614692654 | 0.620689655 |
| Overwatch | 0.565312843 | 0.550438596 | 0.567982456 | 0.576122673 | 0.575027382 |
| Sanctuary Bridge | 0.536151279 | 0.537777778 | 0.558888889 | 0.563192905 | 0.569844789 |
| Riverfort | 0.55 | 0.55337904 | 0.569049951 | 0.579101563 | 0.576171875 |
| Highfort | 0.542120912 | 0.542574257 | 0.562376238 | 0.586377098 | 0.567620928 |
| The Shard | 0.576180258 | 0.57449089 | 0.592711683 | 0.603632479 | 0.586538462 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.59939759 | 0.582831325 | 0.611445783 | 0.623688156 | 0.620689655 |
| Overwatch | 0.568605928 | 0.559210526 | 0.57127193 | 0.578313253 | 0.583789704 |
| Sanctuary Bridge | 0.540600667 | 0.537777778 | 0.563333333 | 0.575388027 | 0.586474501 |
| Riverfort | 0.55 | 0.55337904 | 0.576885406 | 0.583984375 | 0.580078125 |
| Highfort | 0.545094153 | 0.545544554 | 0.569306931 | 0.593287266 | 0.578479763 |
| The Shard | 0.575107296 | 0.578778135 | 0.589496249 | 0.608974359 | 0.590811966 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.551204819 | 0.56626506 | 0.561746988 | 0.505247376 | 0.538230885 |
| Overwatch | 0.563117453 | 0.543859649 | 0.54495614 | 0.519167579 | 0.514786418 |
| Sanctuary Bridge | 0.536151279 | 0.528888889 | 0.537777778 | 0.506651885 | 0.521064302 |
| Riverfort | 0.55 | 0.539666993 | 0.536728697 | 0.51953125 | 0.526367188 |
| Highfort | 0.545094153 | 0.535643564 | 0.534653465 | 0.502467917 | 0.514313919 |
| The Shard | 0.572961373 | 0.56698821 | 0.573419078 | 0.518162393 | 0.536324786 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581709145 | 0.581325301 | 0.581709145 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549835706 | 0.54884742 | 0.549835706 | 0.549342105 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.531701891 | 0.532150776 | 0.532222222 |
| Riverfort | 0.550440744 | 0.55078125 | 0.55 | 0.55078125 | 0.550440744 |
| Highfort | 0.542574257 | 0.542574257 | 0.542120912 | 0.542941757 | 0.542574257 |
| The Shard | 0.572347267 | 0.572649573 | 0.571888412 | 0.572649573 | 0.572347267 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.581325301 | 0.581709145 | 0.615963855 | 0.581709145 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549835706 | 0.571899012 | 0.553121577 | 0.558114035 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.546162403 | 0.534368071 | 0.538888889 |
| Riverfort | 0.550440744 | 0.55078125 | 0.573529412 | 0.551757813 | 0.556317336 |
| Highfort | 0.542574257 | 0.542941757 | 0.567888999 | 0.544916091 | 0.548514851 |
| The Shard | 0.572347267 | 0.572649573 | 0.571888412 | 0.575854701 | 0.57449089 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.581325301 | 0.581709145 | 0.620481928 | 0.586206897 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549835706 | 0.572996707 | 0.553121577 | 0.558114035 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.548387097 | 0.534368071 | 0.534444444 |
| Riverfort | 0.550440744 | 0.55078125 | 0.570588235 | 0.553710938 | 0.555337904 |
| Highfort | 0.542574257 | 0.542941757 | 0.57086224 | 0.544916091 | 0.545544554 |
| The Shard | 0.572347267 | 0.572649573 | 0.589055794 | 0.576923077 | 0.57449089 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.581325301 | 0.581709145 | 0.615963855 | 0.581709145 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549835706 | 0.571899012 | 0.553121577 | 0.558114035 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.546162403 | 0.534368071 | 0.538888889 |
| Riverfort | 0.550440744 | 0.55078125 | 0.573529412 | 0.551757813 | 0.555337904 |
| Highfort | 0.542574257 | 0.542941757 | 0.566897919 | 0.544916091 | 0.548514851 |
| The Shard | 0.572347267 | 0.572649573 | 0.589055794 | 0.575854701 | 0.57449089 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.581325301 | 0.572713643 | 0.516566265 | 0.563718141 | 0.55873494 |
| Overwatch | 0.549342105 | 0.547645126 | 0.515916575 | 0.538882804 | 0.54495614 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.515016685 | 0.521064302 | 0.523333333 |
| Riverfort | 0.550440744 | 0.547851563 | 0.512745098 | 0.541015625 | 0.540646425 |
| Highfort | 0.542574257 | 0.542941757 | 0.508424182 | 0.529121422 | 0.533663366 |
| The Shard | 0.572347267 | 0.572649573 | 0.539699571 | 0.561965812 | 0.56698821 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.581325301 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549342105 | 0.549342105 | 0.549342105 | 0.549342105 |
| Sanctuary Bridge | 0.532222222 | 0.532222222 | 0.532222222 | 0.532222222 | 0.532222222 |
| Riverfort | 0.550440744 | 0.550440744 | 0.550440744 | 0.550440744 | 0.550440744 |
| Highfort | 0.542574257 | 0.542574257 | 0.542574257 | 0.542574257 | 0.542574257 |
| The Shard | 0.572347267 | 0.572347267 | 0.572347267 | 0.572347267 | 0.572347267 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.581325301 | 0.582831325 | 0.584337349 | 0.584337349 | 0.588855422 |
| Overwatch | 0.549342105 | 0.552631579 | 0.551535088 | 0.554824561 | 0.558114035 |
| Sanctuary Bridge | 0.532222222 | 0.535555556 | 0.536666667 | 0.536666667 | 0.534444444 |
| Riverfort | 0.550440744 | 0.55337904 | 0.554358472 | 0.554358472 | 0.560235064 |
| Highfort | 0.542574257 | 0.543564356 | 0.545544554 | 0.545544554 | 0.551485149 |
| The Shard | 0.572347267 | 0.573419078 | 0.572347267 | 0.573419078 | 0.580921758 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.581325301 | 0.582831325 | 0.584337349 | 0.584337349 | 0.588855422 |
| Overwatch | 0.549342105 | 0.552631579 | 0.552631579 | 0.554824561 | 0.557017544 |
| Sanctuary Bridge | 0.532222222 | 0.527777778 | 0.533333333 | 0.535555556 | 0.53 |
| Riverfort | 0.550440744 | 0.550440744 | 0.555337904 | 0.554358472 | 0.547502449 |
| Highfort | 0.542574257 | 0.543564356 | 0.541584158 | 0.545544554 | 0.544554455 |
| The Shard | 0.572347267 | 0.573419078 | 0.572347267 | 0.573419078 | 0.577706324 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.581325301 | 0.582831325 | 0.584337349 | 0.584337349 | 0.588855422 |
| Overwatch | 0.549342105 | 0.552631579 | 0.551535088 | 0.554824561 | 0.558114035 |
| Sanctuary Bridge | 0.532222222 | 0.535555556 | 0.536666667 | 0.536666667 | 0.534444444 |
| Riverfort | 0.550440744 | 0.55337904 | 0.554358472 | 0.554358472 | 0.560235064 |
| Highfort | 0.542574257 | 0.543564356 | 0.545544554 | 0.545544554 | 0.551485149 |
| The Shard | 0.572347267 | 0.573419078 | 0.572347267 | 0.573419078 | 0.580921758 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.581325301 | 0.564759036 | 0.557228916 | 0.555722892 | 0.557228916 |
| Overwatch | 0.549342105 | 0.55372807 | 0.551535088 | 0.557017544 | 0.550438596 |
| Sanctuary Bridge | 0.532222222 | 0.536666667 | 0.536666667 | 0.535555556 | 0.528888889 |
| Riverfort | 0.550440744 | 0.551420176 | 0.556317336 | 0.555337904 | 0.545543585 |
| Highfort | 0.542574257 | 0.543564356 | 0.545544554 | 0.547524752 | 0.538613861 |
| The Shard | 0.572347267 | 0.56698821 | 0.570203644 | 0.571275456 | 0.569131833 |

| (t, t', t'', κ$_1$, κ$_2$) | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Baseline** | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.5814106 |
| Overwatch | 0.549342105 | 0.549342105 | 0.549342105 | 0.549396829 |
| Sanctuary Bridge | 0.532222222 | 0.532222222 | 0.532222222 | 0.532148531 |
| Riverfort | 0.550440744 | 0.550440744 | 0.550440744 | 0.550467441 |
| Highfort | 0.542574257 | 0.542574257 | 0.542574257 | 0.542544302 |
| The Shard | 0.572347267 | 0.572347267 | 0.572347267 | 0.572363462 |
| **Metric 1** | | | | |
| Citadel Gate | 0.593373494 | 0.596385542 | 0.585843373 | 0.593270759 |
| Overwatch | 0.567982456 | 0.559210526 | 0.551535088 | 0.560543028 |
| Sanctuary Bridge | 0.542222222 | 0.538888889 | 0.54 | 0.543557407 |
| Riverfort | 0.568070519 | 0.556317336 | 0.556317336 | 0.560035027 |
| Highfort | 0.556435644 | 0.552475248 | 0.547524752 | 0.553648578 |
| The Shard | 0.59056806 | 0.577706324 | 0.57449089 | 0.579201032 |
| **Metric 2** | | | | |
| Citadel Gate | 0.588855422 | 0.590361446 | 0.587349398 | 0.59218458 |
| Overwatch | 0.564692982 | 0.555921053 | 0.551535088 | 0.558716143 |
| Sanctuary Bridge | 0.543333333 | 0.536666667 | 0.538888889 | 0.540289224 |
| Riverfort | 0.563173359 | 0.554358472 | 0.554358472 | 0.557696229 |
| Highfort | 0.556435644 | 0.545544554 | 0.543564356 | 0.55062637 |
| The Shard | 0.586280815 | 0.575562701 | 0.571275456 | 0.578965464 |
| **Metric 3** | | | | |
| Citadel Gate | 0.593373494 | 0.596385542 | 0.585843373 | 0.593187468 |
| Overwatch | 0.567982456 | 0.559210526 | 0.551535088 | 0.560482179 |
| Sanctuary Bridge | 0.542222222 | 0.538888889 | 0.54 | 0.543557407 |
| Riverfort | 0.567091087 | 0.557296768 | 0.556317336 | 0.559980614 |
| Highfort | 0.556435644 | 0.552475248 | 0.547524752 | 0.55348367 |
| The Shard | 0.591639871 | 0.577706324 | 0.57449089 | 0.58021432 |
| **Metric 4** | | | | |
| Citadel Gate | 0.567771084 | 0.549698795 | 0.546686747 | 0.555343061 |
| Overwatch | 0.535087719 | 0.539473684 | 0.548245614 | 0.542638801 |
| Sanctuary Bridge | 0.537777778 | 0.524444444 | 0.527777778 | 0.528573414 |
| Riverfort | 0.538687561 | 0.541625857 | 0.542605289 | 0.541498446 |
| Highfort | 0.534653465 | 0.536633663 | 0.535643564 | 0.534091693 |
| The Shard | 0.560557342 | 0.564844587 | 0.559485531 | 0.562018897 |

Figure .18: Playthrough Prediction Test Accuracies (Method 2 — Multinomial).

| (t, t', t'', κ$_1$, κ$_2$) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.242346939 | 0.232142857 | 0.227040816 | 0.232142857 | 0.221938776 |
| Overwatch | 0.254901961 | 0.241830065 | 0.222222222 | 0.222222222 | 0.215686275 |
| Sanctuary Bridge | 0.307692308 | 0.301282051 | 0.288461538 | 0.301282051 | 0.294871795 |
| Riverfort | 0.603174603 | 0.555555556 | 0.523809524 | 0.53968254 | 0.476190476 |
| Highfort | 0.485294118 | 0.5 | 0.441176471 | 0.455882353 | 0.426470588 |
| The Shard | 0.29787234 | 0.304964539 | 0.283687943 | 0.319148936 | 0.290780142 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.237244898 | 0.234693878 | 0.232142857 | 0.232142857 | 0.224489796 |
| Overwatch | 0.254901961 | 0.248366013 | 0.22875817 | 0.215686275 | 0.215686275 |
| Sanctuary Bridge | 0.307692308 | 0.301282051 | 0.294871795 | 0.288461538 | 0.275641026 |
| Riverfort | 0.603174603 | 0.587301587 | 0.555555556 | 0.53968254 | 0.492063492 |
| Highfort | 0.5 | 0.485294118 | 0.441176471 | 0.470588235 | 0.441176471 |
| The Shard | 0.290780142 | 0.304964539 | 0.283687943 | 0.312056738 | 0.283687943 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.242346939 | 0.232142857 | 0.227040816 | 0.232142857 | 0.221938776 |
| Overwatch | 0.254901961 | 0.241830065 | 0.222222222 | 0.222222222 | 0.215686275 |
| Sanctuary Bridge | 0.307692308 | 0.301282051 | 0.288461538 | 0.301282051 | 0.294871795 |
| Riverfort | 0.603174603 | 0.555555556 | 0.523809524 | 0.53968254 | 0.476190476 |
| Highfort | 0.485294118 | 0.5 | 0.441176471 | 0.455882353 | 0.426470588 |
| The Shard | 0.29787234 | 0.304964539 | 0.283687943 | 0.319148936 | 0.290780142 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.216836735 | 0.214285714 | 0.211734694 | 0.211734694 | 0.198979592 |
| Overwatch | 0.189542484 | 0.169934641 | 0.176470588 | 0.183006536 | 0.169934641 |
| Sanctuary Bridge | 0.269230769 | 0.243589744 | 0.230769231 | 0.230769231 | 0.25 |
| Riverfort | 0.444444444 | 0.428571429 | 0.349206349 | 0.412698413 | 0.333333333 |
| Highfort | 0.382352941 | 0.367647059 | 0.279411765 | 0.367647059 | 0.279411765 |
| The Shard | 0.262411348 | 0.24822695 | 0.219858156 | 0.255319149 | 0.234042553 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.232142857 | 0.232142857 | 0.214285714 | 0.234693878 | 0.232142857 |
| Overwatch | 0.248366013 | 0.248366013 | 0.235294118 | 0.241830065 | 0.241830065 |
| Sanctuary Bridge | 0.307692308 | 0.307692308 | 0.230769231 | 0.314102564 | 0.307692308 |
| Riverfort | 0.587301587 | 0.587301587 | 0.476190476 | 0.603174603 | 0.571428571 |
| Highfort | 0.485294118 | 0.485294118 | 0.25 | 0.485294118 | 0.5 |
| The Shard | 0.304964539 | 0.304964539 | 0.276595745 | 0.29787234 | 0.304964539 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.234693878 | 0.234693878 | 0.219387755 | 0.234693878 | 0.234693878 |
| Overwatch | 0.248366013 | 0.248366013 | 0.241830065 | 0.241830065 | 0.248366013 |
| Sanctuary Bridge | 0.307692308 | 0.307692308 | 0.230769231 | 0.307692308 | 0.301282051 |
| Riverfort | 0.603174603 | 0.603174603 | 0.476190476 | 0.603174603 | 0.603174603 |
| Highfort | 0.485294118 | 0.485294118 | 0.264705882 | 0.485294118 | 0.5 |
| The Shard | 0.304964539 | 0.304964539 | 0.269503546 | 0.29787234 | 0.304964539 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.232142857 | 0.232142857 | 0.214285714 | 0.234693878 | 0.232142857 |
| Overwatch | 0.248366013 | 0.248366013 | 0.235294118 | 0.241830065 | 0.241830065 |
| Sanctuary Bridge | 0.307692308 | 0.307692308 | 0.230769231 | 0.314102564 | 0.307692308 |
| Riverfort | 0.587301587 | 0.587301587 | 0.476190476 | 0.603174603 | 0.571428571 |
| Highfort | 0.485294118 | 0.485294118 | 0.25 | 0.485294118 | 0.5 |
| The Shard | 0.304964539 | 0.304964539 | 0.276595745 | 0.29787234 | 0.304964539 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.206632653 | 0.219387755 | 0.198979592 | 0.209183673 | 0.204081633 |
| Overwatch | 0.176470588 | 0.248366013 | 0.169934641 | 0.169934641 | 0.169934641 |
| Sanctuary Bridge | 0.256410256 | 0.211538462 | 0.192307692 | 0.243589744 | 0.25 |
| Riverfort | 0.396825397 | 0.46031746 | 0.333333333 | 0.476190476 | 0.492063492 |
| Highfort | 0.338235294 | 0.382352941 | 0.205882353 | 0.338235294 | 0.308823529 |
| The Shard | 0.262411348 | 0.255319149 | 0.19858156 | 0.24822695 | 0.241134752 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.232142857 | 0.232142857 | 0.232142857 | 0.232142857 | 0.227040816 |
| Overwatch | 0.248366013 | 0.248366013 | 0.248366013 | 0.248366013 | 0.235294118 |
| Sanctuary Bridge | 0.307692308 | 0.307692308 | 0.294871795 | 0.294871795 | 0.301282051 |
| Riverfort | 0.587301587 | 0.571428571 | 0.571428571 | 0.571428571 | 0.53968254 |
| Highfort | 0.485294118 | 0.485294118 | 0.5 | 0.5 | 0.470588235 |
| The Shard | 0.304964539 | 0.304964539 | 0.304964539 | 0.304964539 | 0.262411348 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.234693878 | 0.234693878 | 0.234693878 | 0.234693878 | 0.227040816 |
| Overwatch | 0.254901961 | 0.248366013 | 0.248366013 | 0.248366013 | 0.235294118 |
| Sanctuary Bridge | 0.314102564 | 0.307692308 | 0.294871795 | 0.294871795 | 0.294871795 |
| Riverfort | 0.603174603 | 0.603174603 | 0.587301587 | 0.587301587 | 0.53968254 |
| Highfort | 0.485294118 | 0.485294118 | 0.485294118 | 0.470588235 | 0.470588235 |
| The Shard | 0.304964539 | 0.304964539 | 0.304964539 | 0.304964539 | 0.262411348 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.232142857 | 0.232142857 | 0.232142857 | 0.232142857 | 0.227040816 |
| Overwatch | 0.248366013 | 0.248366013 | 0.248366013 | 0.248366013 | 0.235294118 |
| Sanctuary Bridge | 0.307692308 | 0.307692308 | 0.294871795 | 0.294871795 | 0.301282051 |
| Riverfort | 0.587301587 | 0.571428571 | 0.571428571 | 0.571428571 | 0.53968254 |
| Highfort | 0.485294118 | 0.485294118 | 0.5 | 0.5 | 0.470588235 |
| The Shard | 0.304964539 | 0.304964539 | 0.304964539 | 0.304964539 | 0.262411348 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.204081633 | 0.209183673 | 0.209183673 | 0.209183673 | 0.219387755 |
| Overwatch | 0.189542484 | 0.189542484 | 0.196078431 | 0.183006536 | 0.176470588 |
| Sanctuary Bridge | 0.237179487 | 0.25 | 0.217948718 | 0.243589744 | 0.243589744 |
| Riverfort | 0.412698413 | 0.444444444 | 0.444444444 | 0.444444444 | 0.380952381 |
| Highfort | 0.411764706 | 0.367647059 | 0.323529412 | 0.382352941 | 0.352941176 |
| The Shard | 0.262411348 | 0.234042553 | 0.255319149 | 0.24822695 | 0.226950355 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Baseline** | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | |
| Citadel Gate | 0.224489796 | 0.232142857 | 0.232142857 | 0.230300454 |
| Overwatch | 0.235294118 | 0.241830065 | 0.235294118 | 0.239651416 |
| Sanctuary Bridge | 0.307692308 | 0.301282051 | 0.301282051 | 0.298789174 |
| Riverfort | 0.53968254 | 0.555555556 | 0.53968254 | 0.555555556 |
| Highfort | 0.485294118 | 0.470588235 | 0.485294118 | 0.466503268 |
| The Shard | 0.290780142 | 0.312056738 | 0.29787234 | 0.298266351 |
| **Metric 2** | | | | |
| Citadel Gate | 0.227040816 | 0.237244898 | 0.234693878 | 0.232426304 |
| Overwatch | 0.235294118 | 0.248366013 | 0.235294118 | 0.241466957 |
| Sanctuary Bridge | 0.307692308 | 0.294871795 | 0.301282051 | 0.296296296 |
| Riverfort | 0.587301587 | 0.587301587 | 0.571428571 | 0.574074074 |
| Highfort | 0.485294118 | 0.455882353 | 0.485294118 | 0.465686275 |
| The Shard | 0.276595745 | 0.312056738 | 0.29787234 | 0.295902285 |
| **Metric 3** | | | | |
| Citadel Gate | 0.224489796 | 0.232142857 | 0.232142857 | 0.230300454 |
| Overwatch | 0.235294118 | 0.241830065 | 0.235294118 | 0.239651416 |
| Sanctuary Bridge | 0.307692308 | 0.301282051 | 0.301282051 | 0.298789174 |
| Riverfort | 0.53968254 | 0.555555556 | 0.53968254 | 0.555555556 |
| Highfort | 0.485294118 | 0.470588235 | 0.485294118 | 0.466503268 |
| The Shard | 0.290780142 | 0.312056738 | 0.29787234 | 0.298266351 |
| **Metric 4** | | | | |
| Citadel Gate | 0.204081633 | 0.221938776 | 0.204081633 | 0.209608844 |
| Overwatch | 0.196078431 | 0.176470588 | 0.196078431 | 0.184822077 |
| Sanctuary Bridge | 0.262820513 | 0.25 | 0.217948718 | 0.238960114 |
| Riverfort | 0.412698413 | 0.46031746 | 0.571428571 | 0.427689594 |
| Highfort | 0.323529412 | 0.367647059 | 0.323529412 | 0.339052288 |
| The Shard | 0.290780142 | 0.269503546 | 0.24822695 | 0.247832939 |

Figure .19: Playthrough Prediction Training Accuracies (Method 2 — Binary).

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.581709145 | 0.581709145 |
| Overwatch | 0.54884742 | 0.549342105 | 0.549342105 | 0.549835706 | 0.549835706 |
| Sanctuary Bridge | 0.531701891 | 0.532222222 | 0.532222222 | 0.532150776 | 0.532150776 |
| Riverfort | 0.55 | 0.550440744 | 0.550440744 | 0.55078125 | 0.55078125 |
| Highfort | 0.542120912 | 0.542574257 | 0.542574257 | 0.542941757 | 0.542941757 |
| The Shard | 0.571888412 | 0.572347267 | 0.572347267 | 0.572649573 | 0.572649573 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.628012048 | 0.625 | 0.65060241 | 0.673163418 | 0.665667166 |
| Overwatch | 0.610318332 | 0.594298246 | 0.625 | 0.621029573 | 0.635268346 |
| Sanctuary Bridge | 0.604004449 | 0.587777778 | 0.637777778 | 0.630820399 | 0.657427938 |
| Riverfort | 0.591176471 | 0.590597453 | 0.630754163 | 0.619140625 | 0.634765625 |
| Highfort | 0.575817641 | 0.608910891 | 0.627722772 | 0.650542942 | 0.639684107 |
| The Shard | 0.607296137 | 0.601286174 | 0.627009646 | 0.662393162 | 0.616452991 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.629518072 | 0.628012048 | 0.649096386 | 0.673163418 | 0.650674663 |
| Overwatch | 0.605927552 | 0.606359649 | 0.621710526 | 0.61007667 | 0.628696605 |
| Sanctuary Bridge | 0.588431591 | 0.58 | 0.64 | 0.628603104 | 0.650776053 |
| Riverfort | 0.580392157 | 0.593535749 | 0.621939275 | 0.620117188 | 0.629882813 |
| Highfort | 0.581764123 | 0.597029703 | 0.627722772 | 0.656465943 | 0.633761106 |
| The Shard | 0.608369099 | 0.59807074 | 0.628081458 | 0.663461538 | 0.612179487 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.628012048 | 0.623493976 | 0.649096386 | 0.671664168 | 0.665667166 |
| Overwatch | 0.610318332 | 0.594298246 | 0.625 | 0.621029573 | 0.635268346 |
| Sanctuary Bridge | 0.605116796 | 0.587777778 | 0.638888889 | 0.630820399 | 0.65631929 |
| Riverfort | 0.591176471 | 0.590597453 | 0.630754163 | 0.619140625 | 0.634765625 |
| Highfort | 0.575817641 | 0.608910891 | 0.627722772 | 0.649555775 | 0.639684107 |
| The Shard | 0.607296137 | 0.602357985 | 0.627009646 | 0.663461538 | 0.616452991 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.579819277 | 0.606927711 | 0.603915663 | 0.580209895 | 0.580209895 |
| Overwatch | 0.602634468 | 0.584429825 | 0.605263158 | 0.56407448 | 0.577217963 |
| Sanctuary Bridge | 0.590656285 | 0.577777778 | 0.62 | 0.576496674 | 0.611973392 |
| Riverfort | 0.585294118 | 0.576885406 | 0.597453477 | 0.568359375 | 0.583984375 |
| Highfort | 0.575817641 | 0.59009901 | 0.599009901 | 0.589338598 | 0.583415597 |
| The Shard | 0.604077253 | 0.588424437 | 0.617363344 | 0.592948718 | 0.567307692 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581709145 | 0.581325301 | 0.581709145 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549835706 | 0.54884742 | 0.549835706 | 0.549342105 |
| Sanctuary Bridge | 0.532222222 | 0.532150776 | 0.531701891 | 0.532150776 | 0.532222222 |
| Riverfort | 0.550440744 | 0.55078125 | 0.55 | 0.55078125 | 0.550440744 |
| Highfort | 0.542574257 | 0.542941757 | 0.542120912 | 0.542941757 | 0.542574257 |
| The Shard | 0.572347267 | 0.572649573 | 0.571888412 | 0.572649573 | 0.572347267 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.593373494 | 0.584707646 | 0.628012048 | 0.604197901 | 0.614457831 |
| Overwatch | 0.563596491 | 0.56736035 | 0.575192097 | 0.56626506 | 0.592105263 |
| Sanctuary Bridge | 0.56 | 0.535476718 | 0.556173526 | 0.557649667 | 0.598888889 |
| Riverfort | 0.564152791 | 0.557617188 | 0.579411765 | 0.571289063 | 0.598432909 |
| Highfort | 0.547524752 | 0.559723593 | 0.580773043 | 0.562685094 | 0.60990099 |
| The Shard | 0.573419078 | 0.572649573 | 0.592274678 | 0.588675214 | 0.59807074 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.590361446 | 0.583208396 | 0.626506024 | 0.607196402 | 0.615963855 |
| Overwatch | 0.557017544 | 0.557502738 | 0.576289791 | 0.556407448 | 0.604166667 |
| Sanctuary Bridge | 0.546666667 | 0.525498891 | 0.557285873 | 0.557649667 | 0.586666667 |
| Riverfort | 0.554358472 | 0.55859375 | 0.579411765 | 0.564453125 | 0.605288932 |
| Highfort | 0.557425743 | 0.544916091 | 0.581764123 | 0.554787759 | 0.592079208 |
| The Shard | 0.576634512 | 0.572649573 | 0.597639485 | 0.588675214 | 0.592711683 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.593373494 | 0.584707646 | 0.628012048 | 0.604197901 | 0.614457831 |
| Overwatch | 0.563596491 | 0.56736035 | 0.575192097 | 0.56626506 | 0.592105263 |
| Sanctuary Bridge | 0.56 | 0.535476718 | 0.556173526 | 0.557649667 | 0.597777778 |
| Riverfort | 0.564152791 | 0.557617188 | 0.579411765 | 0.571289063 | 0.598432909 |
| Highfort | 0.547524752 | 0.559723593 | 0.579781962 | 0.562685094 | 0.608910891 |
| The Shard | 0.573419078 | 0.572649573 | 0.592274678 | 0.588675214 | 0.59807074 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.59186747 | 0.574212894 | 0.542168675 | 0.586206897 | 0.59186747 |
| Overwatch | 0.557017544 | 0.559693319 | 0.531284303 | 0.552026287 | 0.588815789 |
| Sanctuary Bridge | 0.552222222 | 0.533259424 | 0.536151279 | 0.544345898 | 0.577777778 |
| Riverfort | 0.554358472 | 0.556640625 | 0.530392157 | 0.55859375 | 0.591576885 |
| Highfort | 0.556435644 | 0.553800592 | 0.533201189 | 0.544916091 | 0.584158416 |
| The Shard | 0.575562701 | 0.572649573 | 0.557939914 | 0.573717949 | 0.593783494 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.581325301 | 0.581325301 |
| Overwatch | 0.549342105 | 0.549342105 | 0.549342105 | 0.549342105 | 0.549342105 |
| Sanctuary Bridge | 0.532222222 | 0.532222222 | 0.532222222 | 0.532222222 | 0.532222222 |
| Riverfort | 0.550440744 | 0.550440744 | 0.550440744 | 0.550440744 | 0.550440744 |
| Highfort | 0.542574257 | 0.542574257 | 0.542574257 | 0.542574257 | 0.542574257 |
| The Shard | 0.572347267 | 0.572347267 | 0.572347267 | 0.572347267 | 0.572347267 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.581325301 | 0.596385542 | 0.59939759 | 0.602409639 | 0.614457831 |
| Overwatch | 0.567982456 | 0.582236842 | 0.578947368 | 0.577850877 | 0.596491228 |
| Sanctuary Bridge | 0.55 | 0.566666667 | 0.55 | 0.548888889 | 0.582222222 |
| Riverfort | 0.565132223 | 0.571988247 | 0.571988247 | 0.565132223 | 0.589618022 |
| Highfort | 0.555445545 | 0.579207921 | 0.575247525 | 0.573267327 | 0.589108911 |
| The Shard | 0.59056806 | 0.586280815 | 0.585209003 | 0.587352626 | 0.617363344 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.581325301 | 0.594879518 | 0.593373494 | 0.602409639 | 0.620481928 |
| Overwatch | 0.564692982 | 0.575657895 | 0.57127193 | 0.565789474 | 0.591008772 |
| Sanctuary Bridge | 0.532222222 | 0.566666667 | 0.537777778 | 0.55 | 0.584444444 |
| Riverfort | 0.568070519 | 0.565132223 | 0.567091087 | 0.565132223 | 0.581782566 |
| Highfort | 0.558415842 | 0.581188119 | 0.571287129 | 0.57029703 | 0.589108911 |
| The Shard | 0.59056806 | 0.588424437 | 0.588424437 | 0.591639871 | 0.59807074 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.581325301 | 0.596385542 | 0.59939759 | 0.602409639 | 0.614457831 |
| Overwatch | 0.567982456 | 0.581140351 | 0.578947368 | 0.577850877 | 0.596491228 |
| Sanctuary Bridge | 0.55 | 0.566666667 | 0.548888889 | 0.548888889 | 0.582222222 |
| Riverfort | 0.565132223 | 0.571988247 | 0.571988247 | 0.565132223 | 0.589618022 |
| Highfort | 0.555445545 | 0.578217822 | 0.575247525 | 0.574257426 | 0.589108911 |
| The Shard | 0.59056806 | 0.584137192 | 0.585209003 | 0.587352626 | 0.617363344 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.581325301 | 0.578313253 | 0.570783133 | 0.576807229 | 0.582831325 |
| Overwatch | 0.563596491 | 0.575657895 | 0.574561404 | 0.567982456 | 0.58004386 |
| Sanctuary Bridge | 0.546666667 | 0.565555556 | 0.54 | 0.553333333 | 0.581111111 |
| Riverfort | 0.567091087 | 0.571008815 | 0.567091087 | 0.562193928 | 0.569049951 |
| Highfort | 0.557425743 | 0.576237624 | 0.571287129 | 0.574257426 | 0.576237624 |
| The Shard | 0.59056806 | 0.578778135 | 0.588424437 | 0.588424437 | 0.594855305 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Baseline** | | | | |
| Citadel Gate | 0.581325301 | 0.581325301 | 0.581325301 | 0.5814106 |
| Overwatch | 0.549342105 | 0.549342105 | 0.549342105 | 0.549396829 |
| Sanctuary Bridge | 0.532222222 | 0.532222222 | 0.532222222 | 0.532148531 |
| Riverfort | 0.550440744 | 0.550440744 | 0.550440744 | 0.550467441 |
| Highfort | 0.542574257 | 0.542574257 | 0.542574257 | 0.542605552 |
| The Shard | 0.572347267 | 0.572347267 | 0.572347267 | 0.572363462 |
| **Metric 1** | | | | |
| Citadel Gate | 0.640060241 | 0.644578313 | 0.629518072 | 0.620851472 |
| Overwatch | 0.628289474 | 0.605263158 | 0.609649123 | 0.594285794 |
| Sanctuary Bridge | 0.593333333 | 0.578888889 | 0.588888889 | 0.582493668 |
| Riverfort | 0.6043095 | 0.58863859 | 0.589618022 | 0.58798684 |
| Highfort | 0.614851485 | 0.599009901 | 0.594059406 | 0.591304658 |
| The Shard | 0.620578778 | 0.59807074 | 0.59807074 | 0.601278972 |
| **Metric 2** | | | | |
| Citadel Gate | 0.635542169 | 0.631024096 | 0.629518072 | 0.619014163 |
| Overwatch | 0.626096491 | 0.606359649 | 0.594298246 | 0.589962813 |
| Sanctuary Bridge | 0.575555556 | 0.573333333 | 0.582222222 | 0.575766708 |
| Riverfort | 0.607247796 | 0.599412341 | 0.579823702 | 0.585648094 |
| Highfort | 0.623762376 | 0.597029703 | 0.59009901 | 0.589383594 |
| The Shard | 0.603429796 | 0.602357985 | 0.602357985 | 0.600208117 |
| **Metric 3** | | | | |
| Citadel Gate | 0.638554217 | 0.644578313 | 0.628012048 | 0.620433508 |
| Overwatch | 0.628289474 | 0.605263158 | 0.609649123 | 0.594224877 |
| Sanctuary Bridge | 0.593333333 | 0.577777778 | 0.588888889 | 0.582370417 |
| Riverfort | 0.6043095 | 0.58863859 | 0.589618022 | 0.58798684 |
| Highfort | 0.614851485 | 0.600990099 | 0.594059406 | 0.591249761 |
| The Shard | 0.620578778 | 0.59807074 | 0.59807074 | 0.601278781 |
| **Metric 4** | | | | |
| Citadel Gate | 0.61746988 | 0.593373494 | 0.587349398 | 0.584758825 |
| Overwatch | 0.597587719 | 0.587719298 | 0.589912281 | 0.575528808 |
| Sanctuary Bridge | 0.568888889 | 0.564444444 | 0.582222222 | 0.567937942 |
| Riverfort | 0.584720862 | 0.57884427 | 0.569049951 | 0.570699366 |
| Highfort | 0.600990099 | 0.579207921 | 0.582178218 | 0.573778581 |
| The Shard | 0.581993569 | 0.59056806 | 0.589496249 | 0.585937963 |

Figure .20: Playthrough Prediction Test Accuracies (Method 2 — Binary).

| (t, t', t'', κ₁, κ₂) | (2, 20, 60, 20, 6) | (2, 20, 50, 22, 7) | (4, 10, 50, 22, 4) | (6, 10, 40, 12, 7) | (6, 10, 40, 19, 6) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.489795918 | 0.477040816 | 0.474489796 | 0.50255102 | 0.494897959 |
| Overwatch | 0.424836601 | 0.437908497 | 0.418300654 | 0.418300654 | 0.45751634 |
| Sanctuary Bridge | 0.435897436 | 0.423076923 | 0.429487179 | 0.416666667 | 0.397435897 |
| Riverfort | 0.523809524 | 0.53968254 | 0.492063492 | 0.476190476 | 0.444444444 |
| Highfort | 0.455882353 | 0.397058824 | 0.382352941 | 0.441176471 | 0.397058824 |
| The Shard | 0.539007092 | 0.588652482 | 0.553191489 | 0.567375887 | 0.588652482 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.489795918 | 0.482142857 | 0.474489796 | 0.50255102 | 0.49744898 |
| Overwatch | 0.424836601 | 0.418300654 | 0.424836601 | 0.424836601 | 0.450980392 |
| Sanctuary Bridge | 0.442307692 | 0.416666667 | 0.435897436 | 0.403846154 | 0.378205128 |
| Riverfort | 0.53968254 | 0.53968254 | 0.53968254 | 0.492063492 | 0.476190476 |
| Highfort | 0.441176471 | 0.411764706 | 0.367647059 | 0.455882353 | 0.382352941 |
| The Shard | 0.553191489 | 0.574468085 | 0.546099291 | 0.567375887 | 0.581560284 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.489795918 | 0.477040816 | 0.474489796 | 0.50255102 | 0.494897959 |
| Overwatch | 0.424836601 | 0.437908497 | 0.418300654 | 0.418300654 | 0.45751634 |
| Sanctuary Bridge | 0.435897436 | 0.423076923 | 0.429487179 | 0.416666667 | 0.397435897 |
| Riverfort | 0.523809524 | 0.53968254 | 0.492063492 | 0.476190476 | 0.444444444 |
| Highfort | 0.455882353 | 0.397058824 | 0.382352941 | 0.441176471 | 0.397058824 |
| The Shard | 0.539007092 | 0.588652482 | 0.553191489 | 0.567375887 | 0.588652482 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.479591837 | 0.464285714 | 0.464285714 | 0.482142857 | 0.477040816 |
| Overwatch | 0.392156863 | 0.392156863 | 0.392156863 | 0.39869281 | 0.411764706 |
| Sanctuary Bridge | 0.384615385 | 0.391025641 | 0.371794872 | 0.371794872 | 0.326923077 |
| Riverfort | 0.46031746 | 0.46031746 | 0.46031746 | 0.396825397 | 0.444444444 |
| Highfort | 0.382352941 | 0.397058824 | 0.352941176 | 0.426470588 | 0.323529412 |
| The Shard | 0.524822695 | 0.510638298 | 0.517730496 | 0.553191489 | 0.517730496 |

| (t, t', t'', κ$_1$, κ$_2$) | (6, 16, 50, 12, 3) | (4, 20, 40, 24, 5) | (6, 10, 60, 20, 10) | (6, 10, 40, 12, 3) | (2, 20, 50, 16, 7) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.474489796 | 0.487244898 | 0.492346939 | 0.489795918 | 0.477040816 |
| Overwatch | 0.444444444 | 0.509803922 | 0.431372549 | 0.444444444 | 0.444444444 |
| Sanctuary Bridge | 0.423076923 | 0.467948718 | 0.397435897 | 0.423076923 | 0.416666667 |
| Riverfort | 0.46031746 | 0.492063492 | 0.444444444 | 0.507936508 | 0.523809524 |
| Highfort | 0.367647059 | 0.470588235 | 0.323529412 | 0.382352941 | 0.397058824 |
| The Shard | 0.539007092 | 0.553191489 | 0.574468085 | 0.531914894 | 0.574468085 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.474489796 | 0.487244898 | 0.492346939 | 0.489795918 | 0.479591837 |
| Overwatch | 0.437908497 | 0.503267974 | 0.437908497 | 0.444444444 | 0.437908497 |
| Sanctuary Bridge | 0.41025641 | 0.455128205 | 0.397435897 | 0.429487179 | 0.403846154 |
| Riverfort | 0.46031746 | 0.492063492 | 0.444444444 | 0.492063492 | 0.53968254 |
| Highfort | 0.382352941 | 0.441176471 | 0.323529412 | 0.382352941 | 0.382352941 |
| The Shard | 0.539007092 | 0.553191489 | 0.567375887 | 0.524822695 | 0.553191489 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.474489796 | 0.487244898 | 0.492346939 | 0.489795918 | 0.477040816 |
| Overwatch | 0.444444444 | 0.509803922 | 0.431372549 | 0.444444444 | 0.444444444 |
| Sanctuary Bridge | 0.423076923 | 0.467948718 | 0.397435897 | 0.423076923 | 0.416666667 |
| Riverfort | 0.46031746 | 0.492063492 | 0.444444444 | 0.507936508 | 0.523809524 |
| Highfort | 0.367647059 | 0.470588235 | 0.323529412 | 0.382352941 | 0.397058824 |
| The Shard | 0.539007092 | 0.553191489 | 0.574468085 | 0.531914894 | 0.574468085 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.466836735 | 0.471938776 | 0.471938776 | 0.474489796 | 0.466836735 |
| Overwatch | 0.385620915 | 0.431372549 | 0.366013072 | 0.39869281 | 0.39869281 |
| Sanctuary Bridge | 0.378205128 | 0.403846154 | 0.371794872 | 0.378205128 | 0.391025641 |
| Riverfort | 0.428571429 | 0.412698413 | 0.396825397 | 0.46031746 | 0.492063492 |
| Highfort | 0.352941176 | 0.426470588 | 0.338235294 | 0.323529412 | 0.382352941 |
| The Shard | 0.510638298 | 0.510638298 | 0.531914894 | 0.503546099 | 0.539007092 |

| (t, t', t'', $\kappa_1$, $\kappa_2$) | (2, 20, 50, 10, 3) | (2, 20, 50, 16, 6) | (2, 20, 50, 24, 5) | (2, 20, 50, 22, 5) | (2, 20, 50, 22, 8) |
|---|---|---|---|---|---|
| **Baseline** | | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | | |
| Citadel Gate | 0.487244898 | 0.487244898 | 0.487244898 | 0.484693878 | 0.482142857 |
| Overwatch | 0.424836601 | 0.45751634 | 0.437908497 | 0.444444444 | 0.418300654 |
| Sanctuary Bridge | 0.429487179 | 0.403846154 | 0.403846154 | 0.41025641 | 0.429487179 |
| Riverfort | 0.523809524 | 0.492063492 | 0.507936508 | 0.507936508 | 0.571428571 |
| Highfort | 0.455882353 | 0.367647059 | 0.411764706 | 0.411764706 | 0.426470588 |
| The Shard | 0.524822695 | 0.524822695 | 0.524822695 | 0.546099291 | 0.496453901 |
| **Metric 2** | | | | | |
| Citadel Gate | 0.487244898 | 0.484693878 | 0.484693878 | 0.479591837 | 0.471938776 |
| Overwatch | 0.418300654 | 0.444444444 | 0.444444444 | 0.444444444 | 0.392156863 |
| Sanctuary Bridge | 0.429487179 | 0.397435897 | 0.416666667 | 0.41025641 | 0.423076923 |
| Riverfort | 0.523809524 | 0.492063492 | 0.492063492 | 0.492063492 | 0.571428571 |
| Highfort | 0.455882353 | 0.382352941 | 0.441176471 | 0.411764706 | 0.426470588 |
| The Shard | 0.531914894 | 0.524822695 | 0.517730496 | 0.553191489 | 0.503546099 |
| **Metric 3** | | | | | |
| Citadel Gate | 0.487244898 | 0.487244898 | 0.487244898 | 0.484693878 | 0.479591837 |
| Overwatch | 0.424836601 | 0.45751634 | 0.437908497 | 0.444444444 | 0.418300654 |
| Sanctuary Bridge | 0.429487179 | 0.403846154 | 0.403846154 | 0.41025641 | 0.429487179 |
| Riverfort | 0.523809524 | 0.492063492 | 0.507936508 | 0.507936508 | 0.571428571 |
| Highfort | 0.455882353 | 0.367647059 | 0.411764706 | 0.411764706 | 0.426470588 |
| The Shard | 0.524822695 | 0.524822695 | 0.524822695 | 0.546099291 | 0.496453901 |
| **Metric 4** | | | | | |
| Citadel Gate | 0.471938776 | 0.471938776 | 0.474489796 | 0.464285714 | 0.469387755 |
| Overwatch | 0.385620915 | 0.385620915 | 0.385620915 | 0.39869281 | 0.385620915 |
| Sanctuary Bridge | 0.365384615 | 0.397435897 | 0.384615385 | 0.384615385 | 0.378205128 |
| Riverfort | 0.476190476 | 0.492063492 | 0.46031746 | 0.428571429 | 0.444444444 |
| Highfort | 0.397058824 | 0.411764706 | 0.411764706 | 0.397058824 | 0.441176471 |
| The Shard | 0.496453901 | 0.503546099 | 0.539007092 | 0.531914894 | 0.510638298 |

| $(t, t', t'', \kappa_1, \kappa_2)$ | (2, 20, 50, 22, 9) | (2, 20, 50, 23, 7) | (2, 20, 50, 21, 7) | Mean |
|---|---|---|---|---|
| **Baseline** | | | | |
| Citadel Gate | 0.528061224 | 0.528061224 | 0.528061224 | 0.528061224 |
| Overwatch | 0.594771242 | 0.594771242 | 0.594771242 | 0.594771242 |
| Sanctuary Bridge | 0.666666667 | 0.666666667 | 0.666666667 | 0.666666667 |
| Riverfort | 0.603174603 | 0.603174603 | 0.603174603 | 0.603174603 |
| Highfort | 0.720588235 | 0.720588235 | 0.720588235 | 0.720588235 |
| The Shard | 0.460992908 | 0.460992908 | 0.460992908 | 0.460992908 |
| **Metric 1** | | | | |
| Citadel Gate | 0.477040816 | 0.489795918 | 0.492346939 | 0.485969388 |
| Overwatch | 0.450980392 | 0.437908497 | 0.450980392 | 0.441902687 |
| Sanctuary Bridge | 0.423076923 | 0.416666667 | 0.416666667 | 0.42022792 |
| Riverfort | 0.492063492 | 0.53968254 | 0.492063492 | 0.501763668 |
| Highfort | 0.397058824 | 0.426470588 | 0.397058824 | 0.406045752 |
| The Shard | 0.524822695 | 0.560283688 | 0.574468085 | 0.549251379 |
| **Metric 2** | | | | |
| Citadel Gate | 0.479591837 | 0.489795918 | 0.494897959 | 0.485685941 |
| Overwatch | 0.444444444 | 0.431372549 | 0.431372549 | 0.436456064 |
| Sanctuary Bridge | 0.423076923 | 0.397435897 | 0.416666667 | 0.415954416 |
| Riverfort | 0.492063492 | 0.53968254 | 0.523809524 | 0.507936508 |
| Highfort | 0.382352941 | 0.441176471 | 0.426470588 | 0.407679739 |
| The Shard | 0.517730496 | 0.567375887 | 0.560283688 | 0.546493302 |
| **Metric 3** | | | | |
| Citadel Gate | 0.477040816 | 0.489795918 | 0.492346939 | 0.485827664 |
| Overwatch | 0.450980392 | 0.437908497 | 0.450980392 | 0.441902687 |
| Sanctuary Bridge | 0.423076923 | 0.416666667 | 0.416666667 | 0.42022792 |
| Riverfort | 0.492063492 | 0.53968254 | 0.492063492 | 0.501763668 |
| Highfort | 0.397058824 | 0.426470588 | 0.397058824 | 0.406045752 |
| The Shard | 0.524822695 | 0.560283688 | 0.574468085 | 0.549251379 |
| **Metric 4** | | | | |
| Citadel Gate | 0.469387755 | 0.489795918 | 0.477040816 | 0.472647392 |
| Overwatch | 0.392156863 | 0.418300654 | 0.379084967 | 0.394335512 |
| Sanctuary Bridge | 0.397435897 | 0.358974359 | 0.391025641 | 0.379273504 |
| Riverfort | 0.412698413 | 0.476190476 | 0.46031746 | 0.447971781 |
| Highfort | 0.426470588 | 0.455882353 | 0.382352941 | 0.390522876 |
| The Shard | 0.510638298 | 0.531914894 | 0.517730496 | 0.520094563 |