FACULTY OF SCIENCES

Modeling Player Experience in Web-Based Video Games

Yasmine Bogaert
Student number: 01403612

Supervisor: Prof. dr. ir. Francis wyffels
Counsellor: Andreas Verleysen

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de informatica

Academic year: 2020 - 2021

GHENT UNIVERSITY

Samenvatting

Spelontwerpers streven er bijna altijd naar om spellen te maken waarin de moeilijkheidsgraad van een spel geschikt is voor het vaardigheidsniveau van de speler. Toch zijn ontwerpers meestal niet in staat om het spel aan te passen aan het vaardigheidsniveau van elke speler, en frustrerende mismatches tussen de vaardigheid van een speler en de moeilijkheidsgraad van een spel komen vaak voor. Spelers beginnen aan spellen op verschillende vaardigheidsniveaus en ontwikkelen door te spelen hun eigen vaardigheid op hun eigen tempo. Voor sommige spelers worden zelfs de huidige best ontworpen spellen snel te gemakkelijk, terwijl ze voor anderen frustrerend en moeilijk zijn. Naarmate de vaardigheid van een speler verbetert door te spelen, zou een goed ontworpen spel meer moeilijkheden moeten aanbieden, zodat de speler zich nooit verveelt door een te gemakkelijke gameplay of gefrustreerd raakt door een te moeilijke gameplay. Tegenwoordig kunnen deze moeilijkheidsgraad en de bijbehorende parameters alleen aan het begin van het spel worden bepaald door een vastgelegde moeilijkheidsgraad te selecteren, vaak door de speler zelf. Toch kan dit leiden tot een frustrerende ervaring voor zowel ervaren als onervaren spelers, omdat ze proberen een vooraf geselecteerde leer- of moeilijkheidscurve te volgen die is samengesteld door spel-ontwikkelaars.

Een oplossing voor dit probleem is dynamic difficulty adjustment (DDA), het proces waarbij parameters in een videogame in realtime worden gewijzigd, op basis van het waargenomen gedrag van de speler, om de moeilijkheidsgraad van het spel in evenwicht te brengen. Dit wordt gedaan om te voorkomen dat de speler zich verveelt (als het spel te gemakkelijk wordt) of frustreert (als het te moeilijk wordt) en zo beter past bij het overeenkomstige vaardigheidsniveau van de speler. DDA wordt gedaan tijdens het spel. Het gedrag van de speler wordt bekeken en het spel wordt daaraan aangepast om de speler de juiste uitdagingen te bieden en een goede match te garanderen tussen de vaardigheid van de speler en de moeilijkheidsgraad van het spel.

Voor een goede realisatie van DDA is een goed spelersmodel nodig. Een spelersmodel maakt het mogelijk om het gedrag van spelers te voorspellen in nieuwe situaties die nog nooit eerder zijn waargenomen. Dit model wordt vervolgens gecombineerd met een algoritme om de game aan te passen op basis van die observaties. Een adaptief spel dat is gebaseerd op nauwkeurige spelersmodellen, is in staat om voorspellingen te doen over toekomstig gedrag, het spel aan die veranderingen aan te passen en spelers betere games te kunnen aanbieden waarvan wordt verwacht dat ze ze leuk zullen vinden.

Om onderliggende moeilijkheidsparameters te analyseren, het gedrag van spelers te modelleren en te streven naar een systeem voor DDA, kijken we naar de videogame Chrome Dino, een endless runner waarin de speler een dinosaurus bestuurt die kan springen of bukken om obstakels te vermijden. Er is een aangepaste versie van dit spel opgezet, met verschillende gesamplede moeilijkheidsparameters. De emotionele responses van de speler worden gemeten met een gerangschikte vragenlijst om te bepalen of er statistisch significante verschillen zijn in de perceptie van het spel. Spelerspersona's worden op verschillende manieren onderzocht, met onder andere dimensionaliteitsreductie, clustering en classificatie.

Kernwoorden

Dynamic Difficulty Adjustment, Game Difficulty, Flow, Player Experience, Player Modeling, Chrome Dino.

Summary

Game designers nearly always strive to create games in which the difficulty of a game is appropriate for the player's skill level. Even so, designers are usually not able to accommodate every player's skill level, and frustrating mismatches between a player's skill and a game's difficulty are common. Players begin games at different skill levels, and through playing develop their own skill at their own rates. For some players, even the current best-designed games become uninterestingly easy, while frustrating and difficult for others. As a player's skill improves through practice, a well-designed game will present more difficulties so that the player is never bored by overly easy gameplay or frustrated by overly difficult gameplay. Most of the time, this difficulty and the parameters that come along with it, can only be determined at the beginning of the game by selecting a difficulty level, often done by the player himself/herself. Still, this can lead to a frustrating experience for both experienced and inexperienced gamers, as they attempt to follow a preselected learning or difficulty curve composed by game developers.

A proposed answer to this challenge is dynamic difficulty adjustment (DDA), the process of changing underlying parameters in a video game in real-time, based on the player's observed ability, to balance the game difficulty. This is done in order to avoid making the player bored (if the game is too easy) or frustrated (if it is too hard) and better fit the according skill level of the player. DDA is done during execution, tracking the player's performance and adjusting the game to present proper challenges to the player to ensure a good match between a player's skill and the game's difficulty.

To ensure a proper realization of DDA, a good player model is required. A player model will enable to predict player behavior in new situations that may have never been observed before. This model is then combined with an algorithm to adapt the game content based on those observations. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy.

In an attempt to analyze difficulty parameters, model player behavior, and strive towards a system for DDA, we look at the video game Chrome Dino, an endless runner game in which the player controls a running dinosaur that can jump or duck to avoid obstacles. A custom version of this game is set up, with various sampled difficulty parameters and the player's response is measured with a ranked questionnaire in order to determine whether there are statistically significant differences in the perception of gameplay. Player personas and player models are analyzed with various methods, including dimensionality reduction, clustering and classification.

Keywords

Dynamic Difficulty Adjustment, Game Difficulty, Flow, Player Experience, Player Modeling, Chrome Dino.

Modeling Player Experience in Web-Based Video Games

Yasmine Bogaert[1], Prof. dr. ir. Francis wyffels[2], Andreas Verleysen[2]

Abstract

The video game industry has been growing quickly over the past years and so has the technologies that come with it. Still, one aspect has remained stagnant, the difficulty balancing. For some players, even the current best-designed games often become uninterestingly easy, while frustrating and difficult for others. A proposed answer to this challenge is dynamic difficulty adjustment (DDA), in which the difficulty of the game is dynamically adapted to match the player's skill level. To ensure a proper realization of DDA, an good player model is required that represents the relation between the player's enjoyment and the underlying gameplay parameters. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy. This article looks at player modeling for the web-based Chrome Dino game, in the context of DDA.

Keywords: Dynamic Difficulty Adjustment, Game Difficulty, Flow, Player Experience, Player Modeling, Chrome Dino.

1. Introduction

One might ask what keeps players entertained and come back to games. Even with concepts such as gamification, in which concepts from games (such as rewards, rankings, point systems) are used in non-game environments to enhance the user's experience, creating a game that is able to create enjoyment for many people and is able to retain that, is not an easy task. Players often get bored when a game gets too easy, or frustrated when a game gets too hard. The key factor is balancing the difficulty of a game so that it matches the skill of the player. Gamification and other similar concepts are based on general aspects of psychology that only cover global trends. The gaming audience is wide spread, and so is the individual skill level of players, ranging from new players to players with a lot of experience.

A proposed answer to this challenge is dynamic difficulty adjustment, which is the process of changing underlying parameters in a video game in real-time, based on the player's observed ability, to balance the game difficulty. It does this by building an underlying model of the player, and then makes an informed decision on how to adapt the difficulty to improve upon. The most important step in constructing a DDA system, is creating this player model. Player models are based on objective game metrics, combined with labels about the individual subjective player's experience. A player model will be able to predict player behavior in new situations that may have never been observed before. This model is then combined with an algorithm to adapt the game content based on those observations. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy.

In this work, an attempt is made at constructing player models for player behaviour in the web-based video game Chrome Dino. First, a literature study is given on dynamic difficulty adjustment, player enjoyment, player modeling, how to measure player experience and techniques for player modeling. Secondly, data collection is done to give insight into how player behaviour relates with player enjoyment. Afterwards, several techniques for player modeling are applied, using the collected data records. Finally, results are analysed and a view on adaptations for future work is given.

2. Literature

2.1. Dynamic Difficulty Adjustment

Dynamic difficulty adjustment is a technique used in games to adapt the game's difficulty parameters to the players skill level in order to ensure that the player does not get bored (when the game is too easy) or frustrated (when the game is too hard). The essence of DDA is to retain the interest of the user throughout the game while offering a satisfactory challenge level. DDA is done during execution, tracking the player's performance and adjusting the game to present proper challenges to the player to ensure a good match between a player's skill and the game's difficulty. Different approaches are found in the literature to address dynamic game difficulty balancing. Commonly used techniques for DDA include reinforcement learning (RL), artificial neural networks (ANN), genetic algorithms or dynamic scripting.

2.2. Player Enjoyment

There have been several psychological studies that try to identify what is 'fun' in a game and what engages people

[1]Faculty of Sciences, Ghent University
[2]Faculty of Engineering and Architecture, Ghent University

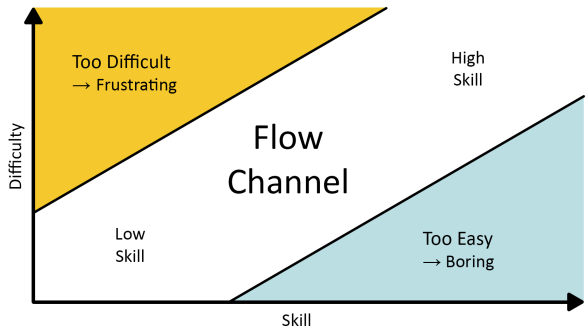


Figure 1: Flow channel.

to play video games. These studies include Malone's principles of intrinsic qualitative factors for engaging gameplay [1], namely challenge, curiosity and fantasy as well as the concept of flow theory by Csikszentmihalyi [2].

The concept of flow defines player states in two dimensions: skill and challenge. It relates the difficulty of a task to the viewers perception of its challenge. The flow channel is a state where a balance between skill and challenge is achieved, often described colloquially as being in "the Zone". When the difficulty of the game is higher than the player's skill, the game becomes frustrating, pushing the player into a state of anxiety. On the other hand, when the player's skill is higher than the difficulty, the game is too easy and pushes the player into a state of boredom. The goal is to keep the player in a state of flow and away from states where the game is far too challenging, or way too easy. [3, 4, 5]

2.3. Player Modeling

The primary goal of player modeling is to understand how a player's interaction with a game has an effect on the player's individual experience. We try to get an understanding of players' cognitive, affective and behavioral patterns [6].

There are two main factors that come into play: player behaviour and player experience. Player behavior is about what a player does in a game whereas player experience refers to how a player feels during gameplay. To construct an accurate player model, both the player behaviour and the player experience are measured and form the inputs for the player model.

2.4. Measuring Player Experience

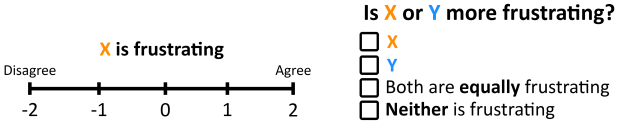
There are different ways of measuring player experience. Labeling can either be done by looking at players emotional responses through physiological measurements (facial expressions, muscle activation, brain waves, posture or speech) or through manual annotation, either done by the player himself/herself or by a third-person. The assigned labels ideally need to be as close to the ground truth as possible. The accuracy of manual annotation is regularly questioned as there are numerous factors contributing to a deviation between a label and the actual underlying player experience [6]. The most common technique of labeling experiences is through a questionnaire.

Player experiences can be evaluated either with free responses (e.g. thinking out loud) or forced responses (e.g. questionnaires). Free response contain more detailed information, but are often unstructured, chaotic and therefore become hard to analyze. On the other hand, forcing players to self-report their experiences using directed questions constrains them to specific items.

There are different ways of structuring data about player experiences: through rating, classes or ranks. Ratings ask a player to pick between a scalar value or a vector of values to indicate his opinion on given statement. Nowadays, ratings are unarguably the most widely used approach for assessing different aspects of a user's experience. The most popular rating-based questionnaires make use of questions in the format of a Likert scale [7], which asks the user to indicate their level of agreement (or disagreement) with a given statement, within a range of predefined points. Rating-based questioning is inherent to limitations, which are often overlooked [8, 9, 6]. Ratings are often analyzed by comparing their values across all participants, which neglects the existence of inter-personal differences as the meaning of each level on a rating scale may differ across participants. Ratings also struggle with primacy and recency effects, in which information at the beginning and at the end are retained better than information in the middle. On top of that, ratings are commonly converted into ordinal values, while in most questionnaires, labels are represented as pictures or adjectives, violating basic axioms of statistics.

Classes ask players to select from a set of two or more options. Classes have less statistical limitations compared to ratings, but do not reach the same level of detail [6].

Rank-based questions ask the player to make a preference ranking between two or more options. In the case of pairwise preferences, the player is asked to compare two options and specifies which one is preferred. With more than two options, the participants are asked to provide a ranking of some or all the options. Rankings do not suffer from the aforementioned limitations that are posed in rating or classes.



(a) Ratings on a 5-point Likert Scale

(b) Rankings through 4-alternative forced choice

Figure 2: Examples of ratings vs rankings.

Similar research questionnaires often ask players to rank games of each game pair with respect to fun, challenge, boredom, frustration, excitement, anxiety and relaxation. The selection of these seven states is based on their relevance to computer game-playing and player experience [9]. Smaller,

initial investigations only focus on three emotions: fun, challenge and frustration [10].

2.5. Techniques for Player Modeling

The following methods are investigated: dimensionality reduction, clustering and classification.

Dimensionality reduction transforms data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data. The data can then be visualized and better interpreted. Tehniques for dimensionality reduction include t-SNE and UMAP. While both UMAP and t-SNE produce somewhat similar results, the increased speed, better preservation of global structure, and more understandable parameters make UMAP a more effective tool [11].

Clustering groups a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. In terms of player modeling, it is the classification of player behavior into a number of clusters depending of their behavioral attributes. Clustering is not only relevant for the personalization of the game, but also for understanding player behavior in association with the game design [12]. Common cluster techniques include k-means, DBScan, spectral clustering or Gaussian mixture models .

Classification learns to predict class labels for new data, based on sample data. In terms of player modeling, the samples consist of data from player behaviour, combined with a label of player experience (i.e. fun). The goal is the classification of new player behavior in classes of player experience. Common classification techniques include decision trees, k-nearest neighbors, logistic regression or support vector machines. Ensemble methods take an additional step and combine the predictions of several base estimators in order to improve generalizability and robustness over a single estimator. Common techniques for ensemble methods include random forests or gradient boosting.

3. Methodology

3.1. Game Selection

To select the game, the following parameters have been taken into consideration:

1. **Duration.** Shorter games are desirable, to avoid loss of memory, maintain player retention and gather as much data entries as possible.
2. **Number of game elements.** A game with a good balance in amount of elements, not too much, but also not too little, would be an ideal fit.
3. **Easy to implement.** An open source model of the game should be available that can be adjusted, or it is doable to start from scratch and remake the game within realizable time.
4. **Expected popularity.** The game should attract attention within the biggest target audience.

A collection of games is gathered that fit at least one or more of the aforementioned parameters. These games include: Chrome Dino, Pacman, Flappy Bird, Snake, Space Invaders, Pong, Super Mario Bors. and Tetris. Each game is assigned a score for each of the parameters from 0 to 5, and globally compared. Chrome Dino is the game which meets the requirements the most.

3.2. Chrome Dino

Chrome Dino is an endless runner game that is built-in in the Google Chrome web browser. The player controls a running dinosaur by jumping or ducking to avoid obstacles, which include cacti and pterodactyls. When a player collides with any of these obstacles, the game is over and a score is accumulated for the total distance traveled. Sometimes, the game switches from black graphics on a white background (representing daytime), to white graphics on a dark background (representing nighttime). At the beginning of the game, a period of time occurs when no obstacles are spawned yet, to let the player adapt to the pace of the game. The game starts at a moderate speed and slowly accelerates over the distance traveled.

3.3. Data Collection and Analysis

Human data will be collected through a custom variant of the Chrome Dino game, made available on a public website. Every new instance of the game will generate a new random version of the game, with different underlying game parameters that are randomly sampled from fixed ranges to make every game change in difficulty.

Game metrics, that log events during gameplay (such as the number of jumps or the achieved score), as well as the experience responses will be gathered through questionnaires in between games. For each consecutive pairs of two games A and B, subjects will report their preference for several emotional states: fun, frustrating and challenging. This is done with a 4-alternative forced choice (4-AFC) protocol. An assumption is made that a correlation exists between fun, frustrating and challenging in such that the latter two contribute to the first, i.e. frustrating has a negative contribution to fun, while challenging has a positive contribution to fun.

A descriptive analysis is be done to gain further insight into possible trends. The UMAP dimensionality reduction method is applied to visualize the data and discover relevant parameters. To discern different different groups of player personas that describe games, clustering techniques like k-means, agglomerative clustering and DBScan are applied. Classification methods can be attempted to build a model that can predict labels of player enjoyment.

4. Data Collection

Three types of data are collected:

1. **Difficulty Parameters.** Underlying game parameters that give each new game a different difficulty (e.g. obstacles generated, speed, acceleration).

2. **Game Metrics.** Upon occurrence of different events, game log are captured live during the game to encapsulate player behaviour(e.g. keyboard and mouse logs, total number of jumps, achieved score)

3. **Player's Experience.** The player's experience is evaluated through a questionnaire for the emotional states of fun, frustrating and challenge.

After playing two or more games, the player is be presented with the questionnaire form that evaluates the player's experience about the last two played games.

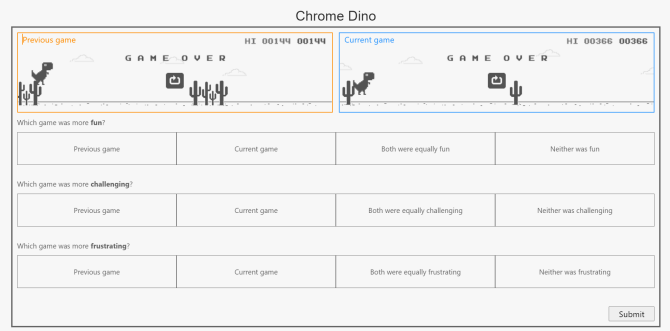


Figure 3: Chrome Dino form.

Data is collected over the Internet. The website is put available online at chromedino.ml. Subjects are recruited via social media and networks of colleagues, friends and family.

5. Player Modeling

A descriptive analysis was done, subdividing the results in positive evaluations for fun, frustrating and challenge. A game A is categorised as 'positively evaluated as X' (X = fun, frustrating, challenging) if it has been marked as being more X than another game B, or if it was one of either games evaluated as equally X. Visualisations for game metrics (achieved score and number jumps), showed skewed distributions, while game parameters were mostly evenly spread. However, combining a lot of the visualisations (clear time, score, number jumps and distance per jump), arise the suspection that a bi-modal distribution might occur, subdividing player behaviour into two groups of inexperienced and experienced players. Interesting observations could also be made about the perception of games in which a game over during night mode was present. Players also value high speeds to more fun and challenging.

To verify the assumption made that a relationship exists between the factors fun, frustrating and challenging, inter correlations are visualized again by the use of a heatmap. As expected, frustrating correlates to fun in a negative way, while challenging contributes to fun in a positive way.

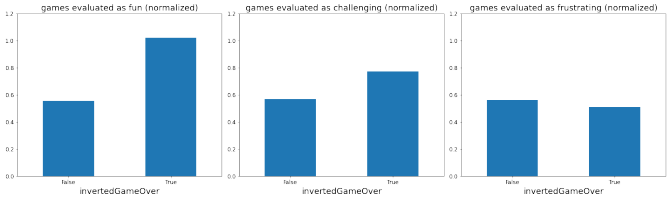


Figure 4: Normalized distributions for games that ended in night-mode (invertedGameOver) for different emotional states.

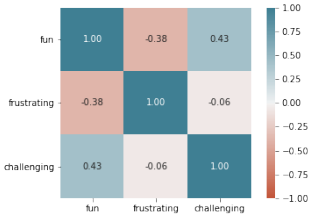


Figure 5: Correlations between fun, frustrating and challenging.

5.1. Dimensionality Reduction

UMAP was able to recognize three different groups. Two of these groups can be perceived as the main groups, with small overlap between them, while the third group is located at a larger isolated distance with no overlaps in any way. It could be observed that these groups have been divided based on the parameters inverted game over, min gap, max gap, and max speed or clear time.

Scores for positive evaluations for each emotional states have been mapped onto the UMAP distribution to analyse if there are any correlations between player behaviour and aspects of player experiences. The score is calculated by aggregating all positive evaluations for that emotional state and taking the mean over them.

To get an idea of a global score, the three emotional states have been combined into one formula that calculates an approximate score for total player enjoyment, based on perceived inter-correlations between the three emotional states, and is estimated as follows:

$$total = fun + 0.5 * challenging - 0.5 * frustrating \quad (1)$$

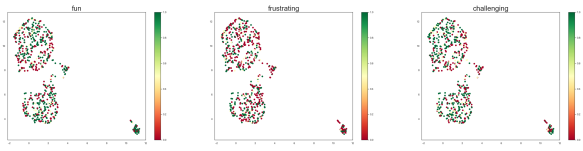


Figure 6: UMAP dimensionality reduction, with mapping of individual evaluations for all three emotional states.

Unfortunately, the outcomes are very spread out over the groups and show no real structure. If instead any structure was distinguishable, it would be possible to link this to the groups discerned, but sadly, this is out of the question.

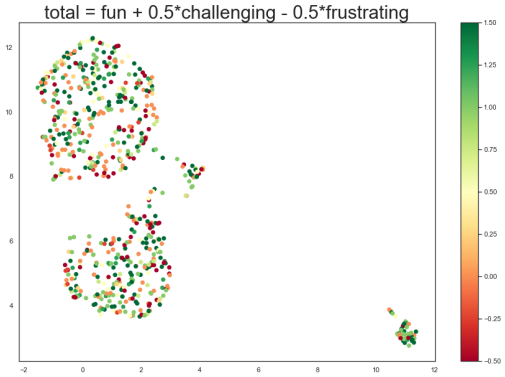
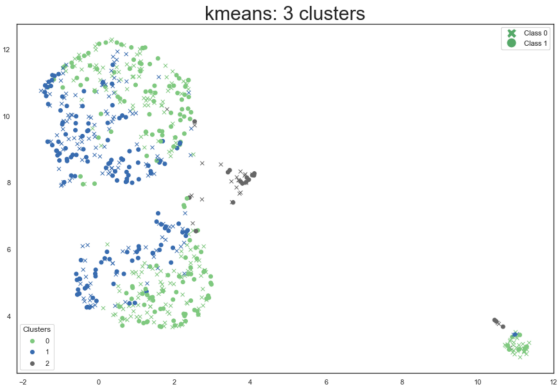


Figure 7: UMAP dimensionality reduction, with mapping of positive evaluation formula.

5.2. Clustering

K-means was able to detect the group of highest scoring participants, and further subdivide the two main groups. The partitioning seems to correlate most with the parameters min gap and speed.

To see how player experience correlates with player behaviour, the player experience is plotted with different markers for positive (class 1) and negative evaluations (class 0). The distribution for each score within the clusters is also calculated. These distributions are roughly equally divided. If a certain cluster exhibited uneven spreads, it could have given indication of correlations between experienced evaluations and player behaviour that was present within that group, but unfortunately, this is not up for discussion.



(a) K-means clustering: total score evaluation mapping.

cluster	score 0	score 1
1.0	48.59	51.4
0.0	41.96	58.03
2.0	42.85	57.14

(b) Class distribution per cluster.

Figure 8: K-means clustering: evaluations.

Other clustering techniques have been applied to attempt to discern clusters of different player behaviour, but all perform poorly.

5.3. Classification

A list of different classification techniques have been applied: logistic regression (with and without inner CV), decision trees, k-nearest neighbors, linear discriminant analysis, gaussian naive bayes and support vector machines. Decision trees are expected to perform good, since they are well-suited for this problem. The realized accuracies are very weak and are situated within the 50% to 60% range. To attempt to improve the results, ensemble methods have been applied that combine the predictions of several base estimators. These include gradient boosting, adaboost and histogram boosting. These methods perform slightly better, but only meagerly.

method	cv_accuracy	cv_standard_deviation	training_accuracy	test_accuracy
Logistic Regression	57.41	4.13	59.15	55.17
Logistic Regression CV	55.1	1.11	58.95	55.74
Decision Trees	59.15	3.21	99.8	48.85
Decision Trees Unscaled	55.67	3.75	99.8	51.14
K Nearest Neighbors	53.94	3.3	68.2	51.14
Linear Discriminant Analysis	57.79	4.31	58.95	55.74
Gaussian Naive Bayes	53.38	6.37	64.73	52.29
Support Vector Machines	56.64	1.14	72.44	57.47
Random Forest	53.95	0.64	65.51	56.32
Gradient Boosting	56.06	3.91	79.96	54.02
AdaBoost	56.26	2.1	79.38	55.17
Histogram Boosting	53.95	3.5	99.8	53.44

Figure 9: Classification methods and accuracy.

5.4. Conclusion

A few interesting observations could be made from descriptive analysis which led to intuitive assumptions about player behaviour and experience. Unfortunately, none of these assumptions could be validated with the succeeding methods. UMAP was able to reduce the space into three groups that were based mostly on game parameters, but was unable to generalize from game metrics. Neither dimensionality reduction methods, clustering or classification methods seem to be able to reduce the space into groups of player behaviour, but lean towards trends in game difficulty parameters. This could be due to the fact that a relatively small amount of player metrics that describe gameplay features are gathered, which are not able to provide enough information about player behaviour. Due to the nature of the simplicity of the selected game, no other game metrics could have been gathered for Chrome Dino. A different game could have given opportunity to more gathered game metrics and generalizations. Upon retrospective, the number of game metrics could have been a fitting parameter to taken into consideration during game selection. A player model that includes more behavioral aspects could yield interesting observations.

Other reasons could be that the data is too noisy. Capturing subjective emotions in an unsupervised environment is difficult and prone to errors. Even though the data was thoroughly preprocessed and participants were only asked three questions that are answered with one press of a button, a lot of noise could still be present. It is also possible that not enough sample points are collected for global generalisations to be made from the gathered data.

6. Conclusion and future work

In this work, an attempt was made at constructing player models for player behaviour in Chrome Dino, a web-based video game, using AI methods. Data collection was done on human subjects over the internet to gain insight into relationships between game difficulty parameters, combined with game metrics and player enjoyment. Multiple methods were applied to analyze player behaviour, including dimensionality reduction using UMAP, clustering and classification. No clear relationships were found. Common flaws point towards a lack of quantity of game metrics that capture enough information about player behaviour, too noisy data, or not enough player samples.

References

[1] Thomas W.Malone, What makes computer games fun? (r 1981) 258–277.

[2] M. Csikszentmihalyi, Flow: the Psychology of Optimal Experience (Harper Collins).

[3] Robin Hunicke, Vernell Chapman , AI for Dynamic Difficulty Adjustment in Games .

[4] M. Zohaib, Dynamic difficulty adjustment (dda) in computer games: A review, Advances in Human-Computer Interaction 2018 (2018) 1–12. doi:10.1155/2018/5681652.

[5] G. Sepulveda, F. Besoain, N. A. Barriga, Exploring dynamic difficulty adjustment in videogames (2019) 1–6doi:10.1109/CHILECON47746.2019.8988068.

[6] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, Springer, 2018, http://gameaibook.org.

[7] R. Likert, A technique for measurement of attitudes, Archives of Psychology 22 (1932) 1–.

[8] G. Yannakakis, H. Martínez, Ratings are overrated!, Frontiers in ICT 2 (07 2015). doi:10.3389/fict.2015.00013.

[9] Yannakakis G.N., Hallam J., Ranking vs. Preference: A Comparative Study of Self-reporting., Lecture Notes in Computer Science 6974 (2011).

[10] C. Pedersen, J. Togelius, G. N. Yannakakis, Modeling player experience in super mario bros (2009) 132–139doi:10.1109/CIG.2009.5286482.

[11] A. P. Andy Coenen, Understanding umap, https://pair-code.github.io/understanding-umap/.

[12] Anders Drachen, Christian Thurau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage., Game Data Mining, Game Analytics (2013) 205–253.

[13] IJsselsteijn, W. A., de Kort, Y. A. W., & Poels, K., The Game Experience Questionnaire. (2013).

Modelleren van Spelervaring in Webgebaseerde Videospellen

Yasmine Bogaert[1], Prof. dr. ir. Francis wyffels[2], Andreas Verleysen[2]

Abstract

De videogame industrie is de afgelopen jaren veel gegroeid, samen met de technologieën die daarbij komen kijken. Toch staat één aspect vrij stil, de moeilijkheidsgraad. Voor sommige spelers worden zelfs de huidige best ontworpen spellen vaak snel te gemakkelijk, terwijl ze voor anderen frustrerend en moeilijk zijn. Een voorgesteld antwoord voor dit probleem is dynamic difficulty adjustment (DDA), waarbij de moeilijkheidsgraad van het spel dynamisch wordt aangepast aan het vaardigheidsniveau van de speler. Om een goede realisatie van DDA te garanderen, is een goed spelersmodel vereist dat de relatie weergeeft tussen de ervaring van de speler en de onderliggende spelparameters. Een adaptief spel dat is gebaseerd op nauwkeurige spelersmodellen, is in staat om voorspellingen te doen over toekomstig gedrag, het spel aan die veranderingen aan te passen en spelers beter te leiden naar inhoud waarvan wordt verwacht dat ze het aangenaam zullen vinden. Dit artikel gaat in op spelermodellering voor het spel Chrome Dino, in de context van DDA.

Keywords: Dynamic Difficulty Adjustment, Game Difficulty, Flow, Player Experience, Player Modeling, Chrome Dino.

1. Introductie

Men kan zich afvragen wat spelers geïnteresseerd houdt en doet terugkomen naar spellen. Zelfs met concepten zoals gamification, waarbij concepten uit games (zoals beloningen, rankings, puntensystemen) worden gebruikt in andere omgevingen om de gebruikerservaring te verbeteren, is een spel maken met een gepaste spelervaring geen gemakkelijke opgave. Spelers beginnen zich vaak te vervelen als een spel te makkelijk wordt, of geraken gefrustreerd als een spel te moeilijk wordt. De belangrijkste factor is het balanceren van de moeilijkheidsgraad van een spel, zodat het overeenkomt met de vaardigheid van de speler. Gamification en andere soortgelijke concepten zijn gebaseerd op algemene aspecten van de psychologie die alleen betrekking hebben op brede trends. Het spelerspubliek is breed verspreid, net als het individuele vaardigheidsniveau van spelers, variërend van nieuwe spelers tot spelers met veel ervaring.

In dit werk wordt een poging gedaan om spelermodellen te construeren voor het gedrag van spelers in het webgebaseerde spel Chrome Dino. Eerst wordt een literatuurstudie gegeven over dynamic difficulty adjustment, spelers ervaring, spelermodellering, hoe spelerservaring te meten en technieken voor spelermodellering. Ten tweede wordt er data verzameld om inzicht te geven in hoe spelersgedrag zich verhoudt tot spelerservaring. Daarna worden verschillende technieken voor spelermodellering toegepast, gebruikmakend van de verzamelde data. Ten slotte worden de resultaten geanalyseerd en wordt een zicht gegeven op toekomstig werk.

[1] Faculty of Sciences, Ghent University
[2] Faculty of Engineering and Architecture, Ghent University

2. Literatuur

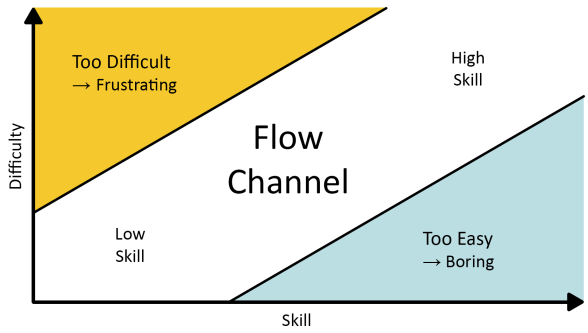
2.1. Dynamic Difficulty Adjustment

Dynamic difficulty adjustment is een techniek die in spellen wordt gebruikt om de moeilijkheidsgraad van het spel aan te passen aan het vaardigheidsniveau van de speler om ervoor te zorgen dat de speler zich niet verveelt (wanneer het spel te gemakkelijk is) of gefrustreerd raakt (wanneer het spel te moeilijk is). De essentie van DDA is om de gebruiker geïnteresseerd te houden en tegelijkertijd een aangenaam uitdagingsniveau te bieden. DDA wordt gedaan tijdens de uitvoering, waarbij de acties van de speler worden gevolgd en het spel daaraan wordt aangepast om de speler de juiste uitdagingen te bieden om een goede match te garanderen tussen de vaardigheid van een speler en de moeilijkheidsgraad van het spel. In de literatuur zijn verschillende technieken te vinden om dynamic difficulty adjustment aan te pakken. Veelgebruikte technieken voor DDA zijn onder meer reinforcement learning (RL), artificial neural networks (ANN), genetic algorithms or dynamic scripting.

2.2. Spelerservaring

Er zijn verschillende psychologische onderzoeken geweest die proberen te identificeren wat 'leuk' is in een spel en wat mensen ertoe aanzet om videogames te spelen. Deze studies omvatten Malone's principes van intrinsieke kwalitatieve factoren voor interessante gameplay [1], namelijk uitdaging, nieuwsgierigheid en fantasie, evenals het concept van de flowtheorie van Csikszentmihalyi [2].

Het concept van flow definieert spelerstoestanden in twee dimensies: vaardigheid en uitdaging. Het relateert de moeilijkheid van een taak aan de perceptie van de uitdaging door de speler. Flow is een toestand waarin een balans tussen vaardigheid en uitdaging wordt bereikt, in de volksmond vaak



Figuur 1: Flow.

omschreven als zijnde in "the Zone". Wanneer de moeilijkheidsgraad van het spel hoger is dan de vaardigheid van de speler, wordt het spel frustrerend, waardoor de speler in een staat van angst raakt. Aan de andere kant, wanneer de vaardigheid van de speler hoger is dan de moeilijkheidsgraad, is het spel te gemakkelijk en drijft het de speler in een toestand van verveling. Het doel is om de speler in een toestand van flow te houden en weg te houden van toestanden waar het spel veel te uitdagend of veel te gemakkelijk is. [3, 4, 5]

2.3. Spelersmodellering

Het primaire doel van spelermodellering is om te begrijpen hoe de interactie van een speler met een spel een effect heeft op de individuele ervaring van de speler. We proberen inzicht te krijgen in de cognitieve, affectieve en gedragspatronen van spelers [6].

Er zijn twee belangrijke factoren die een rol spelen: het gedrag van de speler en de ervaring van de speler. Spelersgedrag gaat over wat een speler doet in een game, terwijl spelerervaring verwijst naar hoe een speler zich voelt tijdens het spelen. Om een nauwkeurig spelersmodel te construeren, worden zowel het spelersgedrag als de spelerservaring gemeten. Deze vormen samen de input voor het spelersmodel.

2.4. Spelerservaring Opmeten

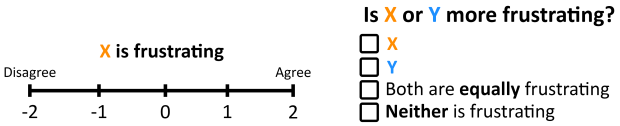
Er zijn verschillende manieren om de ervaring van spelers te meten. Labelen kan worden gedaan door te kijken naar de emotionele reacties van spelers door middel van fysiologische metingen (gezichtsuitdrukkingen, spieractivatie, hersengolven, houding of spraak) of door handmatige annotatie, hetzij door de speler zelf of door een derde persoon. De toegewezen labels moeten idealiter zo dicht mogelijk bij de grondwaarheid liggen. De nauwkeurigheid van handmatige annotaties wordt regelmatig in twijfel getrokken omdat er tal van factoren zijn die bijdragen aan een mogelijke afwijking tussen een label en de werkelijke onderliggende spelerservaring [6]. De meest gebruikelijke techniek om ervaringen te labelen is door middel van een vragenlijst.

Spelerservaringen kunnen worden geëvalueerd met vrije antwoorden (bijv. hardop denken) of geforceerde antwoorden (bijv. vragenlijsten). Vrije antwoorden bevatten meer gedetailleerde informatie, maar zijn vaak ongestructureerd, chaotisch en daardoor moeilijk te analyseren. Aan de andere kant zorgt het dwingen van spelers om hun ervaringen zelf te rapporteren met behulp van gerichte vragen, hen tot specifieke op voorhand gekozen opties.

Er zijn verschillende manieren om gegevens over spelerservaringen te structureren: door middel van ratings, klassen of rankings. Ratings vragen een speler om te kiezen tussen een scalaire waarde of een vector van waarden om zijn mening over een bepaalde uitspraak aan te geven. Tegenwoordig zijn vragenlijsten de meest gebruikte manier om verschillende aspecten van de gebruikerservaring te beoordelen. De meest populaire rating vragenlijsten maken gebruik van vragen in de vorm van een Likert-schaal [7], die de gebruiker vraagt om aan te geven in hoeverre hij het eens (of oneens) is met een bepaalde stelling, binnen een reeks vooraf gedefinieerde punten. Rating gebaseerde vragen zijn inherent aan beperkingen, die vaak over het hoofd worden gezien [8, 9, 6]. Ratings worden vaak geanalyseerd door hun waarden voor alle deelnemers te vergelijken, waarbij het bestaan van interpersoonlijke verschillen buiten beschouwing wordt gelaten, maar de invulling van elk niveau op een beoordelingsschaal tussen verschillende deelnemers kan erg verschillen. Ratings worstelen ook met primacy- en recency-effecten, waarbij informatie aan het begin en aan het einde beter wordt onthouden dan informatie in het midden. Bovendien worden ratings gewoonlijk omgezet in ordinale waarden, terwijl in de meeste vragenlijsten labels worden weergegeven als afbeeldingen of bijvoeglijke naamwoorden, wat in strijd is met de basisaxioma's van statistieken.

Klassen vragen spelers om te kiezen uit een set van twee of meer opties. Klassen hebben minder statistische beperkingen in vergelijking met beoordelingen, maar bereiken niet hetzelfde detailniveau [6].

Vragen in de vorm van rankings, vragen de speler om een voorkeursrangschikking te maken tussen twee of meer opties. In het geval van paarsgewijze voorkeuren wordt de speler gevraagd om twee opties te vergelijken en aan te geven aan welke hij de voorkeur geeft. Bij meer dan twee opties worden de deelnemers gevraagd een rangschikking te geven van enkele of alle opties. Rankings hebben geen last van de bovengenoemde beperkingen die voorkomen in ratings of klassen.



(a) Ratings op een 5-punt Likert-schaal

(b) Rankings met 4-alternative forced choice

Figuur 2: Voorbeelden van ratings vs rankings.

Gelijkaardige vragenlijsten vragen spelers vaak om spellen van elk spelpaar te rangschikken met betrekking tot plezier, uitdaging, verveling, frustratie, opwinding, angst en ont-

spanning. De selectie van deze zeven ervaringen is gebaseerd op hun relevantie voor het spelen van computerspellen en de spelerservaring [9]. Kleinere onderzoeken richten zich alleen op drie emoties: plezier, uitdaging en frustratie [10].

2.5. Technieken voor Spelermodellering

De volgende methoden worden onderzocht: dimensionaliteitsreductie, clustering and classificatie.

Dimensionaliteitsreductie transformeert gegevens van een hoogdimensionale ruimte in een laagdimensionale ruimte, zodat de laagdimensionale weergave betekenisvolle eigenschappen van de originele gegevens behoudt. De gegevens kunnen dan worden gevisualiseerd en beter worden geïnterpreteerd. Technieken voor dimensionaliteitsreductie omvatten t-SNE en UMAP. Hoewel zowel UMAP als t-SNE enigszins vergelijkbare resultaten opleveren, maken de verhoogde snelheid, het betere behoud van de globale structuur en de meer begrijpelijke parameters van UMAP een betere methode [11].

Clustering groepeert een set objecten zodanig dat objecten in hetzelfde cluster meer op elkaar lijken dan op die in andere clusters. In termen van spelermodellering is het de classificatie van spelersgedrag in een aantal clusters, afhankelijk van hun gedragskenmerken. Clustering is niet alleen relevant voor de personalisatie van het spel, maar ook voor het begrijpen van het gedrag van spelers in samenhang met het spelontwerp [12]. Veelgebruikte clustertechnieken omvatten k-means, DBScan, spectrale clustering of Gaussian mixture models.

Classificatie leert klassenlabels voor nieuwe gegevens te voorspellen, op basis van voorbeeldgegevens. Wat spelermodellering betreft, bestaan de voorbeelden uit gegevens van spelersgedrag, gecombineerd met een label van spelerservaring (bijv. plezier). Het doel is de indeling van het gedrag van nieuwe spelers in klassen van spelerservaring. Gebruikelijke classificatietechnieken omvatten beslissingsbomen, k-nearest neighbors, logistische regressie of support vector machines. Ensemble-methoden nemen een extra stap en combineren de voorspellingen van verschillende basistechnieken om de generaliseerbaarheid en robuustheid van een alleenstaande methode te verbeteren. Gebruikelijke technieken voor ensemble-methoden zijn onder meer random forests en gradient boosting.

3. Methodologie

3.1. Spelselectie

Om een spel te selecteren, zijn volgende parameters bekeken:

1. **Duur.** Kortere spellen zijn beter, om te vermijden dat een verlies aan het behoudt van informatie voorkomt, om spelers geïnteresseerd te houden en om zoveel mogelijk data als mogelijk te verzamelen.
2. **Aantal spelelementen.** Een spel met een goede balans in aantal elementen, niet te veel maar ook niet te weinig, zou ideaal zijn.
3. **Implementeerbaarheid.** Een open source model is beschikbaar of het is mogelijk om het spel zelf opnieuw te maken binnen praktisch haalbare tijd.
4. **Verwachtte populariteit.** Het spel moet een zo groot mogelijk doelpubliek hebben.

Een lijst van spellen is verzameld die voldoen aan ten minste een van de bovengenoemde parameters. Deze spellen zijn onder meer: Chrome Dino, Pacman, Flappy Bird, Snake, Space Invaders, Pong, Super Mario Bors. en Tetris. Elk spel krijgt een score toegewezen voor elk van de parameters van 0 tot 5 en wordt globaal vergeleken. Chrome Dino is het spel dat het meest aan de eisen voldoet.

3.2. Chrome Dino

Chrome Dino is een endless runner spel dat is ingebouwd in de Google Chrome-webbrowser. De speler bestuurt een rennende dinosaurus door te springen of te bukken om obstakels te vermijden, waaronder cactussen en pterodactyls. Wanneer een speler tegen een van deze obstakels botst, is het spel afgelopen en wordt een score verzameld voor de totale afgelegde afstand. Soms schakelt het spel over van zwarte afbeeldingen op een witte achtergrond (dag), naar witte afbeeldingen op een donkere achtergrond (nacht). Aan het begin van het spel treedt er een periode op waarin er nog geen obstakels worden gegenereerd, zodat de speler zich kan aanpassen aan het tempo van het spel. Het spel begint met een matige snelheid en versnelt langzaam over de afgelegde afstand.

3.3. Data Verzameling en Analyse

Gegevens worden verzameld via een aangepaste variant van de Chrome Dino-game, die beschikbaar wordt gesteld op een publieke website. Elk nieuw exemplaar van het spel genereert een nieuwe willekeurige versie van het spel, met verschillende onderliggende spelparameters die willekeurig worden gesampled uit vaste gekozen reeksen om elk spel in moeilijkheidsgraad te veranderen.

Spelmetrieken, die gebeurtenissen tijdens het spelen registreren (zoals het aantal sprongen of de behaalde score), evenals de ervaringsreacties worden verzameld via vragenlijsten tussen games. Voor elk opeenvolgend paar van twee games A en B, zullen de proefpersonen hun voorkeur voor verschillende emotionele toestanden aangeven: leuk, frustrerend en uitdagend. Dit gebeurt met een 4-alternative forced choice (4-AFC) protocol. Er wordt een assumptie gemaakt dat er een verband bestaat tussen plezier, frustratie en uitdaging in die zin dat de laatste twee bijdragen aan het eerste, d.w.z. frustratie heeft een negatieve bijdrage aan plezier, terwijl uitdaging een positieve bijdrage heeft aan plezier.

Om meer inzicht te krijgen in mogelijke trends wordt een beschrijvende analyse gedaan. De UMAP dimensionaliteitsreductiemethode wordt toegepast om de gegevens te visualiseren en relevante parameters te ontdekken. Om verschillende groepen van spelerpersona's te onderscheiden die games beschrijven, worden clusteringstechnieken zoals k-means, agglomeratieve clustering en DBScan toegepast. Classificatiemethoden kunnen worden geprobeerd om een model te bouwen dat labels van spelerservaringen kan voorspellen.

4. Data Verzameling

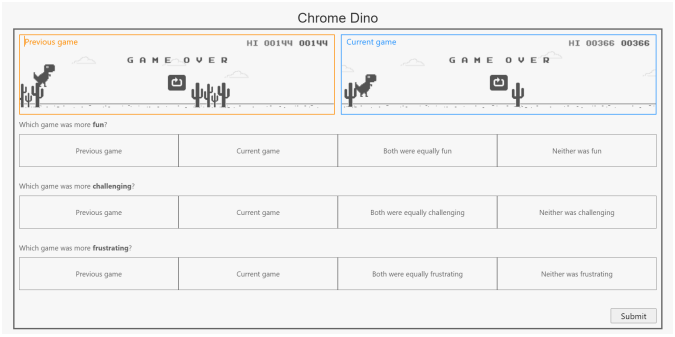
Drie verschillende soorten data worden verzameld:

1. **Moeilijkheidsparameters.** Onderliggende spel parameters die ervoor zorgen dat elk spel een andere moeilijkheidsgraag krijgt (bijv. gegenereerde obstakels, snelheid, versnelling).

2. **Spelmetrieken.** Bij het optreden van verschillende eventsm worden spel logs verzameld om het spelgedrag te omschrijven (bijv. toetsenbord en muis logs, total aantal sprongen, behaalde score)

3. **Spelerservaring.** De spelerservaring wordt geëvalueerd met een vragenlijst voor de verschillende emotionele toestanden: plezier, frustratie en uitdaging.

Na het spelen van twee of meer games, krijgt de speler het vragenlijstformulier te zien dat de ervaring van de speler over de laatste twee gespeelde games evalueert.

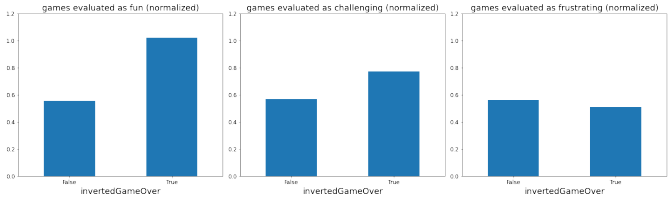


Figuur 3: Chrome Dino formulier.

Gegevens worden verzameld via internet. De website is online beschikbaar op `chromedino.ml`. Proefpersonen worden geworven via sociale media en netwerken van collega's, vrienden en familie.

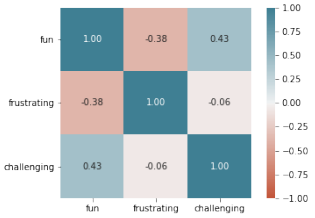
5. Spelermodellering

Er is een beschrijvende analyse gedaan, waarbij de resultaten zijn onderverdeeld in positieve evaluaties voor plezier, frustratie en uitdaging. Een spel A wordt gecategoriseerd als 'positief beoordeeld als X' (X = leuk, frustrerend, uitdagend) als het is gemarkeerd als meer X dan een ander spel B, of als het een van beide games was die als even veel X werden beoordeeld. Spelermetrieken (behaalde score en aantal sprongen) vertonen scheve verdelingen, terwijl spelparameters grotendeels gelijkmatig verdeeld zijn. Door veel van de visualisaties te combineren (begin tijd, score, aantal sprongen en afstand per sprong), ontstaat echter het vermoeden dat er een bimodale verdeling kan optreden, waarbij het spelersgedrag wordt onderverdeeld in twee groepen van onervaren en ervaren spelers. Ook konden interessante observaties worden gedaan over de ervaring van spellen waarin een nachtmodus aanwezig was. Spelers evalueren hoge snelheden ook als leuker en uitdagender.



Figuur 4: Genormaliseerde distributies foor spelen die beëindigd zijn in nacht modus (invertedGameOver), voor verschillende emotionele toestanden.

Om de assumptie te verifiëren dat er een relatie bestaat tussen de factoren leuk, frustratie en uitdaging, worden de onderlinge correlaties gevisualiseerd met behulp van een heatmap. Zoals verwacht correleert frustratie met plezier op een negatieve manier, terwijl utidaging op een positieve manier bijdraagt aan plezier.



Figuur 5: Correlaties tussen plezier, frustratie en uitdaging.

5.1. Dimensionaliteitsreductie

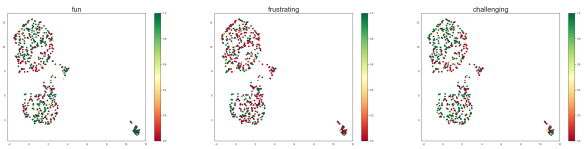
UMAP kon drie verschillende groepen onderscheiden. Twee van deze groepen kunnen worden gezien als de hoofdgroepen, met een kleine overlap tussenin, terwijl de derde groep zich op een grotere geïsoleerde afstand bevindt en op geen enkele manier overlapt. Het kan worden opgemerkt dat deze groepen zijn verdeeld op basis van de parameters voor inverted game over, min gap, max gap en max speed of clear time.

Scores voor positieve evaluaties voor elke emotionele toestand zijn in kaart gebracht op de UMAP-distributie om te analyseren of er correlaties zijn tussen spelersgedrag en aspecten van spelerservaringen. De score wordt berekend

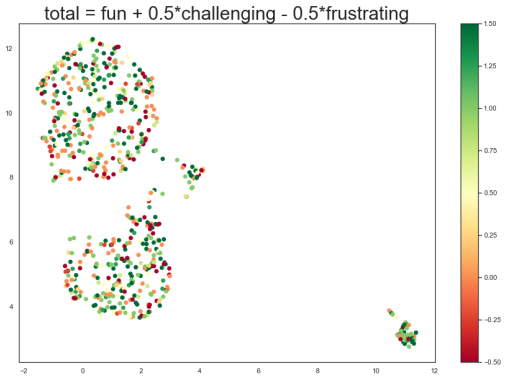
door alle positieve evaluaties voor die emotionele toestand bij elkaar op te tellen en daar het gemiddelde over te nemen.

Om een idee te krijgen van een globale score, zijn de drie emotionele toestanden gecombineerd in één formule die een geschatte score berekent voor het totale spelersplezier, gebaseerd op waargenomen onderlinge correlaties tussen de drie emotionele toestanden, en wordt als volgt berekend:

$$totaal = plezier + 0.5 * uitdaging - 0.5 * frustratie \quad (1)$$



Figuur 6: UMAP dimensionaliteitsreductie, met mapping voor individuele evaluatie voor alle emotionele toestanden.



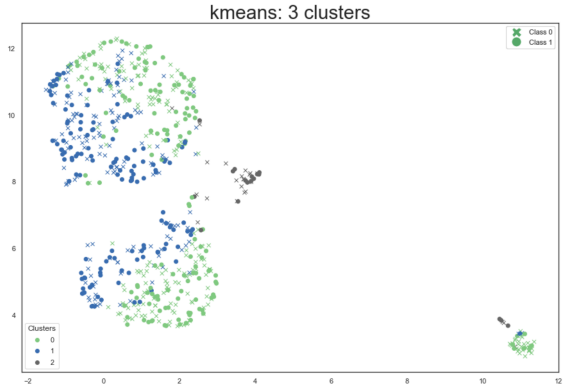
Figuur 7: UMAP dimensionaliteitsreductie, met mapping voor positieve evaluatie formule.

Helaas zijn de uitkomsten erg verspreid over de groepen en vertonen ze geen echte structuur. Als in plaats daarvan enige structuur te onderscheiden zou zijn, zou het mogelijk zijn om deze te koppelen aan de onderscheiden groepen, maar helaas is dit uitgesloten.

5.2. Clustering

K-means kon de groep met de hoogst scorende deelnemers detecteren en de twee hoofdgroepen verder onderverdelen. De partitionering lijkt het meest te correleren met de parameters min gap en speed.

Om te zien hoe de spelerservaring correleert met het spelersgedrag, is de spelerservaring uitgezet met verschillende markeringen voor positieve (klasse 1) en negatieve evaluaties (klasse 0). Ook wordt de verdeling per score binnen de clusters berekend. Deze verdelingen zijn ongeveer gelijk verdeeld. Als een bepaald cluster ongelijke spreidingen vertoonde, had dat een indicatie kunnen geven van correlaties tussen ervaren evaluaties en spelersgedrag dat binnen die groep aanwezig was, maar dit staat helaas niet open voor discussie.



(a) K-means clustering: totale score evaluatie mapping.

cluster	score 0	score 1
1.0	48.59	51.4
0.0	41.96	58.03
2.0	42.85	57.14

(b) Verdeling van klasses per cluster.

Figuur 8: K-means clustering: evaluaties.

Andere clusteringstechnieken zijn toegepast om te proberen clusters van verschillend spelersgedrag te onderscheiden, maar presteren allemaal slecht.

5.3. Classificatie

Een lijst van verschillende classificatietechnieken is toegepast: logistische regressie (met en zonder innerlijke CV), beslissingsbomen, k-nearest neighbors, lineaire discriminantanalyse, Gaussian naive bayes en support vector machines. Van beslisbomen wordt verwacht dat ze goed presteren, omdat ze zeer geschikt zijn voor dit probleem. De gerealiseerde nauwkeurigheden zijn zeer zwak en liggen binnen het bereik van 50% tot 60%. Om te proberen de resultaten te verbeteren, zijn ensemble-methoden toegepast die de voorspellingen van verschillende basisschatters combineren. Deze omvatten boosting, adaboost and histogram boosting. Deze methoden presteren iets beter, maar slechts in kleine mate.

5.4. Conclusie

Een paar interessante observaties konden worden gedaan uit de beschrijvende analyse die leidde tot intuïtieve veronderstellingen over het gedrag en de ervaring van spelers. Helaas kon geen van deze veronderstellingen worden gevalideerd met de daaropvolgende methoden. UMAP was in staat om de ruimte te verdelen in drie groepen die voornamelijk waren gebaseerd op spelparameters, maar kon niet

method	cv_accuracy	cv_standard_deviation	training_accuracy	test_accuracy
Logistic Regression	57.41	4.13	59.15	55.17
Logistic Regression CV	55.1	1.11	58.95	55.74
Decision Trees	59.15	3.21	99.8	48.85
Decision Trees Unscaled	55.67	3.75	99.8	51.14
K Nearest Neighbors	53.94	3.3	68.2	51.14
Linear Discriminant Analysis	57.79	4.31	58.95	55.74
Gaussian Naive Bayes	53.38	6.37	64.73	52.29
Support Vector Machines	56.64	1.14	72.44	57.47
Random Forest	53.95	0.64	65.51	56.32
Gradient Boosting	56.06	3.91	79.96	54.02
AdaBoost	56.26	2.1	79.38	55.17
Histogram Boosting	53.95	3.5	99.8	53.44

Figuur 9: Classificatie methoden en nauwkeurigheden.

generaliseren op basis van spelermetrieken. Noch dimensionaliteitsreductiemethoden, clustering- of classificatiemethoden lijken in staat te zijn de ruimte in groepen van spelersgedrag te reduceren, maar neigen zich naar spelmoeilijkheidsparameters. Dit kan toe te wijten zijn aan het feit dat er een relatief kleine hoeveelheid spelerstatistieken is verzameld die spelerskenmerken beschrijven, die niet voldoende informatie kunnen verschaffen over het gedrag van de speler. Vanwege de eenvoudigheid van het geselecteerde spel, konden er geen andere spelermetrieken worden verzameld voor Chrome Dino. Een ander spel had de mogelijkheid kunnen bieden voor meer verzamelde spelermetrieken en generalisaties. Achteraf gezien zou het aantal spelermetrieken een passende parameter kunnen zijn om in overweging te nemen tijdens de spelselectie. Een spelersmodel met meer gedragsaspecten zou interessante observaties kunnen opleveren.

Andere redenen kunnen zijn dat de gegevens te veel ruis bevatten. Het vastleggen van subjectieve emoties in een omgeving zonder toezicht is moeilijk en heel vatbaar voor fouten. Ondanks dat de data grondig zijn voorbewerkt en de deelnemers slechts drie vragen kregen die met één druk op de knop beantwoord werden, kon er toch veel ruis aanwezig zijn. Het is ook mogelijk dat er niet genoeg steekproefpunten worden verzameld om globale generalisaties te maken van de verzamelde gegevens.

6. Conclusie en toekomstig werk

In dit werk is een poging gedaan om spelersmodellen te construeren voor het gedrag van spelers in Chrome Dino, een webgebaseerde videogame, met behulp van AI-methoden. Er is via internet gegevens verzameld om inzicht te krijgen in de relaties tussen de moeilijkheidsgraad van het spel, gecombineerd met spelermetrieken en het spelplezier. Er zijn meerdere methoden toegepast om het gedrag van spelers te analyseren, waaronder dimensionaliteitsreductie met behulp van UMAP, clustering en classificatie. Er werden geen duidelijke verbanden gevonden. Veelvoorkomende tekortkomingen kunnen wijzen op een gebrek aan hoeveelheid spelermetrieken die voldoende informatie over het gedrag van spelers vastleggen, te veel ruis in de gegevens of niet genoeg steekproefpunten.

Referenties

[1] Thomas W.Malone, What makes computer games fun? (r 1981) 258–277.

[2] M. Csikszentmihalyi, Flow: the Psychology of Optimal Experience (Harper Collins).

[3] Robin Hunicke, Vernell Chapman , AI for Dynamic Difficulty Adjustment in Games .

[4] M. Zohaib, Dynamic difficulty adjustment (dda) in computer games: A review, Advances in Human-Computer Interaction 2018 (2018) 1–12. doi:10.1155/2018/5681652.

[5] G. Sepulveda, F. Besoain, N. A. Barriga, Exploring dynamic difficulty adjustment in videogames (2019) 1–6doi:10.1109/CHILECON47746.2019.8988068.

[6] G. N. Yannakakis, J. Togelius, Artificial Intelligence and Games, Springer, 2018, http://gameaibook.org.

[7] R. Likert, A technique for measurement of attitudes, Archives of Psychology 22 (1932) 1–.

[8] G. Yannakakis, H. Martínez, Ratings are overrated!, Frontiers in ICT 2 (07 2015). doi:10.3389/fict.2015.00013.

[9] Yannakakis G.N., Hallam J., Ranking vs. Preference: A Comparative Study of Self-reporting., Lecture Notes in Computer Science 6974 (2011).

[10] C. Pedersen, J. Togelius, G. N. Yannakakis, Modeling player experience in super mario bros (2009) 132–139doi:10.1109/CIG.2009.5286482.

[11] A. P. Andy Coenen, Understanding umap, https://pair-code.github.io/understanding-umap/.

[12] Anders Drachen, Christian Thurau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage., Game Data Mining, Game Analytics (2013) 205–253.

[13] IJsselsteijn, W. A., de Kort, Y. A. W., & Poels, K., The Game Experience Questionnaire. (2013).

Vulgarizing Summary

Game designers nearly always strive to create games in which the difficulty of a game is appropriate for the player's skill level. Even so, designers are usually not able to accommodate every player's skill level, and frustrating mismatches between a player's skill and a game's difficulty are common. Players begin games at different skill levels, and through playing develop their own skill at their own rates. For some players, even the current best-designed games become uninterestingly easy, while frustrating and difficult for others. As a player's skill improves through practice, a well-designed game will present more difficulties so that the player is never bored by overly easy gameplay or frustrated by overly difficult gameplay. Most of the time, this difficulty and the parameters that come along with it, can only be determined at the beginning of the game by selecting a difficulty level, often done by the player himself/herself. Still, this can lead to a frustrating experience for both experienced and inexperienced gamers, as they attempt to follow a preselected learning or difficulty curve composed by game developers.

A proposed answer to this challenge is dynamic difficulty adjustment (DDA), which is the process of changing underlying parameters in a video game in real-time, based on the player's observed ability, to balance the game difficulty. This is done in order to avoid making the player bored (if the game is too easy) or frustrated (if it is too hard) and better fit the according skill level of the player. DDA is done during execution, tracking the player's performance and adjusting the game to present proper challenges to the player to ensure a good match between a player's skill and the game's difficulty. Some elements (game parameters) of a game that might be changed via DDA include: number of enemies, number of obstacles, generated items or duration of gameplay experience.

To ensure a proper realization of DDA, a good player model is required. This player model captures generalizations from player behaviour, combined with which player experience they provoke. Examples of elements (game metrics) that can capture player behaviour during the game are: number of jumps, collected items, number of deaths or number of opponents killed. These game metrics, in combination with the game parameters and the perceived player experience, are the inputs for constructing the player model. A good player model will enable to predict player behavior in new situations that may have never been observed before. This model is then combined with an algorithm to adapt the game content based on those observations. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy.

In an attempt to analyze difficulty parameters, model player behavior, and strive towards a system for DDA, we look at the video game Chrome Dino, an endless runner game in which the player controls a running dinosaur that can jump or duck to avoid obstacles. A custom version of this game is set up, with various sampled difficulty parameters and the player's response is measured with a ranked questionnaire in order to determine whether there were statistically significant differences in the perception of gameplay. Player personas and player models are analyzed with various methods, including dimensionality reduction, clustering and classification.

Datasets can contain a lot of parameters, which result in large dimensionalities (often 15 or more). These are unable to be visualized directly in two or three dimensions. One approach to simplification is to assume that the data of interest lies within lower-dimensional space. Dimensionality reduction techniques build on this assumption and transform the data from a high-dimensional space into a low-dimensional space that still retains some meaningful properties of the original data. This way, they are able to visualize and understand large, high dimensional datasets.

Clustering is the grouping of a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. In terms of player modeling, we want to group player behavior into a number of clusters depending on their behavioural attributes, so that the player's in each group show similar behaviour. This could lead to different player persona's, such as for example, the inexperienced versus experienced player, or players that prefer a fast paced game versus a slow paced game.

The goal of classification is to predict labels for new data based on gathered sample data. For player modeling, we want to predict if a certain configuration of game difficulty parameters will be enjoyed by the player subject. The algorithm is given the dataset of game parameters and game metrics, each combined with a corresponding label indicating the degree of player enjoyment. This input is processed and generalized to global trends. When a new sample is presented, the algorithm will try to match it with what it sees as the most fitting label, based on other labels from which it has already learned.

Acknowledgement

I would like to express my gratitude to my promotor, Francis wyffels, and counsellor, Andreas Verleysen. Not only for the opportunity I was given to research such an interesting topic in the domain of video games, I was also given continued motivational support and guidance throughout this project. Without the numerous meetings, supervision of the directions I was taking, proofreading of different iterations of text and multitude of suggestions, I would not have been able to achieve a thesis of the shape that it is in now.

I would also like to thank my friends and family, for the emotional support throughout this long process, as well as fellow students who helped me out when I was in need of specialized advice on certain topics.

Yasmine Bogaert
June 2021

Permission for Usage

The author gives permission to make this master's thesis available for consultation and to copy parts of this master's thesis for personal use. In the case of any other use, the copyright terms have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master's thesis.

Yasmine Bogaert
June 2021

Abbreviations

AI	Artificial Intelligence
CV	Cross-Validation
DBSCAN	Density-Based Spatial Clustering of Applications
DDA	Dynamic Difficulty Adjustment
ML	Machine Learning
SSE	Sum of Squared Errors
t-SNE	t-Distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
4-AFC	4-Alternative Forced Choice

Contents

Samenvatting iii

Summary v

Vulgarizing Summary xxi

Acknowledgement xxiv

Permission for Usage xxvi

Abbreviations xxviii

1 Introduction 1

2 Literature 3
- 2.1 Dynamic Difficulty Adjustment . . . 3
- 2.2 Player Enjoyment . . . 3
- 2.3 Player Modeling . . . 5
- 2.4 Measuring Player Experience . . . 6
- 2.5 Techniques for Player Modeling . . . 7

3 Methodology 12
- 3.1 Game Selection . . . 12
- 3.2 Game Mechanics of Chrome Dino . . . 13
- 3.3 Data Collection and Analysis . . . 15
- 3.4 Analysing Results . . . 16

4 Data Collection 18
- 4.1 Types of Data . . . 18
- 4.2 Website . . . 20
- 4.3 Data Collection . . . 24
- 4.4 Analysis . . . 24

5 Player Modeling 34
- 5.1 Global analysis . . . 34
- 5.2 Dimensionality Reduction . . . 41
- 5.3 Clustering . . . 44
- 5.4 Classification . . . 48
- 5.5 Challenges . . . 50
- 5.6 Conclusion . . . 51

6 Conclusions and Future Work 52

A Game Selection Analysis 56

B Global Analysis 59

C UMAP Sweep 68

D Clustering Algorithms 69

1 Introduction

The video game industry has been growing quickly over the past years, and is expected to grow even more in the future. Games begin to take place in more and more parts of our lives. Where they used to only be present in the entertainment industry, games have widely expanded to other markets. One significant trend is the concept of gamification, in which concepts from games (such as rewards, rankings, point systems) are used in non-game environments to enhance the user's experience. The use of serious games has also grown, particularly in such sectors as education, defense, aeronautics, science and health. There is a growing demand for learning through video games and little by little, the market is responding to that demand. [16, 29]

One might ask what keeps players entertained and come back to games. Even with concepts such as gamification, creating a game that is able to create enjoyment for many people and is able to retain that, is not an easy task. Players often get bored when a game gets too easy, or frustrated when a game gets too hard. The key factor is balancing the difficulty of a game so that it matches the skill of the player. Gamification and other similar concepts are based on general aspects of psychology that only cover global trends. The gaming audience is wide spread, and so is the individual skill level of players, ranging from new players to players with a lot of experience. Adapting the difficulty to match the skill level of the player, requires a different approach, one that focuses on individual aspects, to improve personalized player experiences.

Nowadays, adapting difficulty often only comes in the form of a static predefined set of difficulties, from which the user has to make a choice themselves. In the most basic approach, players can choose between an easy, medium, and hard mode of game. It does not come as a surprise that these often fall short for most players as these preset difficulties do not match the widespread individual skill and experience of different players. In addition, a player learns during the game, and his/her skills and expectations of the difficulty may change over time. Every player has an individual learning curve and following a predefined learning curve can either be too easy for a player, when the player learns at a faster rate, or too difficult, when the player needs more time to process and get on the same track. This is where dynamic difficulty adjustment comes into play. With DDA, the game will dynamically adapt to fit the learning curve to the player's skill level. It does this by building an underlying model of the player to gain insight into the player's enjoyment, and then makes an informed decision on how to adapt the difficulty to improve upon.

The most important step in constructing a DDA system, is creating the player model. Player models are based on objective game metrics, combined with labels about the individual subjective player's experience. A player model will enable us to predict player behavior in new situations that may have never been observed before. This model is then combined with an algorithm to adapt the game content based on those observations. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy.

Artificial Intelligence (AI) is a popular topic at the moment. More researchers than ever work on AI in some form and games are often a popular application area for AI research. Video games have during the last decade increasingly become the domain of choice for testing and showcasing new algorithms. Games not only pose interesting and complex problems for AI to solve, games are also a rare domain where science (problem solving) meets art and interaction: these ingredients have made games a unique and favorite domain for the study of AI [39].

The purpose of this thesis is to investigate player modeling within the context of DDA for Chrome Dino, a web-based video game. Human player data will be collected to get insight in how player enjoyment relates to the gathered player metrics and a model that represents these will attempted to be set up. Extensive research is done on different topics, from what defines player enjoyment from a psychological view, to how to capture it in a statistical reliable way and process the data to construct suitable player models.

The contents of this thesis can be summarized as follows: In chapter two, a literature study is done discussing dynamic difficulty adjustment, player enjoyment, player modeling, how to measure player experience and techniques for player modeling. This is followed by the methodology chapter, where the different steps of the approach are laid out. Chapter four deals with the collection and analysis of player data. After this, different methods for player modeling are applied and the results of several experiments are discussed. A conclusion is given in chapter six, together with a view on future work.

2 Literature

2.1 Dynamic Difficulty Adjustment

Dynamic difficulty adjustment (DDA) is a technique used in games to adapt the game's difficulty parameters to the players skill level in order to ensure that the player does not get bored (when the game is too easy) or frustrated (when the game is too hard). Te essence of the DDA is to retain the interest of the user throughout the game while offering a satisfactory challenge level. DDA is done during execution, tracking the player's performance and adjusting the game to present proper challenges to the player to ensure a good match between a player's skill and the game's difficulty. Some elements (game parameters) of a game that might be changed via DDA include: number of enemies, number of obstacles, generated items or duration of gameplay experience.

The most important step in constructing a DDA system, is creating the player model. Player models are based on objective game metrics, combined with labels about the individual subjective player's experience. A player model enables to predict player behavior in new situations that may have never been observed before. This model is then combined with an algorithm to adapt the game content based on those observations. An adaptive game that is constructed on accurate player models, is able to make predictions about future behavior, adapt the game to those changes, and better direct players toward content they are expected to enjoy.

Different approaches are found in the literature to address dynamic game difficulty balancing. Commonly used techniques for DDA include reinforcement learning (RLF), artificial neural networks (ANN), genetic algorithms or dynamic scripting, in which different weights are assigned to behaviour rules in intelligent agents.

2.2 Player Enjoyment

There have been several psychological studies that try to identify what is 'fun' in a game and what engages people to play video games. These studies include Malone's principles of intrinsic qualitative factors for engaging gameplay [31], namely challenge, curiosity and fantasy as well as the concept of flow theory by Csikszentmihalyi [14].

The concept of flow defines player states in two dimensions: skill and challenge. It relates the difficulty of a task to the viewers perception of its challenge. The flow channel is a state where a balance between skill and challenge is achieved, often described colloquially as being in "the Zone". When doing an activity that is neither boring nor frustrating, the person becomes immersed, is able to perform longer and keep focused on the task. This is a phenomena that is present in all fields, not only in the scope of video games.

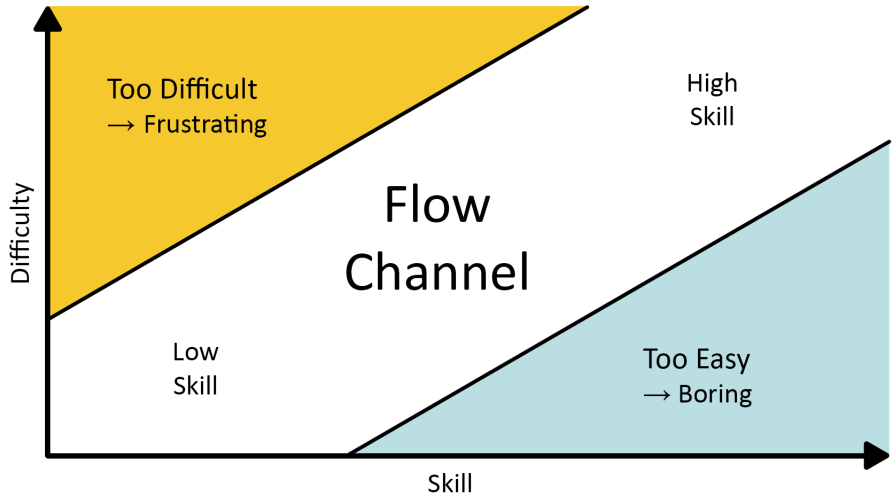


Figure 2.1: Flow channel.

When the difficulty of the game is higher than the player's skill, the game becomes frustrating, pushing the player into a state of anxiety. On the other hand, when the player's skill is higher than the difficulty, the game is too easy and pushes the player into a state of boredom. The goal is to keep the player in a state of flow and away from states where the game is far too challenging, or way too easy. One should only adapt the difficulty when a player can be kept in a state of flow. This is the precise task of the DDA system. DDA will enhance difficulty gradually, when it evaluates that the player has had sufficient time to learn and improve so that their skill can be matched with new challenges. [23, 41, 25]

To reach the initial state of flow, proper conditions have to be met. First of all, the player must perform a challenging activity that compels them to improve their skills. Second, the activity should have a clear and easy to achieve goal (levels, missions, high scores, etc.) with immediate feedback on progression. Lastly, the final result of the activity the player is performing should be uncertain, but at the same time, the player has to have a direct impact on its outcome. [15, 6]

In a different study, Malone [31] suggested that if the three factors of fantasy, challenge and curiosity need to be in balance at all times during gameplay. With challenge, we mean the uncertainty of achieving a goal (due to e.g., variable difficulty level, multiple level goals, hidden information or randomness). Curiosity refers to the feeling of uncertainty of what will happen next. Lastly, fantasy is the ability of the game to evoke situations that are actually not present. These three dimensions can be quantified and maximized together to achieve a system for optimal player enjoyment.

There have been many interesting studies on what defines 'fun' in video games and many of these models have a tendency to overlap and relate to each other in one way or another. In the following sections, the focus is put on general trends that are present in terms of fun, challenge and boredom.

2.3 Player Modeling

The primary goal of player modeling is to understand how a player's interaction with a game has an effect on the player's individual experience. We try to get an understanding of players' cognitive, affective and behavioral patterns [39].

There are two main factors that come into play: player behaviour and player experience. While player behavior and player experience are interwoven concepts, there is a subtle difference between them. Player behavior is about what a player does in a game whereas player experience refers to how a player feels during gameplay. It goes without saying that a player's feelings during a game (the player experience), correlate with what the player does during the game (the player behaviour). To construct an accurate player model, both the player behaviour and the player experience are measured. Both factors put together form the inputs for the player model.

When measuring player behaviour, gameplay input is taken into consideration, also called gameplay metrics. These concern any elements that can be derived from the direct interaction between the player and the game (e.g. number of jumps, time elapsed between milestones, buttons pressed). A major limitation of gameplay metrics is that they are always only indirectly observed. For example, a player who has little interaction with a game might either be really thoughtful and captivated, or just bored and busy doing something else. Gameplay metrics can only be used to approach the likelihood of the existence of certain player experiences. Therefore, one should never attempt to use pure player metrics to make estimates of player experiences and make the game respond accordingly to those estimates, as they could be inaccurate. It is best to also keep track of the player feedback, and adapt only when these indicate that the player experience was estimated incorrectly.

There are different ways of measuring player experience. One way is to look at the players emotional responses through physiological measurements. These include responses like player's facial expressions, muscle activation, brain waves, posture or speech. The relationship between psychology and these physiological effects has been studied extensively [39]. Monitoring such bodily alterations could help us in constructing the player's model. These signals are usually obtained through motion tracking, electrocardiography (ECG), galvanic skin response (GSR), respiration, electroencephalography, and electromyography (EMG). A downside is that measuring these factors can be intrusive to the player's experience. Measures are often also hard to implement and rather costly. However, physiology gives us an unbiased continuous measure of the player's state in real-time, without any interference of the player himself/herself.

Labeling player experience can also be done through manual annotation, either done by the player himself/herself or by a third-person. These assigned labels ideally need to be as close to the ground truth as possible. The accuracy of manual annotation is regularly questioned as there are numerous factors contributing to a deviation between a label and the actual underlying player experience [39]. The most common technique of labeling experiences is through a questionnaire. Manually annotating players and their gameplay is a challenge in its own right and arises many questions, such as: Will the labeling be dong by the player himself/herself or a third person? Will the labeling consist of discrete states or instead involve a continuous representation? Should the responses be queried in an absolute or relative fashion?. The details on how to measure player experiences are exlpained in the following section 2.4.

2.4 Measuring Player Experience

In this section, an overview is given of techniques to measure player experience through manual annotation. Topics include free vs forced responses, different types of formats to represent data, how to structure questionnaires and how to measure the player experience.

2.4.1 Free vs. Forced Responses

Player experiences can be evaluated either with free responses (e.g. thinking out loud) or forced responses (e.g. questionnaires). Free response contain more detailed information, but are often unstructured, chaotic and therefore become hard to analyze. On the other hand, forcing players to self-report their experiences using directed questions constrains them to specific items. In the remainder of this section, the focus is on forced responses as these are easier to analyze and are far more appropriate for data analysis and player modeling.

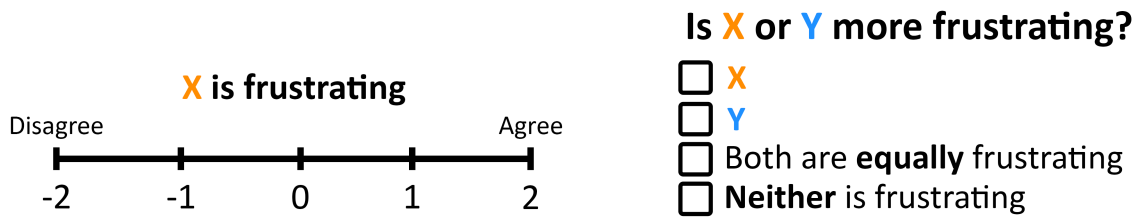
2.4.2 Forms of Data

There are three possible types of data for structuring player experiences: ratings, classes and ranks. Firstly, with ratings a player is asked to pick between a scalar value or a vector of values to indicate his opinion on given statement. Examples are "Did you find the game challenging?" or "Did you lose track of time?". Nowadays, ratings are unarguably the most widely used approach for assessing different aspects of a user's experience. The most popular rating-based questionnaires make use of questions in the format of a Likert scaled [12], which asks the user to indicate their level of agreement (or disagreement) with a given statement, within a range of predefined points. See figure 2.2a for an example.

Rating-based questioning is inherent to limitations, which are often overlooked [38, 40, 39]. Firstly, ratings are often analyzed by comparing their values across all participants. This neglects the existence of inter-personal differences as the meaning of each level on a rating scale may differ across participants. For example, two participants assessing the difficulty of a level may assess it as exactly the same difficulty, but then one rates it as "very easy to play" and the other as "extremely easy to play". Ratings also struggle with primacy and recency effects, in which information at the beginning and at the end are retained better than information in the middle. On top of that, ratings are commonly converted into ordinal values, while in most questionnaires, labels are represented as pictures or adjectives (such as "moderately", "fairly" and "extremely"). These labels are often converted to integer numbers, violating basic axioms of statistics. Since ratings suffer from these shortcomings, other options are taken into consideration.

The second data type is the class-based format. Classes ask players to select from a set of two or more options. Examples of questions are "Was that game level frustrating or not?" or "Which level was the most stressful?". Classes have less statistical limitations compared to ratings, but do not reach the same level of detail [39].

Finally, rank-based questions ask the player to make a preference ranking between two or more options, such as different gameplay sessions. In the case of pairwise preferences, the player is asked to compare two options and specifies which one is preferred. With more than two options, the participants are asked to provide a ranking of some or all the options. An example of a ranked-base questionnaire is 4-alternative forced choice (see figure 2.2b). Rankings do not suffer from the aforementioned limitations that are posed in rating or classes.



(a) Ratings on a 5-point Likert Scale

(b) Rankings through 4-alternative forced choice

Figure 2.2: Examples of ratings vs rankings.

2.4.3 Questionnaire

Questionnaires can contain elements of player experience (e.g. the Game Experience Questionnaire [9]), demographic data and/or personality traits. Similar research questionnaires often ask players to rank games of each game pair with respect to fun, challenge, boredom, frustration, excitement, anxiety and relaxation. The selection of these seven states is based on their relevance to computer game-playing and player experience [40]. Smaller, initial investigations only focus on three emotions: fun, challenge and frustration. [18]

2.4.4 Annotation

Annotation can happen either within particular time intervals or continuously. The process of annotation however, appears to require a higher amount of cognitive load [39]. Therefore, One should aim for short annotation at intervals where a change in the player's state occurs. Annotation can also happen pre, during or post gameplay. It should be taken into consideration that self-reports fade over time and the experience felt near the end of a session is often stronger, this effect is called the peak-end rule [30].

Evaluating player experience can either be done by the player himself/herself or by a third-person. The player's annotations should normally be closer to their actual experience (the ground truth), compared to a third person. Self-evaluation, however, may suffer from self-deception and memory limitations (memory-experience gap) [30].

2.5 Techniques for Player Modeling

Models of player behaviour and experience are often built using machine learning methods, typically supervised learning methods like support vector machines or neural networks [39]. In this thesis, the goal is to discern player models that can be used in a system for predictions based on these models in the scope of DDA.

Discovering player personas can be done in a first step with the help of a descriptive analysis, in which several different underlying game parameters and metrics are visualized and how they correlate to player experience. This may give insight into which variables contribute to which player behaviour and how they unfold in different player experiences. This analysis can shed light on aspects of player behaviour, which may provide answers to questions such as “when will this player stop playing?” or "will the player get stuck in that area of the level?". Discovering these associations is useful when frequent patterns or sequences of actions (or in-game events) can determine how a player behaves in a game.

The following methods are investigated: dimensionality reductions to compress the multi-dimensional space into one with less dimensions, clustering from AI methods that try to make out different clusters within the data space and classifications to make predictions about player experiences on new player data.

2.5.1 Dimensionality Reduction

High-dimensional data, meaning data that requires more than two or three dimensions to represent, can be difficult to interpret. One approach to simplification is to assume that the data of interest lies within lower-dimensional space. If the data of interest is of low enough dimension, the data can be visualized in the low-dimensional space. This is where dimensionality reduction comes into play. Dimensionality reduction is a powerful tool to visualize and understand large, high dimensional datasets. It transforms data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data. One of the most widely used techniques for visualization is t-SNE (t-distributed stochastic neighbor embedding). However, the performance of t-SNE suffers with large datasets and using it correctly can be challenging. An alternative to t-SNE is UMAP (Uniform Manifold Approximation and Projection). Visually, UMAP is very similar to t-SNE, but assumes that the data is uniformly distributed and locally connected. While both UMAP and t-SNE produce somewhat similar results, the increased speed, better preservation of global structure, and more understandable parameters make UMAP a more effective tool for visualizing high dimensional data [3].

UMAP starts by building a large connected weighted graph, with edge weights representing the likelihood that two points are connected. To determine if two points are in fact connected, UMAP extends a radius outwards from each point, connecting points when those radii overlap. Choosing this radius is critical - too small a choice will lead to small, isolated clusters, while too large a choice will connect everything together [3]. This radius is defined by the UMAP parameters `n_neighbors` and `min_dist` which are used to control the balance between global and local structure. These parameters are very intuitive to understand and shed light on how the UMAP algorithm works.

1. **n_neighbors.** The number of approximate nearest neighbors used to construct the initial high-dimensional graph. Low values will focus more on local structure by constraining the number of neighboring points considered, while high values will push UMAP towards representing the big-picture structure and lose fine details. (default: 15)
2. **min_dist.** The minimum distance between points in a low-dimensional space. This parameter controls how tightly UMAP clumps points together, with low values leading to more tightly packed embeddings and larger values causing UMAP to pack points together more loosely, focusing instead on the preservation of the broad topological structure. (default: 0.1)

2.5.2 Clustering

Clustering is the grouping a set of objects in such a way that objects in the same cluster are more similar to each other than to those in other clusters. In terms of player modeling, it is the classification of player behavior into a number of clusters depending of their behavioral attributes. Clustering is not only relevant for the personalization of the game, but also for understanding player behavior in association with the game design [2]. Common cluster techniques include k-means, DBScan, spectral clustering or Gaussian mixture models.

The k-means algorithm clusters data by trying to separate samples in n groups of equal variance. It chooses centroids that minimise the inertia metric, or within-cluster sum-of-squares criterion. This algorithm requires the number of clusters (k) to be specified beforehand. It scales well to large number of samples and has been used across a large range of application areas in many different fields [19].

In basic terms, the algorithm has three steps. The first step chooses the initial centroids, with the most basic method being to choose certain samples from the dataset. After initialization, k-means consists of looping between the two other steps. The first step assigns each sample to its nearest centroid, forming the clusters. The second step creates new centroids by taking the mean value of all of the samples assigned to each previous centroid and the redistributing all the samples over the new centroids. The difference between the old and the new centroids are computed and the algorithm repeats these last two steps until this value is less than a certain threshold. In other words, it repeats until the centroids do not move significantly [19].

Inertia makes the assumption that clusters are convex and isotropic, which is not always the case. It responds poorly to elongated clusters, or manifolds with irregular shapes (see figure 2.3).

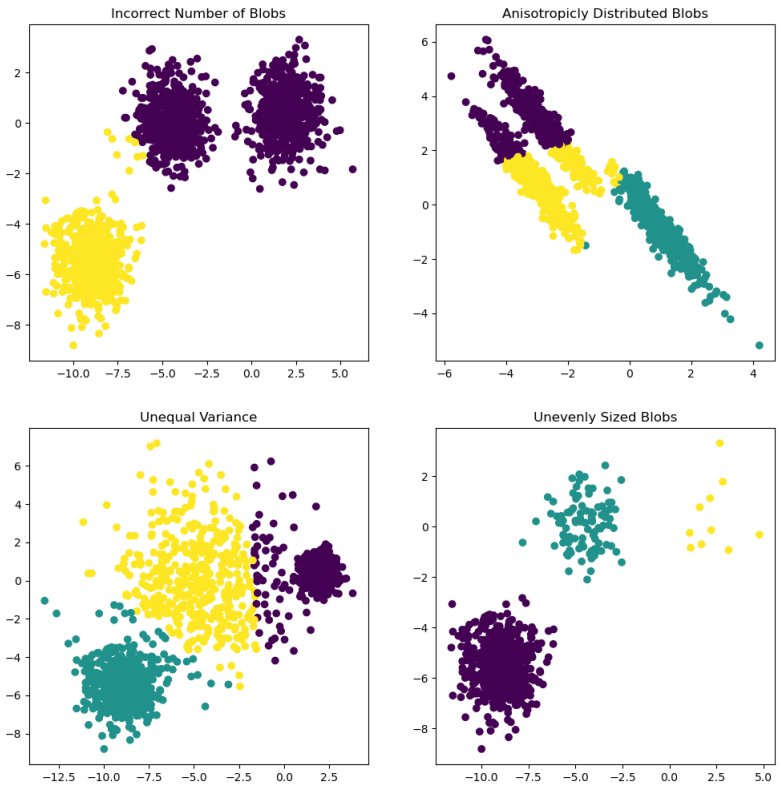


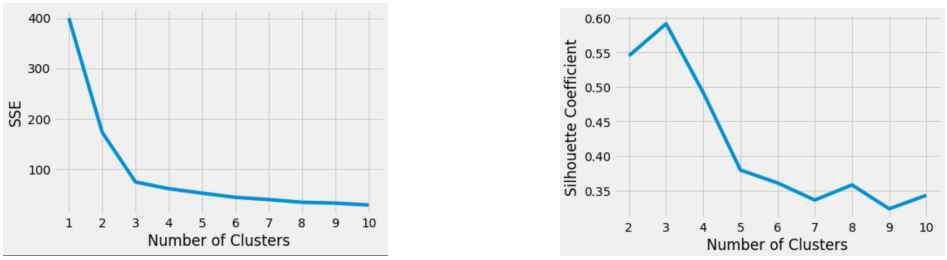
Figure 2.3: K-means clustering [19].

Hierarchical clustering determines cluster assignments by building a hierarchy. This is implemented by either a bottom-up or a top-down approach. Agglomerative clustering is the bottom-up approach. It merges the two points that are the most similar until all points have been merged into a single cluster. This produces a tree-based hierarchy of points called a dendrogram. Similar to k-means clustering, the number of clusters (k) is predetermined by the user. Clusters are assigned by cutting the dendrogram at a specified depth that results in k groups of smaller dendrograms. Different linkage criteria can be used that determines which distance to use between sets of observation. The algorithm will merge the pairs of clusters that minimize this criterion. Agglomerative clustering often reveals finer details about the relationships between data objects than other clustering methods and provides an interpretable dendrogram. Disadvantages are that they are often more computationally expensive and are sensitive to noise and outliers.

When the number of clusters has to be predefined by the user, finding the right value can be the most difficult, but also the most important task. Two methods that are commonly used to evaluate the appropriate number of clusters are the the elbow method and the silhouette coefficient.

When plotting sum of squared errors (SSE) as a function of the number of clusters, the SSE continues to decrease as k increases. As more centroids are added, the distance from each point to its closest centroid will decrease. There's a sweet spot where the SSE curve starts to bend known as the elbow point. This is where the optimal value for k is situated.

The silhouette coefficient is a similar measure of cluster cohesion and separation. It quantifies how well a data point fits into its assigned cluster based on two factors: how close the data point is to other points in the cluster, and, how far away the data point is from points in other clusters. Silhouette coefficient values range between -1 and 1. Larger numbers indicate that samples are closer to their clusters than they are to other clusters [4], which is the desired objective.



(a) Elbow Point Location

(b) Silhouette Coefficient

Figure 2.4: Methods for choosing the appropriate number of clusters.

Density-based clustering determines cluster assignments based on the density of data points in a region. Clusters are assigned where there are high densities of data points separated by low-density regions. Unlike the other clustering techniques, this approach doesn't require the user to specify the number of clusters. Instead, there is a distance-based parameter that acts as a tunable threshold. This threshold determines how close points must be to be considered a cluster member. An example of density-based clustering is DBSCAN (Density-Based Spatial Clustering of Applications). While density-base clustering excels at identifying clusters of nonspherical shapes and are resist to outliers, they aren't well suited for clustering in high-dimensional spaces and have trouble identifying clusters of varying densities.

2.5.3 Classification

Classification is a task that requires the use of machine learning algorithms that learn to predict class labels for new data, based on sample data. Samples can belong to two or more classes and the objective is to predict classes of unlabeled data based on already seen labels. In terms of player modeling, the samples consist of data from player behaviour, combined with a label of player experience (i.e. fun). The goal is the classification of new player behavior in classes of player experience, based on existing samples. Common classification techniques include decision trees, k-nearest neighbors, logistic regression or support vector machines. Ensemble methods take an additional step and combine the predictions of several base estimators in order to improve generalizability and robustness over a single estimator. Common techniques for ensemble methods include random forests or gradient boosting.

3 Methodology

To attempt to model player behaviour in the scope of DDA, an experiment is set up where data will be gathered through human data collection. A suitable game has to be selected that fits within the practical scope of this research. Annotation of player experiences will happen in between games to lower the cognitive load of subjects. Questioning will be done in a forced, ranked manner, as these methods have been proven to be statistically more suitable (see section 2.3). Results are analyzed with methods for discerning player personas and modeling player behaviour, which include dimensionality reduction, clustering and classification. In this section, the methodology for each of these steps is explained.

3.1 Game Selection

Before being able to gather information about player experience, a game has to be selected that is suitable for the context for this thesis. Due to the current circumstances of COVID-19, real-life experiments are out of the question and this search was limited to remote evaluations.

To select the game, the following parameters have been taken into consideration:

1. **Duration.** Shorter games are desirable. The player's experience will be evaluated in between games and to avoid loss of memory, maintain player retention and gather as much data entries as possible, games should be as short as possible.
2. **Number of game elements.** Too few elements are not enough to make enough variation in difficulties. On the other hand, too much elements can overload the complexity and make it too hard to discern a proper amount of difficulties. A game with a good balance in amount of elements, not too much, but also not too little, would be an ideal fit.
3. **Easy to implement.** To set up the application, an open source model of the game should be available that can be adjusted, or it is doable to start from scratch and remake the game within realizable time considering the time frame of this thesis.
4. **Expected popularity.** Taking into account the current circumstances, a big switch has been made to online activities. An online game is expected to gain interest. The game should also attract attention within the biggest target audience.

To find a game that fits these parameters, a collection of games was gathered that fit at least one or more of the aforementioned parameters. These games include: Chrome Dino, Pacman, Flappy Bird, Snake, Space Invaders, Pong, Super Mario Bors. and Tetris. See appendix A for a visual overview of all these games. To make a decision about which of these game fit the requirements the best, each game was assigned a score for each of the parameters from 0 to 5, and globally compared. These scores are of course annotated manually, and are subjective to personal interpretation, but effort has been put into making them as objective and close to the ground truth as possible.

From these scorings, there are three games that come out on top: Chrome Dino, Pacman and Flappy Bird. See figure 3.1 for a radar chart with the individual scores of these games. It is clear that Chrome Dino is the game which meets the requirements the most. See appendix A for an overview of the analysis of all games.

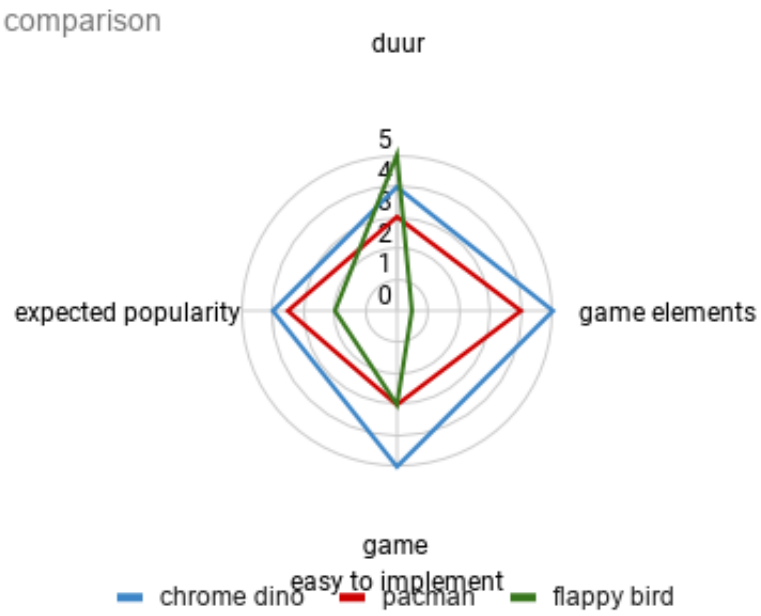


Figure 3.1: Comparison of Chrome Dino, Pacman and Flappy Bird.

3.2 Game Mechanics of Chrome Dino

Chrome Dino, also known as the Dinosaur Game, T-Rex Game, or Dino Runner, is a built-in browser game in the Google Chrome web browser. A user can encounter this game while browsing and a connection to the internet was not able to be established.

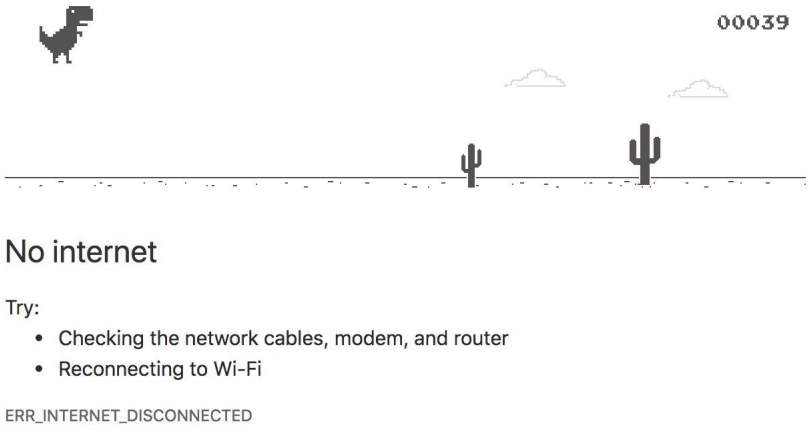


Figure 3.2: Chrome Dino game in the Google Chrome browser.

The game can be started by tapping the dinosaur (on mobile) or pressing the space or arrow-up keys (on desktop). The player controls the running dinosaur by jumping (tap, space, or arrow-up keys) or ducking (arrow-down key) to avoid obstacles, which include cacti and pterodactyls. When a player collides with any of these, the game is over. A score is accumulated for the distance traveled and the goal of the game is to stay alive as long as possible, hence getting a score as high as possible. A new game can be started by pressing the space or arrow-up keys again.

When the player reaches 700 points, the game switches from black graphics on a white background (representing daytime) to white graphics on a dark background (representing nighttime). Reaching the score of 900 (200 points further ahead) will switch the color scheme back to daytime and the next switches will occur at consecutive multiples of the milestones. For example, 1400 points for the next nighttime and 1600 for the corresponding switch to daytime.

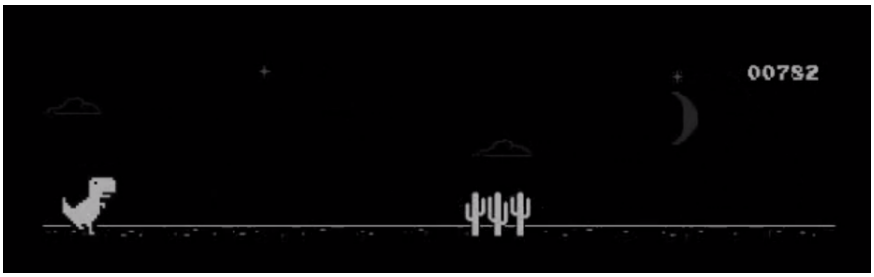


Figure 3.3: Chrome Dino: night mode.

At the beginning of the game, a period of time occurs when no obstacles are spawned yet, to let the player adapt to the pace of the game. The game starts at a moderate speed and slowly accelerates over the distance traveled.

Obstacles consist of: large cacti, small cacti and pterodactyl. Both large and small cacti can occur in different width sizes, up to a maximum of three. Each one making it harder to jump over. Pterodactyl can occur at three different heights. While a pterodactyl that flies low over the ground requires the player to jump over, a pterodactyl that flies in the middle can either be ducked under or jumped over. A pterodactyl that that is positioned at the highest point does not per se require a duck. The dinosaur fits right under it while running regularly. A player can still duck underneath it, but it is not necessary. See figure 3.4 for an overview.

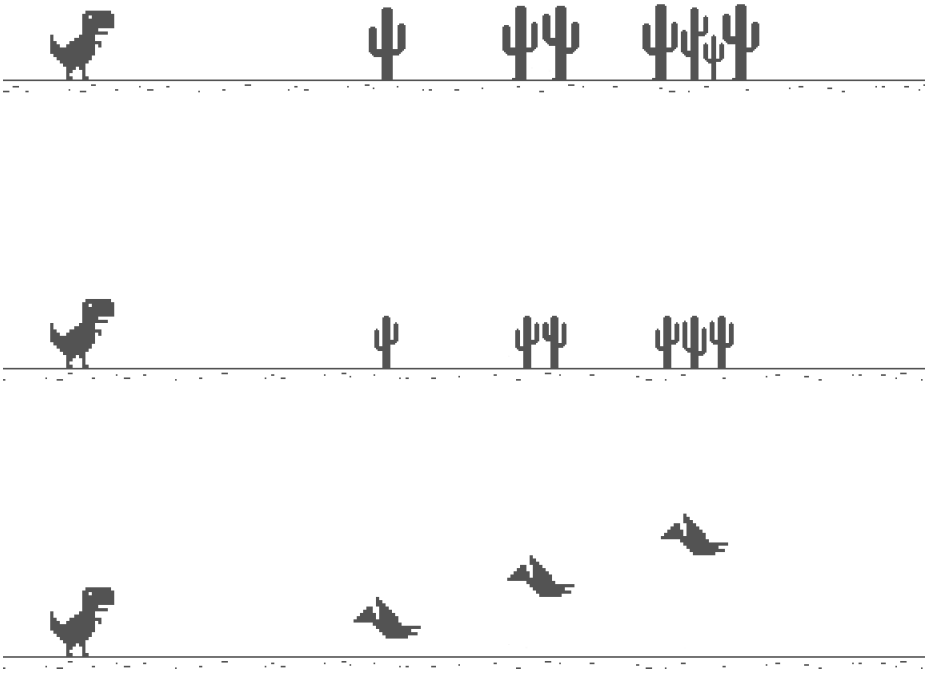


Figure 3.4: Obstacles in Chrome Dino.
Large cacti, small cacti, and pterodactyl, in their available sizes and heights.

Large and small cacti are positioned on the horizon and hence reach the dinosaur at the same speed that the horizon travels over the screen. Pterodactyl, however, are not attached to the horizon, and can get a speed offset (0.8), either positive or negative, which makes them move relatively faster or slower than the speed of the horizon. While the distance and appropriate time to make a jump for arriving obstacles usually can be statically predicted by the player since they are linked to the horizon, the player will have to make a more adaptive decision upon being approached by a pterodactyl.

3.3 Data Collection and Analysis

To evaluate player experience, attempt to construct a model for player behaviour and analyze it in the scope of DDA, player game data has to be collected. This is done with the goal of giving insight into how the underlying game metrics correlate to the game experience. Two types of data will be recorded. Game metrics, that log events during gameplay (such as the number of jumps, the achieved score), as well as the experience responses will be gathered through questionnaires in between games. Next to these recorded responses, the sampled game difficulty parameters will also be stored correspondingly.

For each consecutive pairs of two games A and B, subjects will report their preference for several emotional states: fun, frustrating and challenging. This is done with a 4-alternative forced choice (4-AFC) protocol. The 4-AFC format was selected based on related research which demonstrated that a rank-based approach is statistically most suitable (see section 2.4.2). Emotional states were limited to a set of three to keep the questionnaire as short as possible, to prevent breaking the tempo and retain players to play as much games as possible. An assumption is made that a correlation exists between fun, frustrating and challenging in such that the latter two contribute to the first, i.e. frustrating has a negative contribution to fun, while challenging has a positive contribution to fun. Similar research has also made use of the set of these three states in the past. (see section 2.4.3). Putting these parameters in the scope of flow theory (see section 2.2 and figure 2.1), a game that meets all three requirements of being fun, frustrating and challenging, is situated within the desired flow channel. A game that is too frustrating for a player, will be situated above the flow channel. It's difficulty needs to be lower so that the challenge of the game matches the player's skills. On the other hand, if the game is not fun and boring for a player, it will be situation below the flow channel, indicating that there is not enough challenge and the difficulty needs to be increased to make a good match for the player's skill.

Human data is collected through a custom variant of the Chrome Dino game, made available on a public website. Every new instance of the game will generate a new random version of the game, with different underlying game parameters that are randomly sampled from fixed ranges to make every game change in difficulty. The details of how this application was set up, will be explained in the data collection chapter (see chapter 4).

3.4 Analysing Results

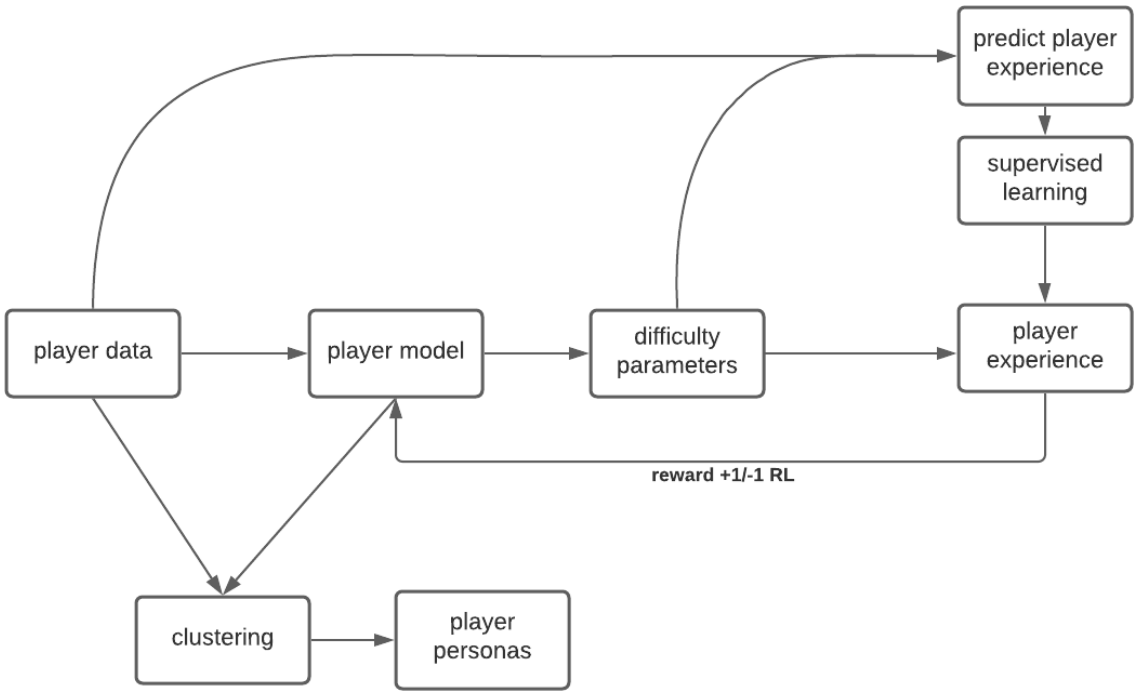


Figure 3.5: Analysis: structure.

Figure 3.5 shown the full scope of analysis. Player data will be collected to construct a player model. The goal is to find a player model that can predict if a certain set of difficulty parameters will be enjoyed by the player. This prediction can be done through supervised learning methods, like classification, that learn to make these predictions about player experience, based on player behaviour and difficulty parameters. On the other hand, clustering can be applied to discern different player personas that give insight into player behaviour and which factor contribute to an enjoyable experience.

To get a global overview of the gathered data, a descriptive analysis is be done to gain further insight into possible trends. Dimensionality reduction methods like UMAP or t-SNE can be applied to visualize the data and discover relevant parameters. To discern different different groups of player personas that describe games, and construct an associative player model, clustering techniques like k-means, agglomerative clustering and DBScan can be applied. Classification methods can be attempted to build a model that can predict labels of player enjoyment for new data, based on gathered samples. See chapter 5 for the full analysis.

4 Data Collection

In order to evaluate player experience in Chrome Dino, player data has to be collected. The goal of this collection is to find possible relationships between the evaluated player experience and collected records from gameplays and gain insight into which factors from game metrics contribute to player enjoyment.

4.1 Types of Data

Before any modeling can take place, data has to be collected. Three types of data are collected:

1. **Difficulty Parameters.** Underlying game parameters that give each new game a different difficulty. For example, the different obstacles generated, or the speed. See table 1 for a full overview of all the difficulty parameters.
2. **Game Metrics.** Upon occurrence of different events, game log are captured live during the game to encapsulate player behaviour. Examples are keyboard or mouse logs, the total number of jumps or the achieved score. A summary of all the different game metrics that are logged is given below in table 2.
3. **Player's Experience.** The player's experience is evaluated through a questionnaire for the emotional states of fun, frustrating and challenge.

Underlying game parameters should represent different game difficulties. The could be based on predetermined groups of difficulty settings (e.g. easy, medium, hard). When trying to construct those kind of difficulties for Chrome Dino, this turned out to be a harder task than expected. For example, an easy game could intuitively be thought of as a game with a relatively low start speed or only one type of obstacle. However, when testing out these settings, these games were sometimes very tedious and became harder to focus on. The decision was made not to make any presumptions about which parameters correlated to which difficulties. After all, putting more degrees of freedom towards these parameters allows to make a more free data analysis, without grouping them under 'easy' , 'medium' or 'hard. After all, this is the exact goal of this research, to study how these underlying parameters correlate to the player's experience. Since Chrome Dino is a rather simple game, the total amount of underlying parameters is fairly compact and can be captured almost completely. In the experiment, each game's difficulty is dynamically altered by randomly sampling the chosen underlying difficulty parameters to get a wide spread on presented game difficulties and corresponding experiences.

	Difficulty Parameter	Type	Description
1	`speed`	integer	start speed value between `5` and `10`
2	`acceleration`	float	acceleration added to `speed` in increments value between `0.002` and `0.01`
3	`max_speed`	integer	acceleration stops when `max_speed` is reached value between `speed` and `15`
4	`obstacle_types`	array	obstacle types that will be generated any combination of `cactus_large`, `cactus_small`, and `pterodactyl`
5	`obstacle_types_chances`	dictionary	chance of every obstacle type to occur chances assigned to `obstacle_types`, together 100%
6	`max_obstacle_length`	integer	the max width of small and big cacti `1`, `2`, or `3`
7	`pterodactyl_height`	array	pterodactyl can fly over ground, in the sky or in between any combination of `50`, `75`, and `100`
8	`clear_time`	integer	time before the first obstacle will occur value between `1000` and `5000`
9	`min_gap`	integer	minimum gap between obstacles value between `250` and `400`
10	`max_gap`	integer	maximum gap between obstacles value between `min_gap` and `400`
11	`night_mode_enabled`	boolean	is the switch between night- and daytime enabled `true` or `false`
12	`night_mode_distance`	integer	the distance after which will be switched to nighttime value between `100` and `1000` with steps of `100`

Table 1: Difficulty Parameters.

The game metrics should be meaningful in relation to the player's experience. However, the same reasoning as with game parameters as mentioned before can be applied. No assumptions are made about possible correlations, and the amount of game metrics is maximized. Because of the deterministic nature of the game, when capturing all possible game metrics, it should be possible to make a complete replay of individual gameplays. For testing purposes, this functionality tool was constructed and will also allow to look deeper into individual games that exhibit interesting features after data collection. It also validates the logging system as replayed and played games must have the same outcome.

	Game Metric	Type	Description
1	`actual_distance`	integer	achieved score at the end of a game
2	`collision_obstacle`	dictionary	obstacle upon which the dinosaur collided and caused death `large_cactus`, `small_cactus` or `pterodactyl`, with corresponding configuration parameters (`width` and `height`)
3	`inverted_gameover`	boolean	did the player die while in night mode
4	`nr_jumps`	integer	the total number of jumps done by the dinosaur
5	`key_logs`	array	all key and mouse events
6	`generated_obstacles`	array	all generated obstacles and their specifications

Table 2: Game Metrics.

When a game ends, the game parameters and game metrics are stored together as a single record in the gameplays database, together with the unique browser session id of the user.

After playing two (or more) consecutive games A and B, players are presented with a form structured in a 4-AFC way. Players can choose whether they experienced game A or B to be more or less fun, challenging, and frustrating. These factors were chosen based on other similar research (see section 3.3). Since the game has a shorter duration, factors were limited to 3, to prevent that the time spent on submitting the questionnaire dominates over the time playing the game. Breaking the tempo of the game is avoided to retain players as long as possible and maximise the amount of gathered data per subject.

Questions are structured as follows:

`Which game was more X?`

- `A`
- `B`
- `Both`
- `Neither`

Where A is the previous game, B is the current game and X is the emotional state in question: fun, challenging or frustrating. Upon submitting a form, the answers are stored as a single record in the questionnaire database, with references to the id's of the corresponding games A and B, combined with the unique browser session id of the user.

4.2 Website

This section covers how the website is set up, how progress is tracked during the data collection and which challenges had to be overcome in that process.

4.2.1 Game

A variant of the Chrome Dino is constructed based on a version of the original source code[5]. This version is extracted from the offline error-page that is built-in in the chromium browser, an open source project that is the base for the original Google Chrome browser.

The main components of the application exist of the Javascript frontend that runs the game, and a backend that runs a MongoDB database for storage of gameplay and questionnaire records. MongoDB is chosen for it's simplicity. The website is put available online at `chromedino.ml`.

Upon opening the website, the player is presented with a screen as in figure 4.1. Instructions on how to start the game are visible on the bottom of the playing field. Upon performing one of the start actions, the game will begin. The horizon will expand and the dinosaur will start running. After the initial clear time, the first obstacle will be generated.

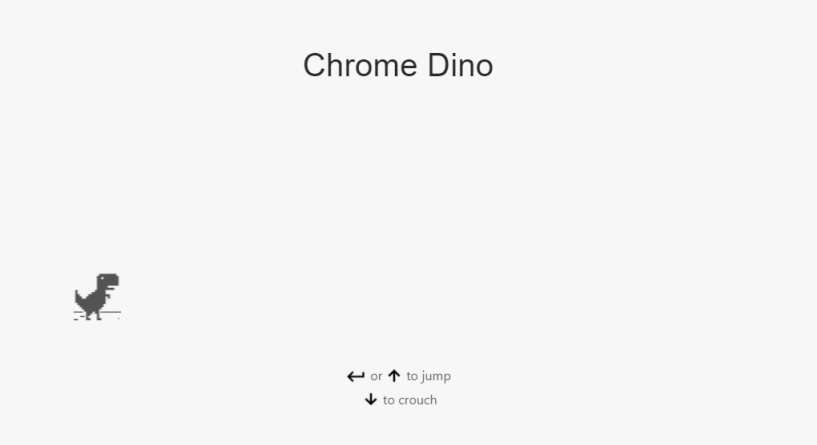


Figure 4.1: Chrome Dino startscreen.

Upon death, the difficulty parameters and game metrics (logs) are saved in the database, together with a snapshot of the gameover screen. This snapshot will be used in the questionnaire to display the played game. If the player only played one game so far, the game will be reset to the start screen and can begin a new game by performing one of the start actions. If this is not the player's first game (in this session), he will be presented with the questionnaire.

4.2.2 Questionnaire

After playing two or more games, the player will be presented with the questionnaire form that evaluates the player's experience about the last two played games.

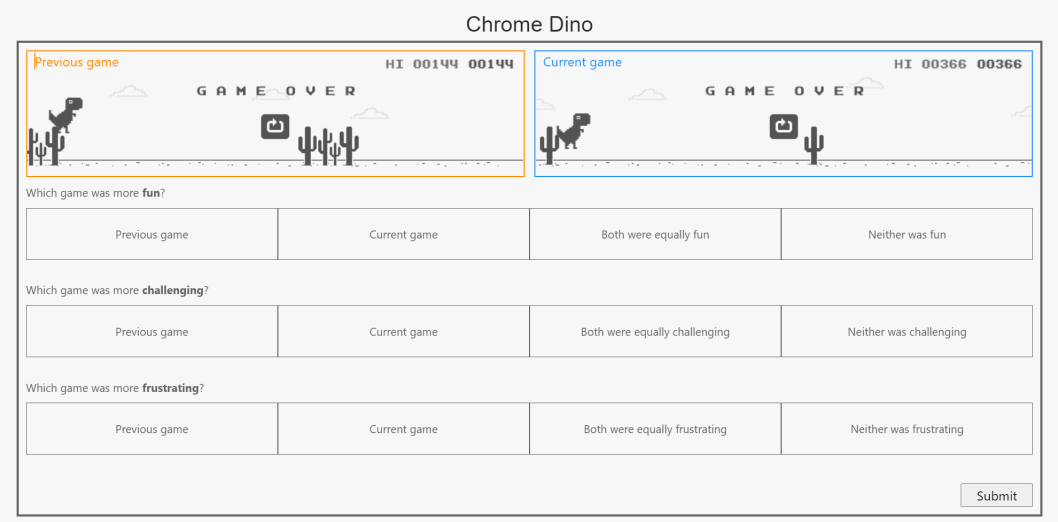


Figure 4.2: Chrome Dino form.

The form displays the two previously played games on top, with their associated snapshot at time of game over. The two games are labeled as 'previous game' and 'current game' and each get a different color, orange and blue. These colors will also be used to highlight responses on the questions, to make a clear association. The responses 'both were equally fun' and 'neither was fun' were also given the colors red and green correspondingly, indicating a negative and positive feeling.

The time it takes for a subject to complete the questionnaire is tracked and stored upon submission together with the answers to the questions in the database. This time is tracked to verify how long it takes players to form their answers to the questions. This factor will be taken into account when preprocessing the data, as the integrity of the answers might be flawed if a subject took too little or too much time. Also keeping in mind the retention of information, the decision was made to only show the form when the time between last 2 games is not more than 1 hour. When the player submits the form, he is redirected to a new game.

4.2.3 Dashboard

To keep track of incoming data, a dashboard has been composed to give a quick overview of the most important data. The dashboard contains:

1. **Record counters.** The amount of total accumulated games played and filled in questionnaires are displayed.
2. **Descriptive graphs.** Small descriptive graphs (e.g. score distribution) are made to get a first feeling of global statistics. These graphs give a head start for further analysis and will be examined further in chapter 4.4.
3. **Gameplays overview.** An tabular overview of the entries in the gameplay database, with their most relevant parameters. Each row in the table is clickable and connects to the replay functionality for that instance.

Gameplays

_id	dateTime	actualDistance	gameOverScreen	invertedGameOver	nr_jumps	collisionObstacle	newHighScore	nr obstacles	nr events	obstacletypes	parameters
6091a1911066e2827b2a3a04	2021-05-04T19:33:37.030Z	366		false	36	CACTUS_LARGE	true	36	74	CACTUS_LARGE	CLEAR_TIME: 1722 SPEED: 5 MAX+SPEED: 6 MIN_GAP: 251 MAX_GAP: 300 NIGHT_MODE_ENABLED: true NIGHT_MODE_DISTANCE: 500 OBSTACLE_TYPES: (CACTUS_LARGE) OBSTACLE_TYPES_SPEC: (1)
6091a1401066e2827b2a3a02	2021-05-04T19:32:15.364Z	144		false	45	CACTUS_LARGE	true	46	105	CACTUS_LARGE	CLEAR_TIME: 1079 SPEED: 8 MAX+SPEED: 8 MIN_GAP: 254 MAX_GAP: 270 NIGHT_MODE_ENABLED: false NIGHT_MODE_DISTANCE: 1000 OBSTACLE_TYPES: (CACTUS_LARGE) OBSTACLE_TYPES_SPEC: (1)
6091a1261066e2827b2a3a01	2021-05-04T19:31:49.870Z	434		true	32	CACTUS_SMALL	true	33	77	CACTUS_LARGE CACTUS_SMALL	CLEAR_TIME: 2809 SPEED: 8 MAX+SPEED: 11 MIN_GAP: 390 MAX_GAP: 397 NIGHT_MODE_ENABLED: true NIGHT_MODE_DISTANCE: 400 OBSTACLE_TYPES: (CACTUS_LARGE, CACTUS_SMALL) OBSTACLE_TYPES_SPEC: (0.8, 0.2)
605e0e5e1066e2827b2a3a00	2021-03-26T16:39:58.168Z	69		false	5	CACTUS_SMALL	true	2	39	CACTUS_SMALL	CLEAR_TIME: 2524 SPEED: 9 MAX+SPEED: 14 MIN_GAP: 272 MAX_GAP: 285 NIGHT_MODE_ENABLED: false NIGHT_MODE_DISTANCE: 400 OBSTACLE_TYPES: (CACTUS_SMALL) OBSTACLE_TYPES_SPEC: (1)
605770a61066e2827b2a39fe	2021-03-21T16:13:25.898Z	294		false	44	CACTUS_LARGE	true	77	121	CACTUS_LARGE PTERODACTYL	CLEAR_TIME: 3882 SPEED: 8 MAX+SPEED: 15 MIN_GAP: 315 MAX_GAP: 396 NIGHT_MODE_ENABLED: true NIGHT_MODE_DISTANCE: 400 OBSTACLE_TYPES: (CACTUS_LARGE, PTERODACTYL) OBSTACLE_TYPES_SPEC: (0.6, 0.4)

Figure 4.3: Chrome Dino dashboard: gameplays overview.

4.2.4 Challenges

When designing the form, it initially had the exact structure as mentioned in 4.1. The responses to the questions were: 'A', 'B', 'both were equally X' and 'neither was X'. With X being fun, frustrating or challenging. The gameover screens of the corresponding games were given the labels 'Game A' and 'Game B'. From an outsider's perspective, however, this can be confusing. It is hard to discern which game corresponds to A and which game corresponds to B, if it was the game hat had just been played or the game before that, and in which order. In an attempt to make the form more unambiguous, the labels were changed to 'previous game' and 'current game' as can be seen in figure 4.2. It was also taken into account that both games appeared in chronological order from left to right. To make it even more clear, colors were added. The previous game was given an orange color, and the current game was given a blue color. The responses 'both were equally fun' and 'neither was fun' were also given the colors red and green correspondingly, to signal a negative and positive feeling.

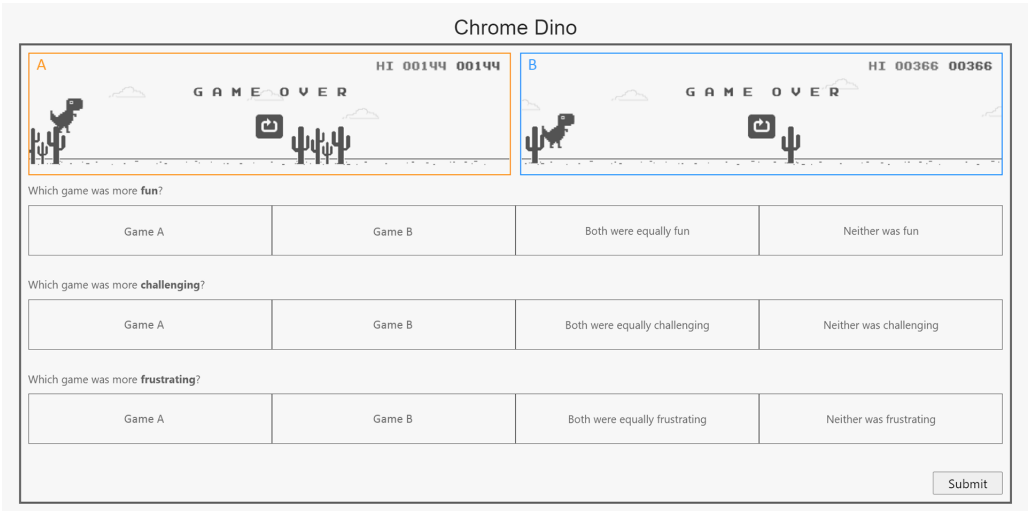


Figure 4.4: Chrome Dino form: old format.

The original Chrome Dino game does not contain instructions on how to begin the game, i.e. "`enter` or `up` to jump, `down` to crouch". However, a few subjects reported that it was unclear on how to precisely to play the game, presumable subjects who have never played the Chrome Dino game before. This being the case, action was taken to add instructions to the begin screen. The instructions also stay beneath the playing field when the game has started.

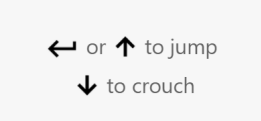


Figure 4.5: Chrome Dino instructions.

Due to the deterministic nature of the game, it was expected to easily be able to set up the replay functionality by just replaying the recorded jumps and ducks of the dinosaur, combined with giving the games the same generated difficulty parameters and generated obstacles. However, compared to general game mechanics, key and mouse events are not associated to time ticks, but are timestamped. Timestamps are unfortunately very system dependent and hard to get right. Due to this trait, a lot of precaution had to be taken in making the system act according to the replay events. An adjustable parameter was added for individual tweaking of system time dependencies, to compensate for incorrect gameplays as much as possible.

4.3 Data Collection

Data is collected over the Internet. Subjects are recruited via social media and networks of colleagues, friends and family.

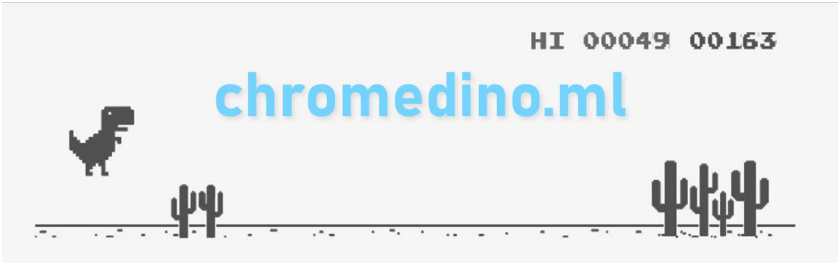


Figure 4.6: Image used to advertise on the internet.

4.4 Analysis

In total, 1074 gameplays were recorded, together with 524 answers to the questionnaire. In a first step, some preprocessing is done to the datasets, after which a descriptive analysis is made.

4.4.1 Preprocessing

Before starting to analyse the data, the datasets need some preprocessing. Preprocessing of the data is needed so that it only contains valuable information for the next stages of data processing. Unwanted data that is not clean and adds no extra knowledge is removed with the purpose of player modeling in mind.

When looking at the top scores (`actualDistance`) in relation to the total amount of jumps per game (`nr_jumps`) (see figure 4.7), there are a few gameplays present with exceptionally high scores and a very low amount of correlating total jumps. This is game behaviour that is only possible in one situation: a game where no obstacles appear that require the player to make jumps, but still achieve a high score. This corresponds to the configuration in which only pterodactyl obstacles appear at the highest flying position, so that the dinosaur fits right underneath when just walking and not making any actions. Although these games are interesting exceptions to regular games, they do not provide much useful information about player behaviour. In an endless runner game, the assumption can be made that achieved scores should generally be congruent to the amount of jumps made. There are, of course, exceptions to the rule, but not to these extremes.

To improve the integrity of the data with regards to player modeling, these games are filtered out. An additional indicator is added to look at `distance_per_jump`, which is the ratio of `actualDistance` over `nr_jumps`. See figure 4.7 for a full overview. The exceptional games from before correspond to very high values of `distance_per_jump` (> 10.000). The mean `distance_per_jump` of all the gameplays is 14.58 and 97% of all games have a `distance_per_jump` that is lower than 100. Hence, the choice was made to filter out gameplays that are above this threshold. Gameplays with a score higher than 5000 are also filtered out. The games that are left over are the ones with lower scores, and a lower `distance_per_jump`, and are in fact also the ones that contain the most information about general player behaviour, which is of interest in this analysis. After preprocessing, the top scores and corresponding total number of jumps are more suitable.

	actualDistance	nr_jumps	distance_per_jump
0	20005	2	10002.500000
1	10598	529	20.034026
2	10245	5	2049.000000
3	8182	782	10.462916
4	7954	582	13.666667

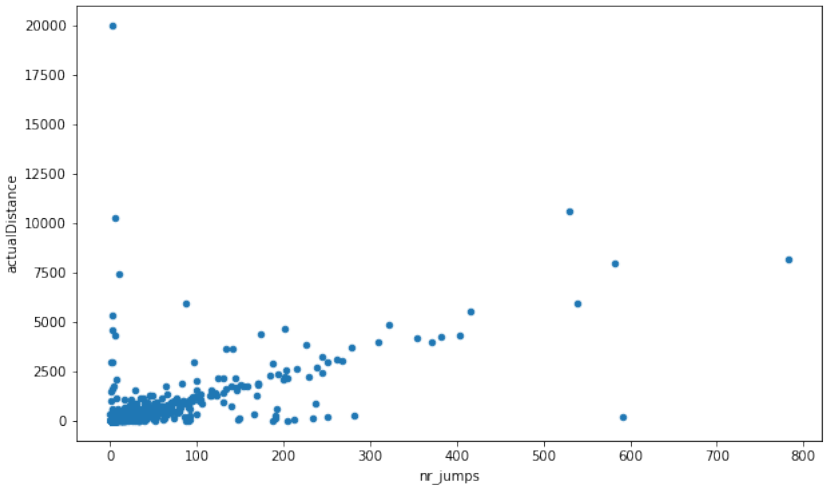
(a) Before Preprocessing

	actualDistance	nr_jumps	distance_per_jump
0	4850	321	15.109034
1	4667	201	23.218905
2	4379	173	25.312139
3	4359	404	10.789604
4	4254	381	11.165354

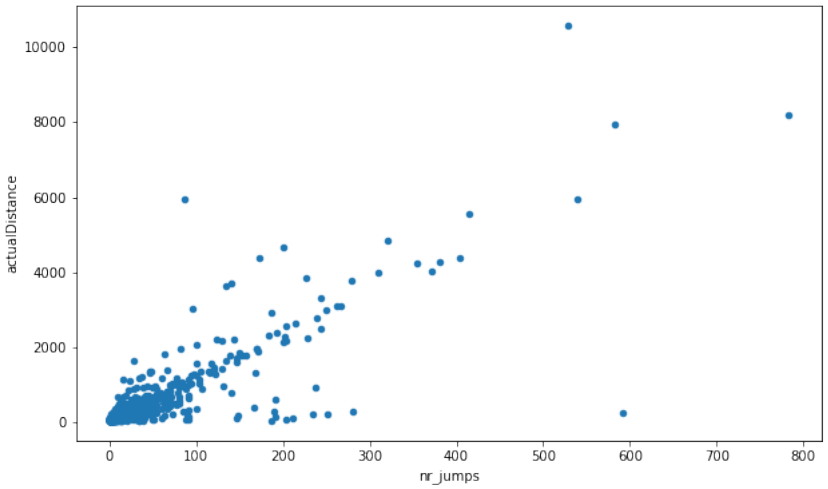
(b) After Preprocessing

Figure 4.7: Top 5 highest scores before and after preprocessing.

To verify this ratio and the thresholds, the correlation between `actualDistance` and `nr_jumps` is plotted. The ratio follows a clear linear correlation. However, before preprocessing, a lot of outliers are present. Almost concurrent to the y-axis, a line is detectable that contains all the games with exceptional high scores relative to their amount of jumps, that correspond to the game configuration mentioned before. After preprocessing, these are no longer present, as expected. By preprocessing the data in this way, extreme cases are dropped, while maintaining enough data that provides valuable information.



(a) Before preprocessing



(b) After preprocessing

Figure 4.8: Correlation between the total number of jumps per game (nr_jumps) and score (actualDistance) before and after preprocessing.

The time it takes for people to fill in the questionnaire is recorded. When looking at the longest times recorded, one can easily see that these are quite undesirable (see figure 4.9). The longest time is 22 minutes. Even though it is possible that some players may have a longer information retention span, generally most players will have already forgotten the main gist of the last 2 games they played by that time, even when those two games were short ones. Another exception could also be that the player forgot to submit his answers to the questionnaire and only did at a later time. However, the choice is made to not take these extremes into consideration.

To improve the integrity of the answers to the questionnaires, these exceptionary responses are filtered out. The mean submission time is 12 seconds and 95% of all the responses are submitted in a time below one minute. Hence, the choice was made to filter out submissions that took longer than one minute. After preprocessing, the time distribution follows the form of a skewed distribution, as expected, without extreme outliers (see figure 4.10).

time	minutes
1370129.625	22.835494
814790.505	13.579842
594588.980	9.909816
510120.665	8.502011
200455.380	3.340923

(a) Before Preprocessing

time	minutes
56565.195	0.942753
54661.430	0.911024
49774.060	0.829568
47862.000	0.797700
46666.000	0.777767

(b) After Preprocessing

Figure 4.9: Top 5 longest times to fill in the questionnaire before and after preprocessing.

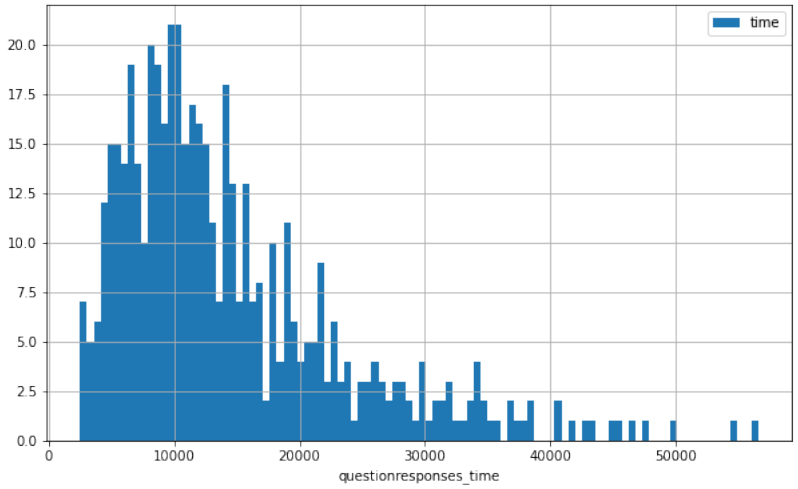


Figure 4.10: Distribution of the time it took to submit the questionnaire (after preprocessing).

In the rest of what follows, the data is looked at from a point after preprocessing is done to both datasets.

4.4.2 Descriptive Analysis

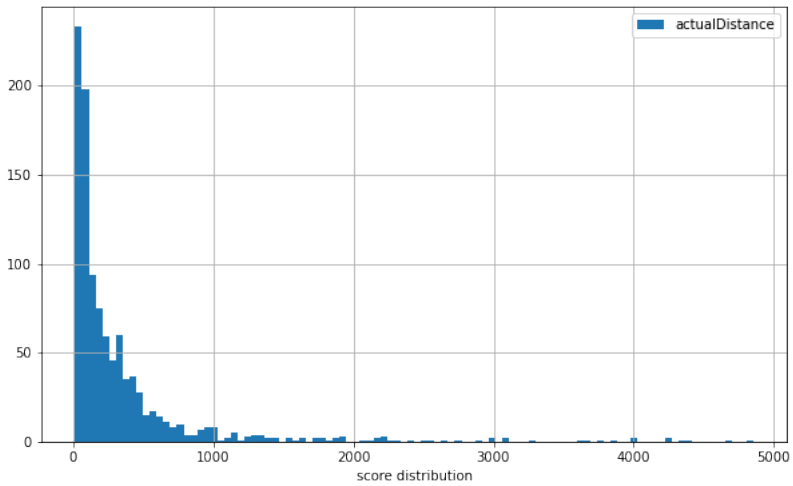


Figure 4.11: Score distribution.

The score distribution follows a right positive skew, meaning that most players achieve a very low score. This is as expected, as this effect can be assigned to the initial player learning curve. It can be assumed that a large portion of the players have no experience with the game yet and often collide with one of the first few obstacles, since they are not familiar yet with the pace of the game or do not know how the game controls work. The median score over all games is 160, which is also very low.

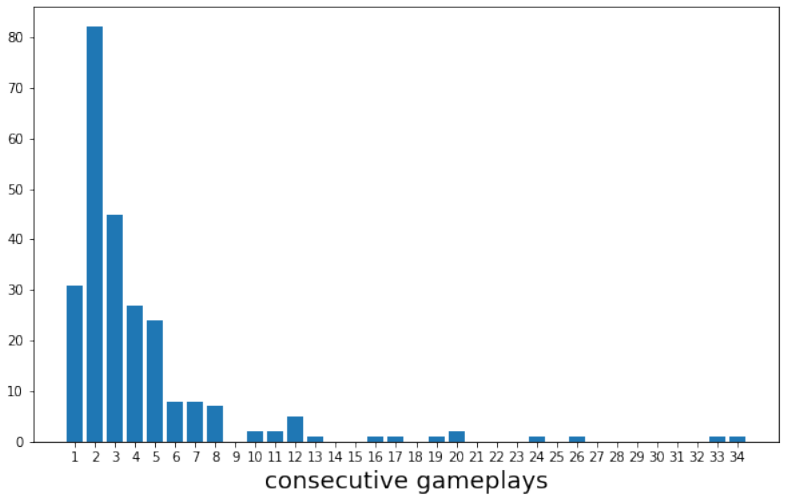


Figure 4.12: Number of consecutive games played per session: histogram.

Subjects are not obligated to play a certain amount of games. They can decide to stop playing at any time and leave the application. The histogram in figure 4.12 shows that most people play only two games in one session, and then leave. The average amount of games played is $4,15$. Consecutive gameplays should be more reliable, since they are less prone to noise. However, the biggest portion of the data is from low amounts of consecutive games. It remains to be seen what effect this has on the data analysis.

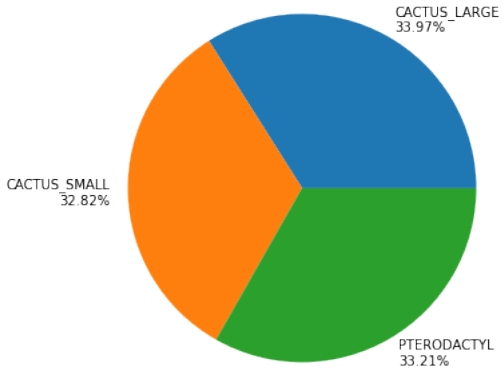


Figure 4.13: Collision obstacle pie-chart.

Different obstacles could be more difficult than others. If a certain type of obstacle causes more deaths, it could be an indication that it is harder to avoid collisions with. However, when looking at which obstacles cause the players to die, they are uniformly distributed. Players die upon collision with large cacti, small cacti and pterodactyl nearly equally. This is unexpected, since it could be reasoned that some obstacles are inherently more difficult than others. Pterodactyl travel slightly faster or slower than the regular speed, which requires the player to make a more dynamic action. This could become more difficult. A different distribution in number of total aggregated collision obstacles over all games, could have given an indication about the underlying difficulty. However, the data demonstrates that this is not a valid assumption.

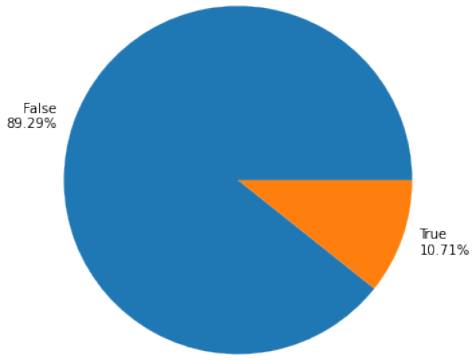


Figure 4.14: Inverted game over pie-chart (conditional on gameplays where night-mode is enabled).

Another interesting factor to look at could be whether people struggle more or less when night-mode is enabled. During data collection, a few players personally communicated that night-mode was one of the biggest issues for them. The percentage of players that die during night-mode when night-mode is enabled, is visualized in figure 3.3. Roughly half of the games have night-mode enabled. About 10% of players die during night-time, while the others all die during day-time. The median score over all games with night-mode is also 160, the exact same as for all total games. Taking into account that some players stated that they had a hard time with games in which night-mode was present, a larger amount of deaths could be expected. However, some players also indicated that they got dizzy when the concurrent switches between night-mode and day-mode occured frequently.

It is hard to make any concrete verdicts about findings that are solely deduced from factors upon player death. They only look at a single events and fall short of illustrating the overall complexity of what pays a role in player behaviour over the full length of the game. No conclusions can be made about correlations to game difficulty after the previous analysis of collision obstacles or game overs during night time. Nevertheless, the review of these statistics sheds light on global trends of player behaviour.

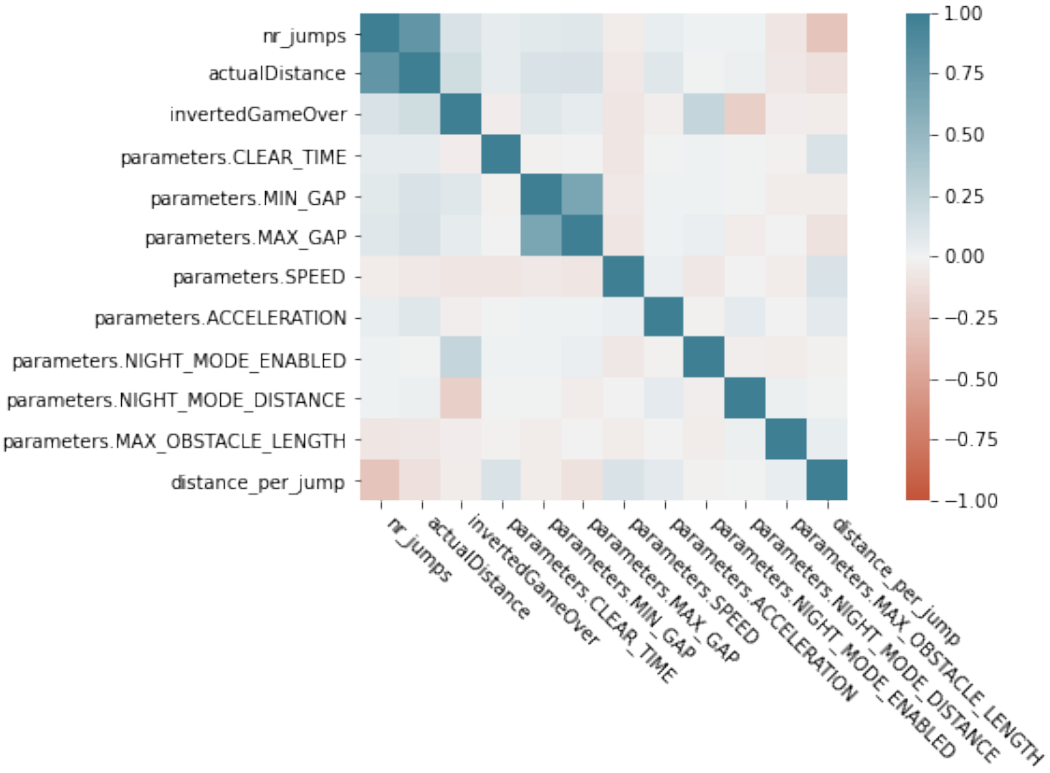


Figure 4.15: Heatmap of correlations between difficulty parameters and game metrics. Difficulty parameters have the prefix 'parameters', while game metrics have no prefix.

To get a global insight into how all the gameplay parameters and metrics that define player behaviour correlate to each other, a heatmap is constructed that visualizes the magnitude of these correlations by intensity in hue values (see figure 4.15). Blue indicates a positive correlation, while red indicated a negative correlation. As expected, there is a high correlation between the achieved score (`actualDistance`) and the number of jumps (`nr_jumps`). This trend was already established in 4.4.1 and the linear relationship was visualized in figure 4.8. Another self-evident link is present between `min_gap` and `max_gap`, as the `max_gap` range is restricted by `min_gap`. A positive correlation can also be perceived between `night_mode_enabled` and `invertedGameOver`, which is also self-explanatory since an inverted game over, or a game during night-time, can only be realized when the night-mode is enabled. A negative correlation is present between `night_mode_distance` and `invertedGameOver` that can be attributed to the fact that a shorter `night_mode_distance` results in night-mode to occur more often, and consequently make the chance that a player dies during night-mode higher.

In an attempt to get a feeling of an estimate of underlying game difficulty in relation to the score, of the mean score per configuration set of obstacles that are present is examined and visualized. The configuration consists of which obstacle types will be generated in a game, i.e. any combination of large cacti, small cacti and pterodactyl. See figure 4.16 for a visualization.

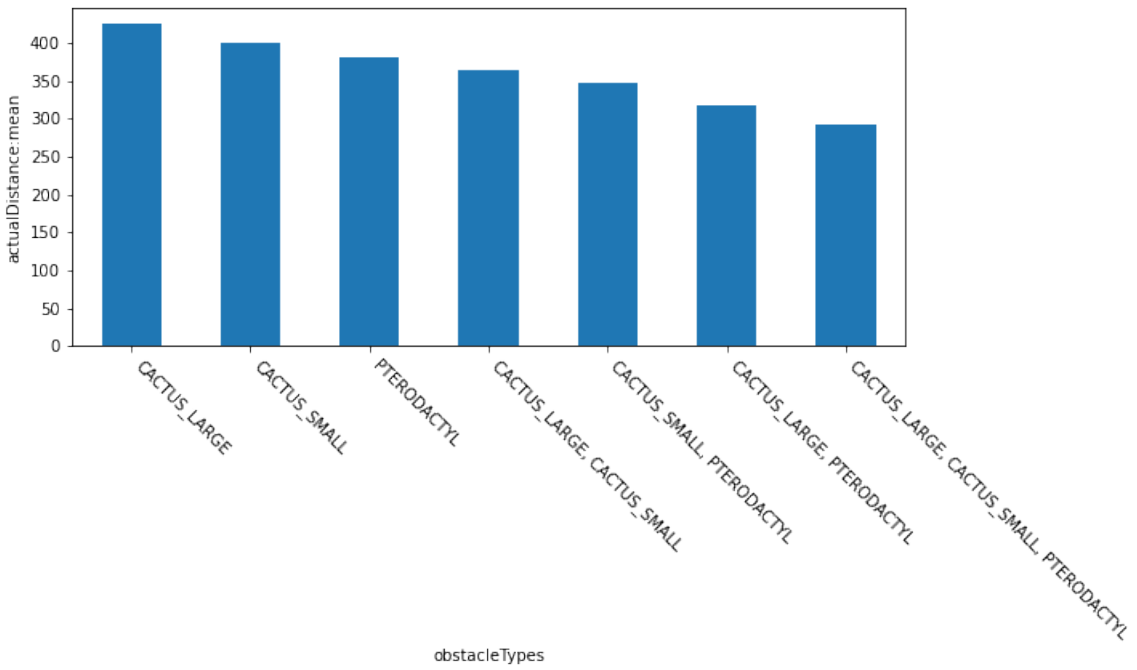


Figure 4.16: Barchart of the mean score per configuration of obstacle types.

The overall trend emerges that games with more variety in obstacles, in which more different obstacles are generated, generally have a lower mean score. Games in which all three obstacle types are present, have the lowest scores, while games in which only one type of obstacle is generated are the ones with the highest scores. Games with two different obstacle types lie in between. Another point that is noticeable, is the order in decreasing scores from large cacti firstly, to small cacti second and lastly pterodactyl.

To verify the assumption made in section 3.3, that a relationship exists between the factors fun, frustrating and challenging, inter correlations are visualized again by the use of a heatmap (see figure 4.17. As expected, frustrating correlates to fun in a negative way, while challenging contributes to fun in a positive way. There is not real relationship detectable between frustrating and challenging.

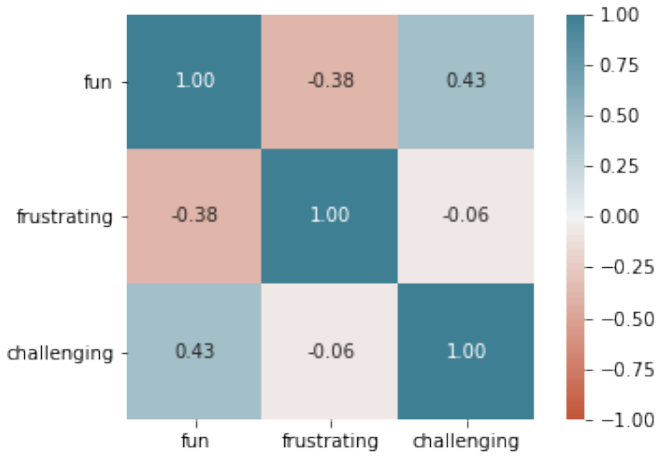


Figure 4.17: Correlations between fun, frustrating and challenging.

5 Player Modeling

The goal of this chapter on player modeling is to gain insight into how a player's interaction with a game has an effect on the player's experience by doing a global analysis together with applying different methods for player modeling, which include dimensionality reduction, clustering and classifictation.

5.1 Global analysis

Player experience is measured by subject's responses to the questionnaire, indicating their emotional state in terms of fun, frustration and challenge. These answers form the main point of interest when trying to gain insight into player experience. To get an initial oversight into which underlying difficulty variables contribute to each of these emotional states, one could make separate visualizations with the use of box-plots for the distribution of games under positive evaluations of each state. Meaning that for each evaluated game, the values of the metrics at the end of the game were used and linked with positive responses to fun, frustrating and challenging individually. However, since a lot of these distributions are very skewed, box-plots become hard to analyze, so an attempt was made to use violin plots as an alternative.

An example of such a violin plot for achieved scores can be seen in figure 5.1. In this plot, the distribution of different obtained scores is shown per positive evaluated emotional state.

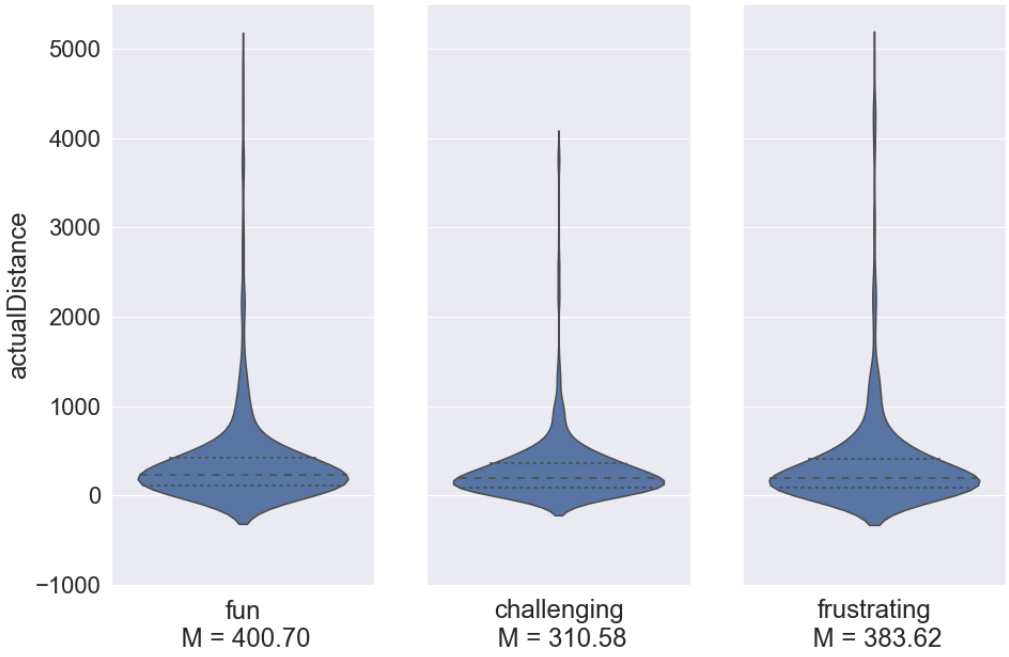


Figure 5.1: Distributions with violin plots for achieved scores (actualDistance) for different emotional states.

A game `A` is categorised as 'positively evaluated as X' (X = fun, frustrating, challenging) if it has been marked as being more X than another game B, or if it was one of either games evaluated as equally X. In other words, when the criteria under evaluation was X, and the question was as follows:

`Which game was more X?`

- `A`
- `B`
- `Both`
- `Neither`

, the responses '`A`' and '`Both`' contribute to game A being labeled as 'positively evaluated as X'. Games can be added more than once in the aggregation if they occur multiple times in the questionnaires, since this analysis focuses on player enjoyment and hence uses the evaluations on emotional states as the primary starting point.

Overall, it is observed that many of these distributions have as similar spread with not much deviations. This section addresses a few plots with the most interesting parameters and observations. See appendix B for a full overview of the plots for all game parameters and metrics. During the this study of game parameters, it was discovered that the `acceleration` parameter has not been correctly sampled. The only two values assigned were 0.002 and 1.002, instead of a continuous range, over 1030 and 12 games correspondingly. This is quite unfortunate, as the acceleration parameter could have been quite a relevant and insightful factor. Due to the extreme unbalanced distribution of this factor, the decision was made to remove it from further investigations.

In figure 5.1 it can be observed that fun games on average have a higher mean score, while games that are frustrating and challenging have lower scores. The distribution for challenging games is also slightly less spread out and indicates that games with a very low or very high score contribute less to a game being challenging. This is as expected, since games that are not challenging enough can be subdivided into two parts. On the one hand, games that are too easy are not challenging enough and easily facilitate getting a higher score, while on the other hand, games that are too difficult result in the player dying very early with a low score and are hence also not challenging enough.

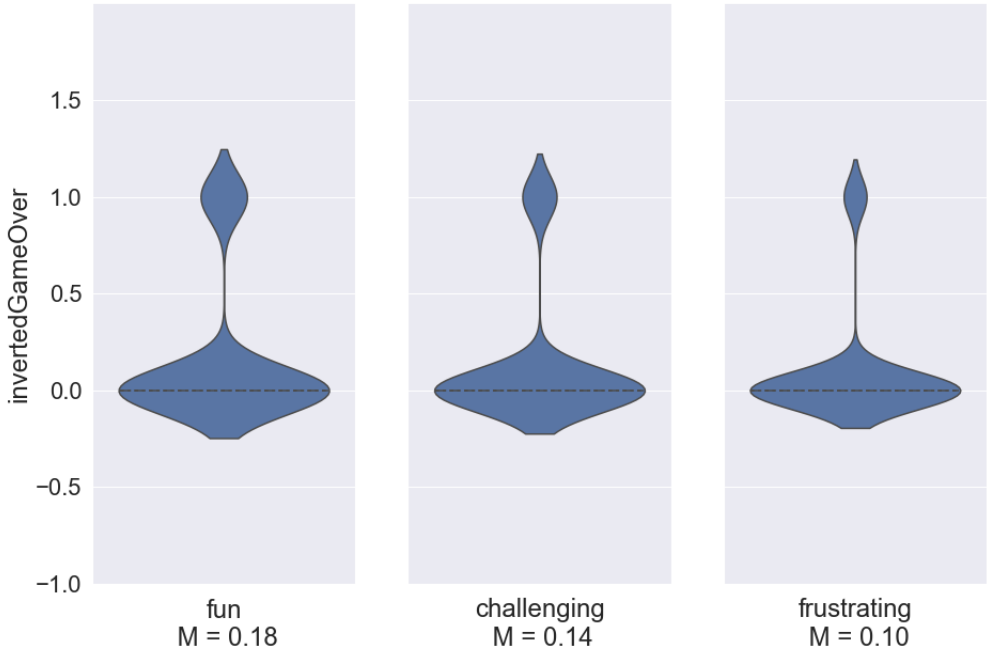


Figure 5.2: Distributions with violin plots for games that ended in night-mode (invertedGameOver) for different emotional states.

In figure 5.2 the distribution for the three separate positively evaluated emotional states has been visualized for `invertedGameOver`, the boolean that indicates whether or not the player died during night-mode. These distributions only take into account the subset of games in which night-mode was also enabled, and are hence only visualized for a part of the original dataset, compared to the other violin plots. Plots that do use all games for a factor such as `invertedGameOver`, would give a distorted view as games where night-mode is not enabled can never end in an `invertedGameOver` will only add skew to the distribution. It can be observed that games that were evaluated as fun, had more instances where `invertedGameOver` was reached. Games that are considered as challenging also have a relatively larger amount, while games that are considered to be frustrating have a low amount of `invertedGameOver`s. At first, this can seem counter-intuitive as one expects games where a player died during night-mode to be harder, more challenging and frustrating. However, the occasion where a player dies during night-mode happens rarely: only 5.0% over all games, and 10.7% over the subset of games where night-mode is enabled (discussed in section 4.4.2, figure 4.14). A game that does end in night-mode is exceptional, brings variation and could awake different emotions within subjects. In conjunction with this, the night-mode is also displayed with a different color in the questionnaire, which can attract more attention to the human eye, and could cause players to make different decisions solely based on the aspect of a different visualization. It is possible that this phenomenon is what leads player's to evaluate games with a `invertedGameOver` to be more fun.

Although these violin plots provide a quick overview of how variables are distributed, they still lack in portraying quantitative and relative information. Violin plots are still more readable than boxplots for these skewed distributions, but do not give information about proportional evaluations. It could be, for example, that for a certain value, a large amount is positively indicated under an emotional state in a violin plot, but it does not take into account how many total games were generated for that value. Hence, the decision was made to transform these violin plots into bar charts, and in a next step, normalize them so that relative relations can be analyzed. Bar charts also visualize more detailed information per bin, while a violin plot only sheds light onto global distribution shapes.

An example of such a bar chart can be seen in figure 5.3. This figure shows the distribution for the three separate emotional states for `invertedGameOver`, similar to figure 5.2. The same observations can be made. There are more instances of games with an `invertedGameOver` that were evaluated as fun, followed by challenging and lastly frustrating. Additionally to the three plots for each emotional state, a fourth plot has been added that counts the total generated games for each value.

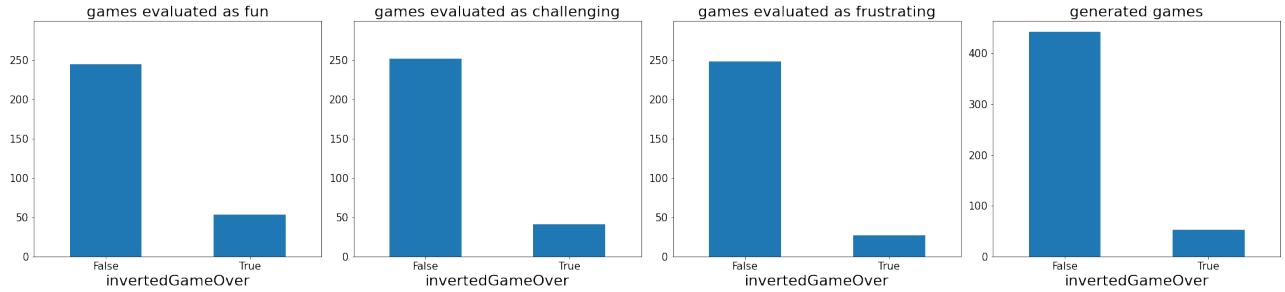


Figure 5.3: Distributions with bar charts for games that ended in night-mode (invertedGameOver) for different emotional states.

All previous visualizations allow us to make observations about absolute quantities. Nontheless, they do not take into account the full scope of all associated total generated amount of games. The case of `invertedGameOver` is a prime example. It would be more meaningful to look at this data in a relative fashion and compare the aggregation of positively evaluated games against their corresponding total generated quantities. Therefore, it proves useful to normalize the bar charts based on the total generated amount of games. See figure 5.4 for the normalized alternative of `invertedGameOver`. Normalizing was done by dividing every value by it's corresponding total generated quantity. These could be though of as portions or percentages, but are in fact actually not, since aggregations of evaluations are concerned with answers to the questionnaire, in which a game can occur more than once, while the total generated amounts are concerned with gameplays, that identify unique occurences of games.

In figure 5.4, the relationships from before can still be distinguished, only much stronger. Bar charts clearly give a better representation for this type of data. It is now clear that of all games with an `invertedGameOver`, a lot of them were perceived to be fun, while for games with no `invertedGameOver` a much smaller portion was perceived as fun. The same relationship is translated in games that were evaluated as challenging, meaning that games with a `invertedGameOver` are generally perceived to be more challenging. For games that were frustrating, there is not as big of a disparity to be discerned. On the contrary, games without `invertedGameOver` seem to contribute slightly more to being perceived as frustrating.

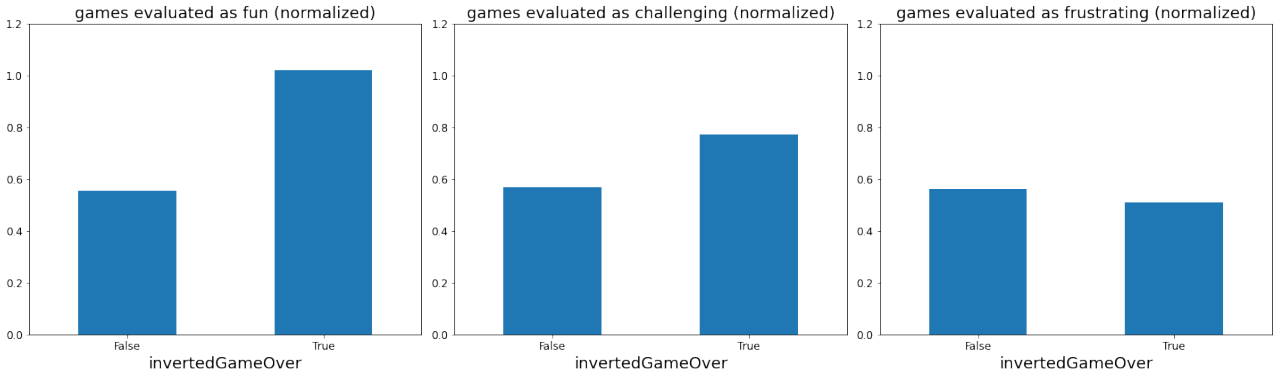
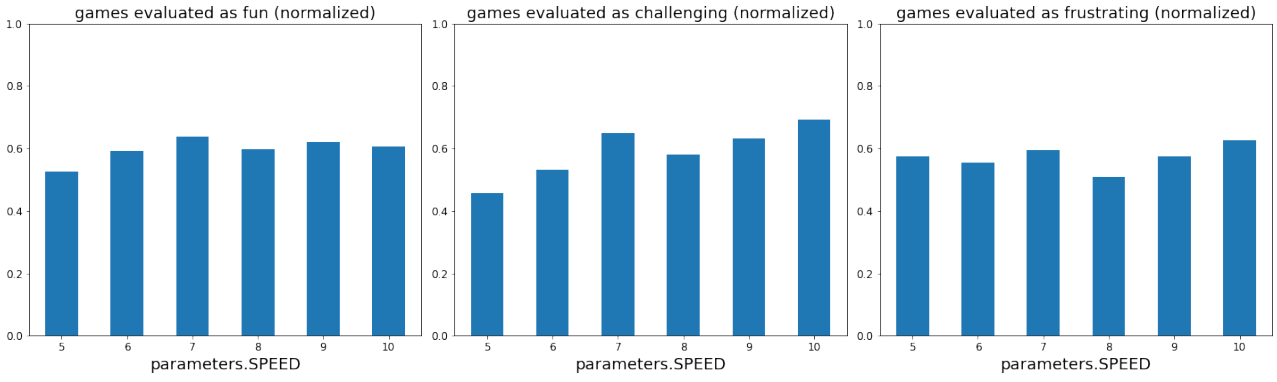


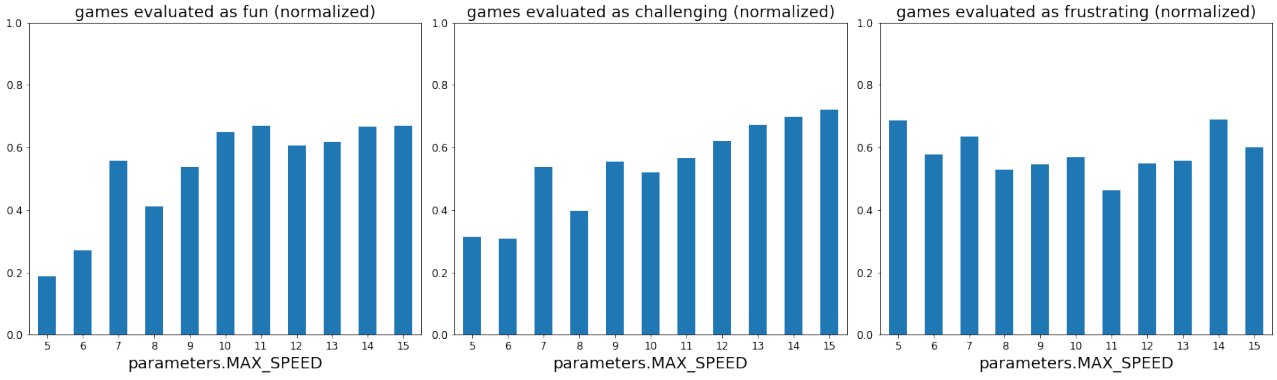
Figure 5.4: Normalized distributions with bar charts for games that ended in night-mode (invertedGameOver) for different emotional states.

An overview of bar charts for all underlying game parameters and metrics can be found in appendix B in both the normalized as well as the not-normalized manner.

Despite the fact that a lot of care was taken in the preprocessing to filter out unreliable outliers for player behaviour, for `actualDistance`, `nr_jumps` and `distance_per_jump`, the distributions are still very skewed and result in skewed relative proportions when it comes to normalization. It is hard to make any significant observations from these plots. Nonetheless, hey do provide a useful tool when when accompanied with plots of other parameters.



(a) Speed



(b) Maximum speed

Figure 5.4: Normalized distributions for speed and maximum speed for different emotional states.

Looking at speed, overall distributions are evenly spread, apart from the evaluations of challenging games, which are slightly skewed towards higher speeds. This is as expected, since games that start out with a low speed are intuitively less challenging. The corresponding violin plot for the speed parameter also verifies this observation (see figure 5.5).

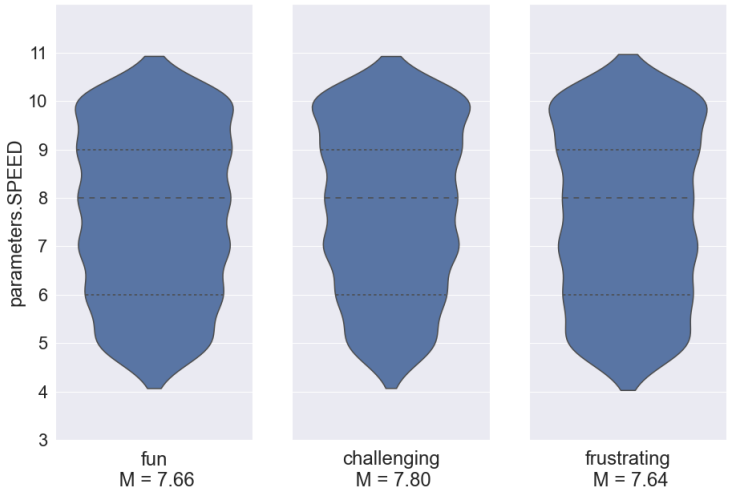


Figure 5.5: Distributions of the game starting speed with violin plots for different emotional states.

Observing these figures together with the distributions for the maximum speed, leads to stronger indications. Games that are evaluated as fun and challenging both have a distribution skewed towards higher speeds, indicating that players value higher speeds to be more fun and challenging. On the other hand, games that are evaluated as being frustrating have a much bigger portion of low speeds in their distribution. This is as expected, since games that have not much room for an increase in speed tend to lead to a stagnating pace in a game, becoming more boring and frustrating.

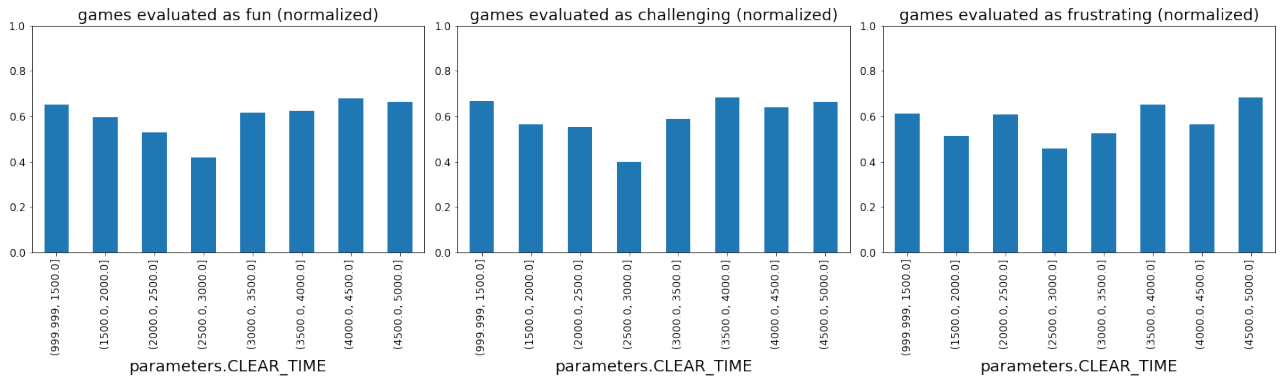
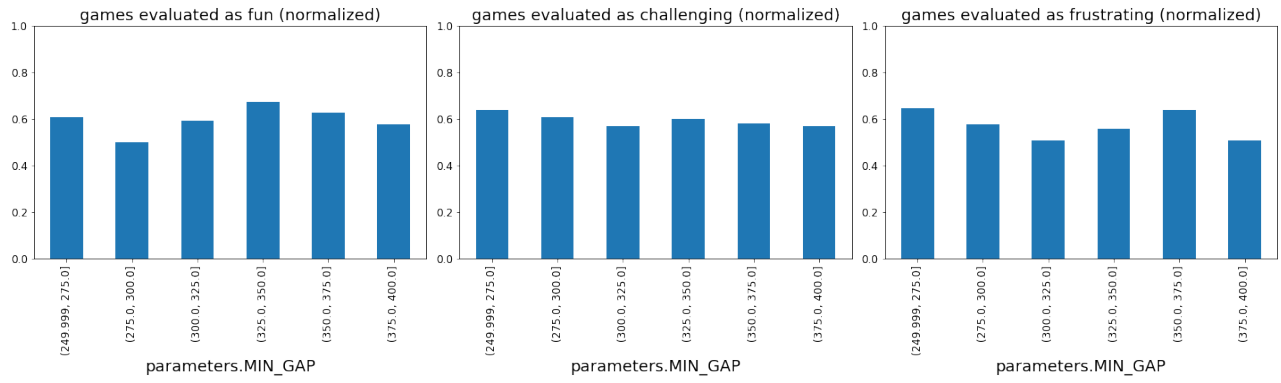
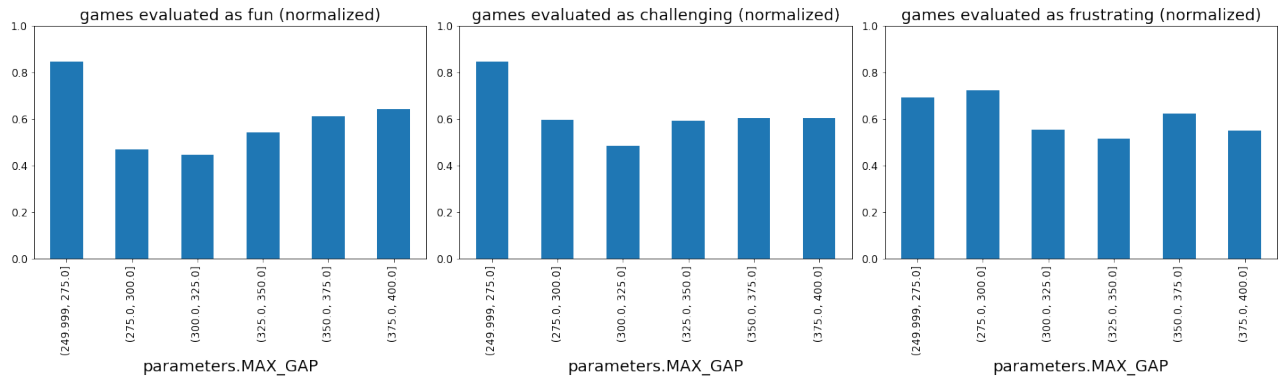


Figure 5.6: Normalized distributions for clear time (time before the first obstacle is generated) with bar charts for different emotional states.

At first sight, `clear_time` (figure 5.6) has a bi-modal distribution for every emotional state, meaning that it has two different peaks, one towards lower values and another one more centered in the high values. This could be explained by two groups of potential player behaviour. On the one hand, there are players with no experience in the Chrome Dino game yet, that need some time at the start of the game to prepare for the first obstacle. On the other hand, players who are familiar with the pace of the game, don't need this period and just want to start with the game itself as fast as possible. This unwanted pause could result in feelings of boredom or frustration. Speaking globally, this bi-modal pattern would be resolved in the long term when taking into account the learning curve. A longer clear time will become boring for most players if they get more familiar with the pace of the game after they played it a few times. However, in this setup, a lot of subjects only played the game a few times (see figure 4.12), amplifying the intensity of the lower value peak in the distribution. Putting these observations together with the distributions for `actualDistance`, `nr_jumps` and `distance_per_jump`, slight trends could become visible. It is possible that in this experiment, two main groups of player behaviour are present for the experienced and inexperienced player.



(a) Minimum gap



(b) Maximum gap

Figure 5.6: Normalized distributions for the minimum and maximum gap between obstacles for different emotional states.

Considering the minimum and maximum gap (see figure 5.6) clear peaks are visible for very low values of the maximum gap for all three emotional states. This can be explained by the fact that a low maximum gap broadly leads to obstacles being close together, which brings more variation to the tempo of the game, which could bring more challenge and frustration.

5.2 Dimensionality Reduction

The goal of dimensionality reduction is to understand and visualise the dataset of player behaviour (gameplays) in a lower-dimensional space. Afterwards, it is useful to compare this reduction with the dataset of player experiences (game evaluations) to find possible correlations. Therefore, only gameplays should be considered that have also been evaluated at least once. Both datasets have therefore been reduced to their inner joins for this section, reducing the recorded gameplays to 693 entries and answers to the questionnaire to 491.

5.2.1 UMAP

Making a dimensionality reduction with UMAP requires a good choice for it's internal parameters `n_neighbors` and `min_dist` to find a good balance between global and local structure. Instead of making guesses, a full sweep is done with `n_neighbors` in the range of 2 to 200 and `min_dist` with a range for 0 to 1. See appendix C for an overview of the full sweep. This sweep proves that the default parameters of `n_neighbors` = 15 and `min_dist` = 0.1 are adequate. See figure 5.7 for the resulting visualization. Three clear groups can be distinguished. The next section investigates these different groups further.

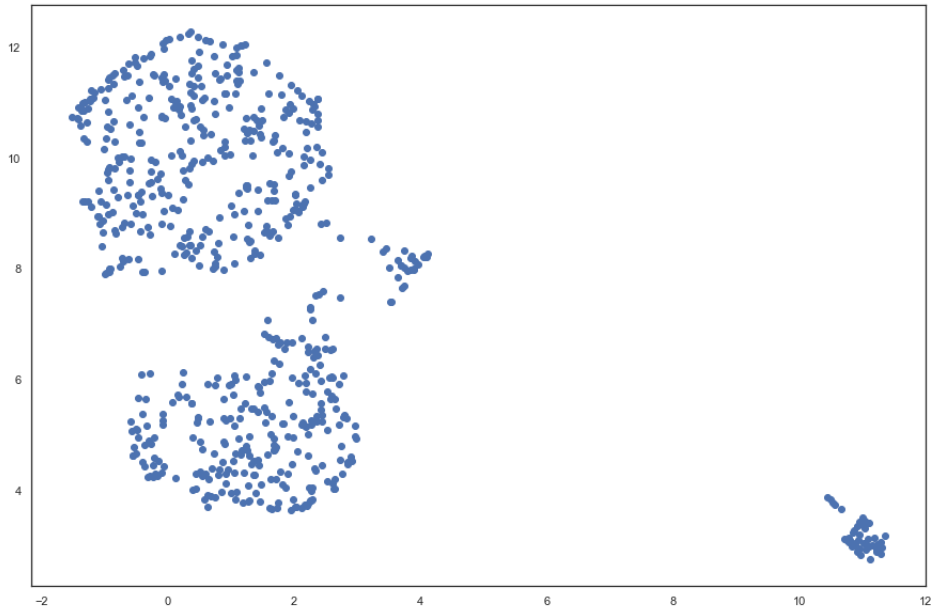


Figure 5.7: UMAP dimensionality reduction.

5.2.2 Mapping Underlying Game Parameters and Metrics

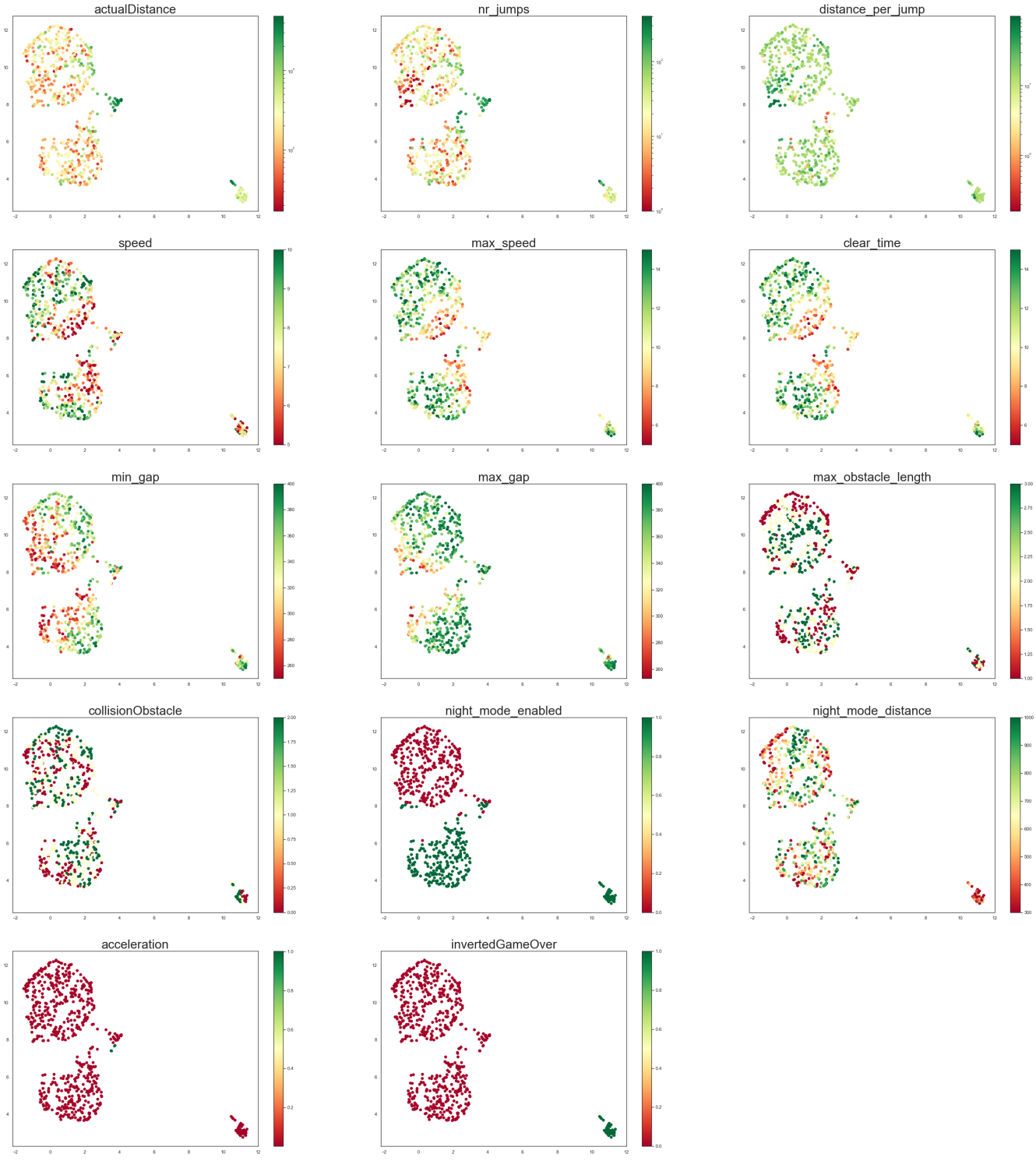


Figure 5.8: UMAP dimensionality reduction, with individual mappings for the underlying parameters.

UMAP was able to recognize three different groups. Two of these groups can be perceived as the main groups, with small overlap between them, while the third group is located at a larger isolated distance with no overlaps in any way. In figure 5.8, different underlying game parameters and metrics are mapped. Colors display low to high ranging values, indicated from green to red correspondingly. Some scales have been converted logarithmically to atone for skewed distributions.

The first point of interest is the isolated third group. One of the sub-figures shows that it's seclusion is a consequence of `invertedGameOver`. Within this group, there is not much structure to be detected, apart from two small protrusions, one for games with very high scores, and one for games with a very low minimum gap. The two main groups are separated by one dominant factor, `night_mode_enabled`. It goes to mention that these large discrepancies are both related to boolean factors in the dataset. Compared to all other variables, which have a more spread out range, it is a self-evident matter that UMAP detects these to be at a larger distance from eachtother.

Within the two largest groups, similar reciprocal trends are noticeable. Each group is subdivisible, focused around `min_gap`, `max_gap`, and `max_speed` or `clear_time`. It almost looks like the larger groups are subdividable into quadrants, but it is hard to deduce if they each represent a certain kind of player behaviour. The overlap between the groups can be linked to the games with very high scores, almost forming a separate group.

5.2.3 Mapping Associated Player Enjoyment

The next step is to analyse if there is any correlation betweem the previous observations about player behaviour and aspects of player experiences. For this, a label is needed for each game that indicates how the game was experienced. This label is built on (possibly multiple) ranked evaluations that were made for that game. For each game, a score is assigned for every emotional state. This score is calculated by aggregating all positive evaluations for that emotional state (see section 5.1) and taking the mean over them. In other words, a game will get a score of 1 if in every evaluation form that it was present, it has been evaluated for the emotional state X as 'this game was more X than the other game' or 'both games were equally X'. If it was never positively evaluated, it will get a score of 0. Since a single gameplay can only occur in a form up to a maximum of two times, it is possible that the game was one time positively evaluated, but another time negatively evaluated, and will then get a score of 0.5. These scores have been mapped on the UMAP clusters in figure 5.9. To get an idea of a global score, the three emotional states have been combined into a formula that calculates an approximate score for total player enjoyment. The formula is based on inter-correlations between the three emotional states (see figure 4.17) and is estimated as follows:

$$total = fun + 0.5 * challenging - 0.5 * frustrating \tag{1}$$

See figure 5.10 for a mapping with this formula for an approximate total score of player enjoyment.

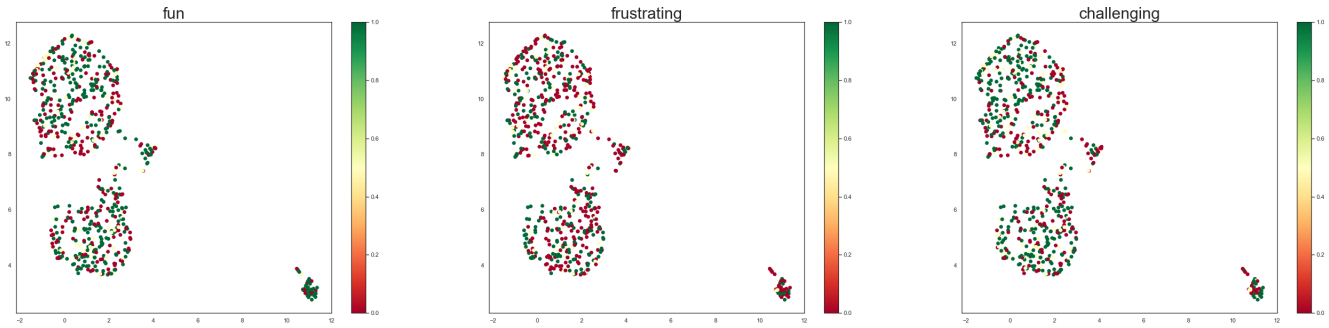


Figure 5.9: UMAP dimensionality reduction, with mapping of individual evaluations for all three emotional states.

Unfortunately, the outcomes are very spread out over the groups and show no real structure. If instead any structure was distinguishable, it would be possible to link this to the groups discerned in the previous section 5.2.2, but sadly, this is out of the question.

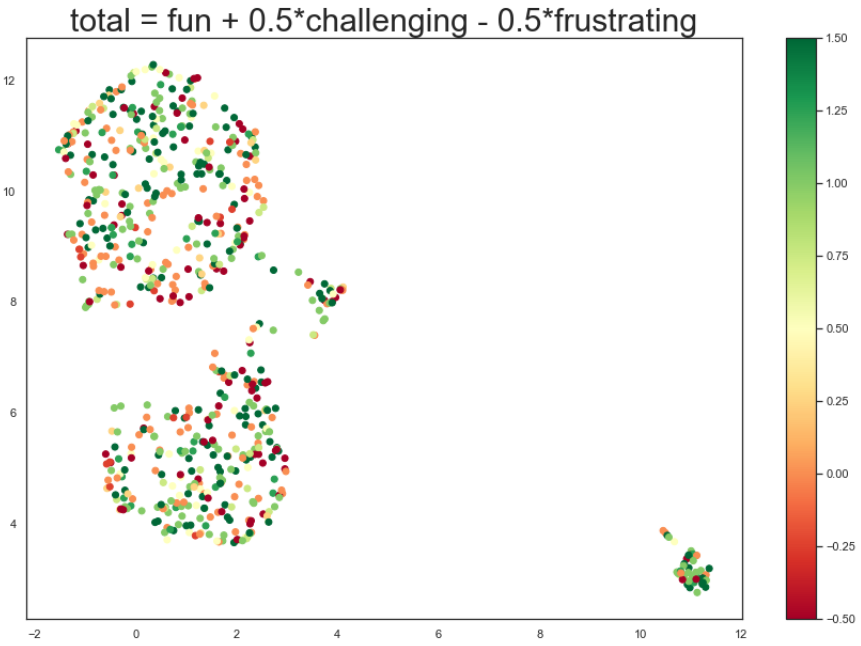


Figure 5.10: UMAP dimensionality reduction, with mapping of positive evaluation formula.

5.3 Clustering

To find different groups of player personas that describe games, player behaviour or player experience, clustering algorithms can be applied. The goal is to find groups of gameplays or player behaviour such that players that belong to the same group demonstrate the same characteristics. As seen in section 2.5.2, many algorithms for clustering are available, each with their own strengths and weaknesses.

The most common algorithm for clustering is k-means. The number of clusters (k) has to be predefined by the user. Finding the right value for k is often the most difficult but also the most important task. To validate the value for k, two methods are used: the elbow method and the silhouette coefficient. To verify the analysis and gain further insight into the algorithm, a full sweep over the hyper parameter k is done (see figure 5.11). It is not expected that up to 9 clusters could be discerned. These values are only used to illustrate the underlying algorithm.

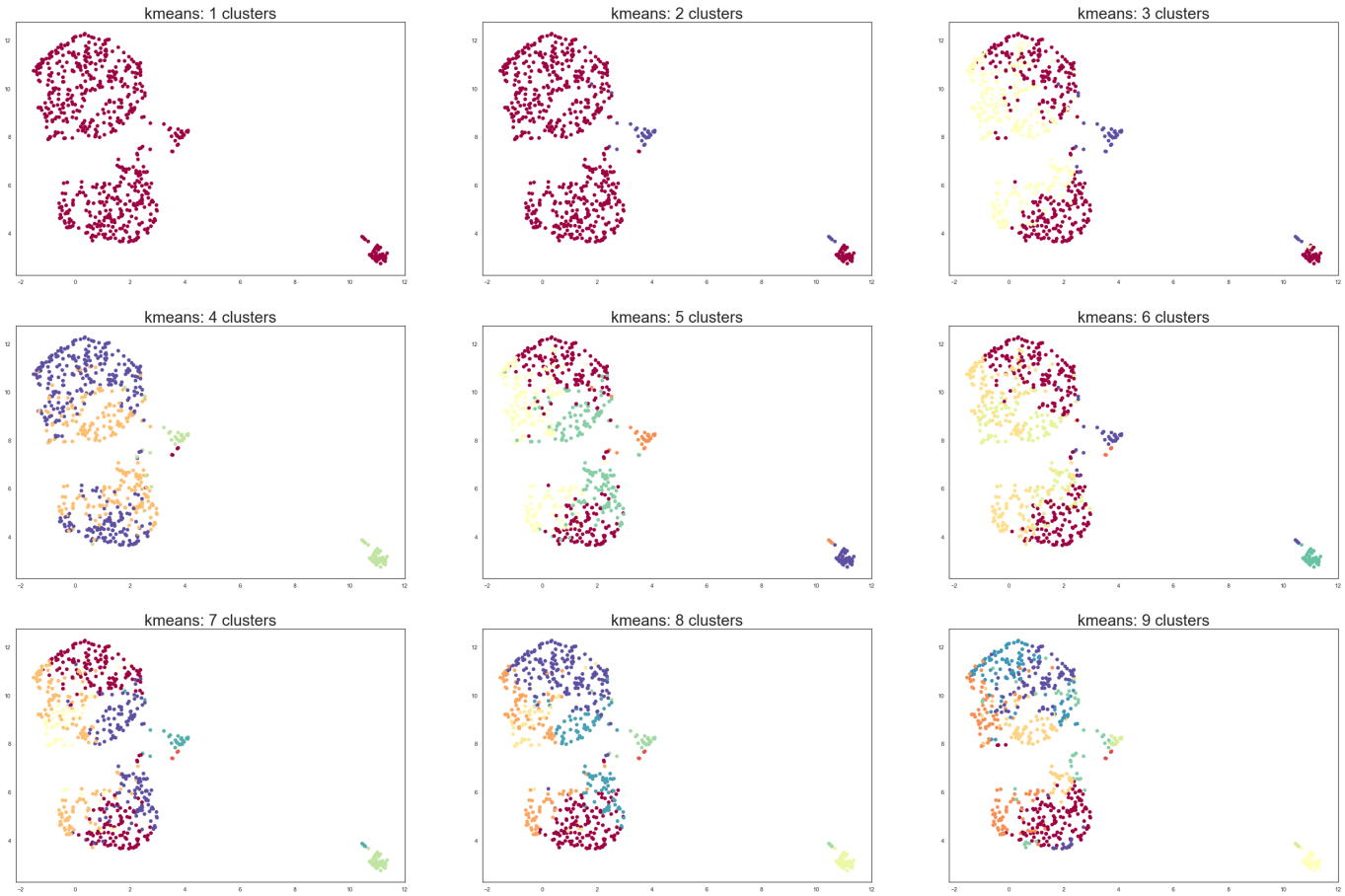
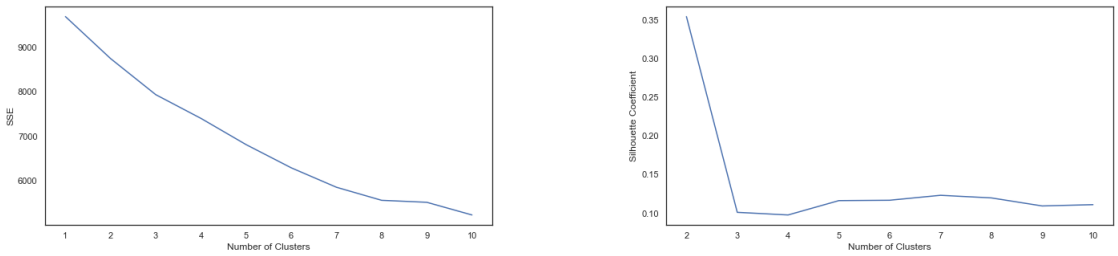


Figure 5.11: K-means clustering with different values for k (number of clusters).

As can be seen in figure 5.12, no real bend or elbow point in the SSE curve can be observed. Neither is a large silhouette coefficient present, apart from the high value at k = 2. Looking at the corresponding distribution in figure 5.11, and comparing the clusters with figure 5.8, it can be observed that k-means was able to distinguish the extremely high scores (`actualDistance`). This is expected, as these scores are still relative outliers compared to the large portion of average player data, and can be seen as a separate group of player personas. However, the goal is to find even more relations within the other cluster about player behaviour. For values of 3 and 4 for k, k-means is able to further subdivide the other data and more viable clustering becomes apparent. The partitioning seems to correlate most with the parameters `min_gap` and `speed`.

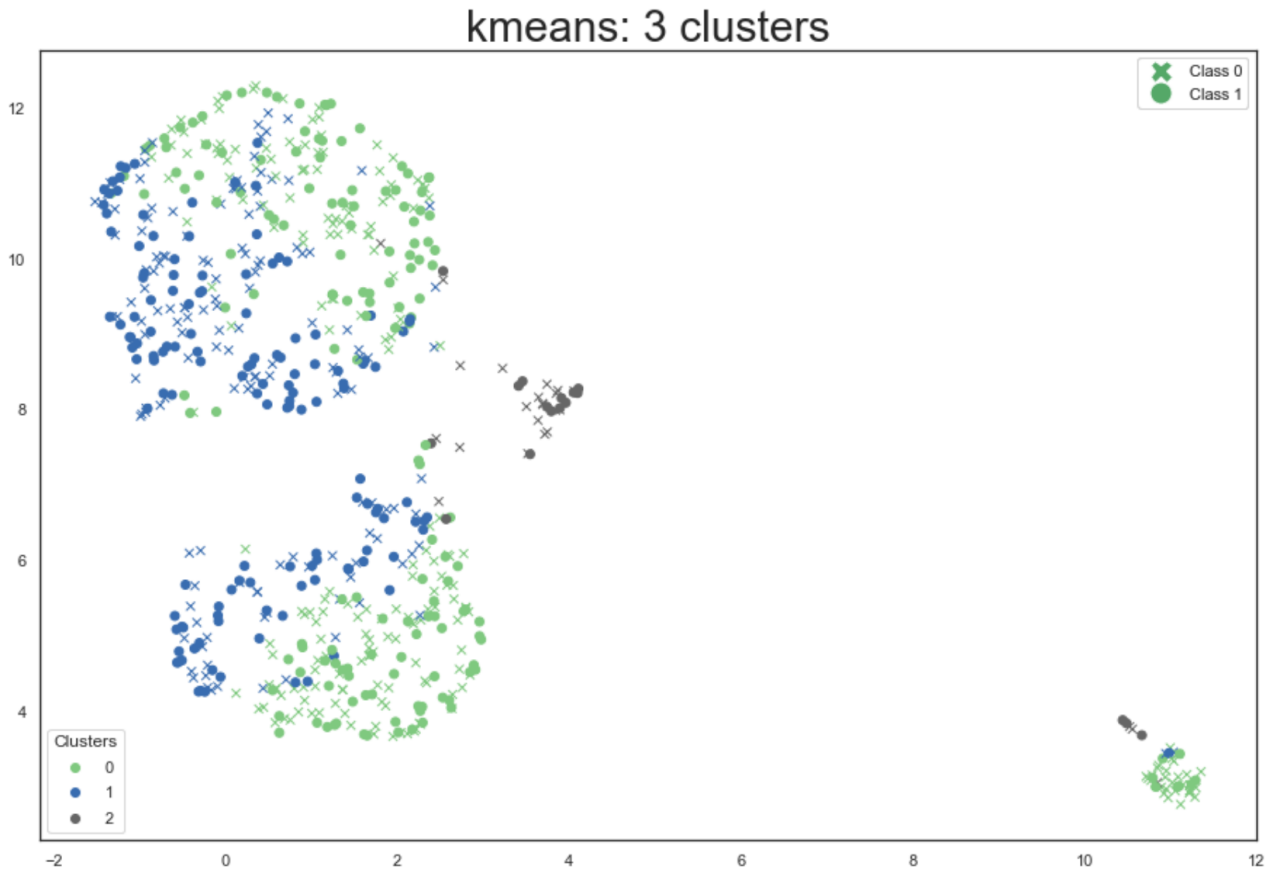


(a) Elbow Point Location

(b) Silhouette Coefficient

Figure 5.12: Methods for choosing the appropriate number of clusters.

To see how player experience correlates with player behaviour, the player experience is plotted in 5.13a with different markers for positive and negative total evaluations, based on formula 1. A game is marked as 'x' is the total estimated score is smaller than or equal to 0.5, and 'o' otherwise. The distributions for each score within the clusters is calculated in figure 5.13b. These distributions are roughly equally divided. If a certain cluster exhibited uneven spreads, it could have given indication of correlations between experienced evaluations and player behaviour that was present within that group, but sadly, this is not up for discussion.



(a) K-means clustering: total score evaluation mapping.

cluster	score 0	score 1
1.0	48.59	51.4
0.0	41.96	58.03
2.0	42.85	57.14

(b) Class distribution per cluster.

Figure 5.13: K-means clustering: evaluations.

Other clustering techniques can be applied to in an attempt to discern clusters of different player behaviour. Agglomerative clustering has been applied for four different linkage criteria. The linkage criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion. The following linkage criteria have been tested:

1. **ward.** minimizes the variance of the clusters being merged.
2. **average.** uses the average of the distances of each observation of the two sets.
3. **complete or maximum.** uses the maximum distances between all observations of the two sets.
4. **single.** uses the minimum of the distances between all observations of the two sets.

Ward linkage is expected to have the overall best performance. Since data parameters consist mostly of gameplay parameters compared to gameplay metrics, complete linkage is attempted to keep games with different gameplay metrics separated as long as possible during the algorithm.

For each linkage criteria, a sweep has been performed again over the hyper parameter k. See appendix D for a full overview of the sweeps for each linkage criterion. The ward agglomerative clustering for 5 clusters shows similar patterns to k-means for 3 clusters, also further dividing the biggest group based on the parameter `min_gap` (see figure 5.14). Other linkage criteria perform poorly.

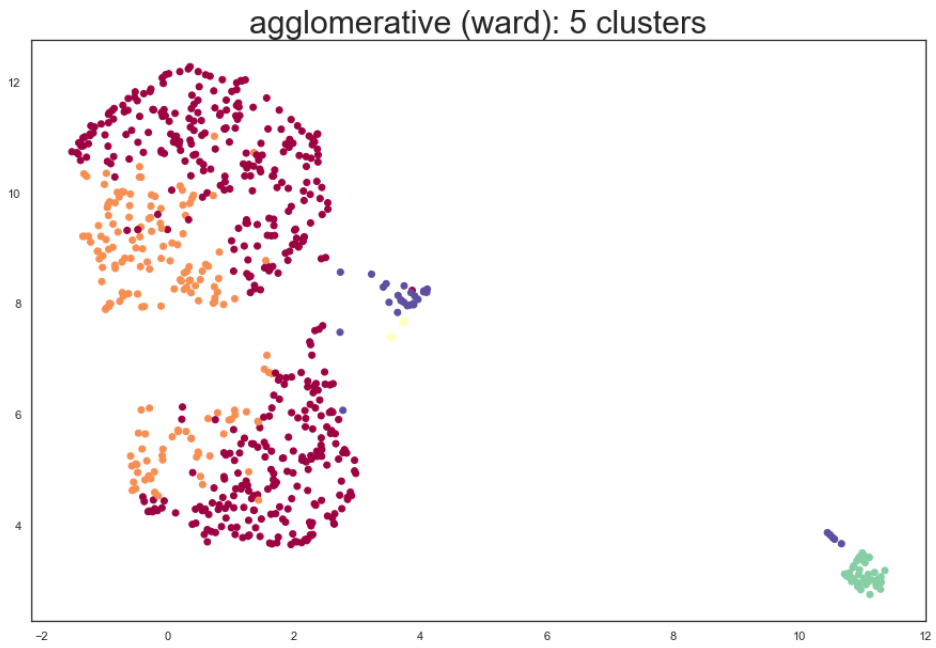


Figure 5.14: Agglomerative clustering with ward linkage.

An approach that does not require predefined number of clusters is DBSCAN. DBSCAN assigns clusters where high densities of data points are situated, separated by low-density regions. Multiple existing distance metrics, as well as custom user defined metrics can be used. The metrics for euclidian and cosine distance are tested but unfortunately lead to extremely poor clustering results.

5.4 Classification

The goal of classification is to predict class labels for new data based on already labeled sample data. In terms of player modeling and player experience, the samples consist of data from player behaviour, combined with a label of player experience. In this case, samples can belong to either of two classes, indicating if the game was positively or negatively evaluated, based on formula 1. A game if labeled as 'class 0' is the total estimated score is lower than or equal to 0.5, and 'class 1' otherwise.

Before any classification can be done, the data needs to be preprocessed. To avoid overfitting the data by training and testing the model on the same data, the dataset has to be split in a training set and a test set. The test set is a part of the data that is hold out and unseen by the model. The original dataset of 693 entries is split into a 75%-25% train-test distribution each containing 519 and 174 entries correspondingly. Both train and test set should have the same proportions of class label counts as the original dataset. Both the original dataset, training set and test set have a distribution of 45% and 55% over class 0 and class 1 respectively.

There is still a risk of overfitting on the test set when the hyperparameters are tweaked until the estimator performs optimally. To solve this problem, cross-validation (CV) is applied, in which other parts of the dataset are held out in a k-fold, by splitting the training set into k smaller sets, training the model on k-1 of those sets and validating with the remaining part. The performance measure reported by k-fold cross-validation is then the average over all k values. After the model is tuned, final evaluation can be done on the test set.

A list of different classification techniques have been applied: logistic regression (with and without inner CV), decision trees, k-nearest neighbors, linear discriminant analysis, gaussian naive bayes and support vector machines. Decision trees are expected to perform good, since they are well-suited for this problem. Figure 5.15 gives an overview of the accuracy on cross-validation, training and test sets for the algorithms. The realized accuracies are very weak and are situated within the 50% to 60% range. To attempt to improve the results, ensemble methods have been applied that combine the predictions of several base estimators. These include gradient boosting, adaboost and histogram boosting. These methods perform slightly better, but only meagerly.

method	cv_accuracy	cv_standard_deviation	training_accuracy	test_accuracy
Logistic Regression	57.41	4.13	59.15	55.17
Logistic Regression CV	55.1	1.11	58.95	55.74
Decision Trees	59.15	3.21	99.8	48.85
Decision Trees Unscaled	55.67	3.75	99.8	51.14
K Nearest Neighbors	53.94	3.3	68.2	51.14
Linear Discriminant Analysis	57.79	4.31	58.95	55.74
Gaussian Naive Bayes	53.38	6.37	64.73	52.29
Support Vector Machines	56.64	1.14	72.44	57.47
Random Forest	53.95	0.64	65.51	56.32
Gradient Boosting	56.06	3.91	79.96	54.02
AdaBoost	56.26	2.1	79.38	55.17
Histogram Boosting	53.95	3.5	99.8	53.44

Figure 5.15: Classification methods and accuracy.

Based on the previous assumption that games within a session of multiple consecutive games are less prone to noise, it could be useful to limit the dataset to only those gameplays. The same classification algorithms have been applied to gameplays which are part of sessions of at least 8 consecutive games. This threshold is based on the distribution of consecutive games (see figure 4.12), as the majority of games are played in sessions with up to a maximum of 8 games. See figure 5.16 for an overview of the results.

method	cv_accuracy	cv_standard_deviation	training_accuracy	test_accuracy
Logistic Regression	45.71	6.22	57.97	53.96
Logistic Regression CV	49.41	5.81	52.12	66.66
Decision Trees	50.99	6.88	100.0	44.44
Decision Trees Unscaled	51.59	1.93	100.0	57.14
K Nearest Neighbors	48.43	6.73	69.68	49.2
Linear Discriminant Analysis	47.88	5.42	57.44	53.96
Gaussian Naive Bayes	48.89	4.44	55.31	36.5
Support Vector Machines	51.06	4.87	80.85	47.61
Random Forest	53.78	5.21	72.34	58.73
Gradient Boosting	55.33	5.48	95.21	55.55
AdaBoost	51.03	3.85	95.21	53.96
Histogram Boosting	53.76	4.47	98.93	46.03

Figure 5.16: Classification methods and accuracy: games in sessions with at least 8 consecutive games.

Unexpectedly, the results are unsatisfactory and even more weak than the global classifiers. Logistic regression (with inner CV) performs slightly better on the test set, but is unable to generalize from the full data. The constructed decision trees have been attempted to be visualized and analyzed to gain insight into which parameters are the most determining factors for player enjoyment. Unfortunately, many iterations gave different results and no real deciding parameters could be distinguished.

5.5 Challenges

As discussed in section 2.4.2, the collection of ranked-based data as opposed to ratings or classes is preferred since it has less statistical limitations. However, ranking-based data is harder to employ in existing AI methods, which are often based on ordinal data. Simply transforming ranked-based data into ordinal data is considered bad practice, as this presents the same statistical limitations that other forms of data have. Care has to be taken to not break these axiomas. Working with ranked-based data posed a lot of new practical issues. Nontheless, considering the increase in integrity of the data, the extra work is worth the effort.

5.6 Conclusion

A few interesting observations could be made from the descriptive analysis which led to intuitive assumptions about player behaviour and experience. Unfortunately, none of these assumptions could be validated with the succeeding methods. UMAP was able to reduce the space into three groups that were based mostly on game parameters, but was unable to generalize from game metrics. Neither dimensionality reduction methods, clustering or classification methods seem to be able to reduce the space into groups of player behaviour, but lean towards trends in game difficulty parameters. This could be due to the fact that a relatively small amount of player metrics that describe gameplay features are gathered, which are not able to provide enough information about player behaviour. Due to the nature of the simplicity of the selected game, no other game metrics could have been gathered for Chrome Dino. A different game could have given opportunity to more gathered game metrics and generalizations. Upon retrospective, the number of game metrics could have been a fitting parameter to taken into consideration during game selection. A player model that includes more behavioral aspects could yield interesting observations.

A workaround for this dataset could be to use a custom defined distance metric that assigns different weights to different parameters, and apply it in clustering or classification algorithms, such as DBScan or k-nearest neighbors. By giving more weights to player metrics instead of game parameters, the models might be able to generalize from player behaviour instead of game parameters and increase accuracies.

Other reasons could be that the data is too noisy. Capturing subjective emotions in an unsupervised environment is difficult and prone to errors. Even though the data was thoroughly preprocessed and participants were only asked three questions that are answered with one press of a button, a lot of noise could still be present. It is also possible that not enough sample points are collected for global generalisations to be made from the gathered data.

6 Conclusions and Future Work

In this work, an attempt was made at constructing player models for player behaviour in Chrome Dino, a web-based video game, using AI methods. Data collection was done on human subjects over the internet to gain insight into relationships between game difficulty parameters, combined with game metrics and player enjoyment. Multiple methods were applied to analyze player behaviour, including dimensionality reduction using UMAP, clustering and classification. No clear relationships were found.

Common flaws mostly point towards a lack of quantity of game metrics that capture enough information about player behaviour. Even for a simple game as Chrome Dino, it seems to appear that more gameplay features are necessary. A player model that includes more behavioral aspects could yield interesting observations. It could also be possible that the data was too noisy or contained not enough information. More data could be gathered, either in a more controlled setting with human participant, or by generating artificial input data with AI methods. Applying the methods for player modeling to a larger dataset could make it possible to make more reliable generalizations about player behaviour and form a basis for more sound player models.

References

[1] The fun of gaming: Measuring the human experience of media enjoyment. *FUGA*, 2006-2009.

[2] Anders Drachen, Christian Thurau, Julian Togelius, Georgios N. Yannakakis, and Christian Bauckhage. Game Data Mining. *Game Analytics*, pages 205–253, 2013.

[3] Adam Pearce. Andy Coenen. Understanding umap. https://pair-code.github.io/understanding-umap/.

[4] Kevin Arvai. K-Means Clustering in Python: A Practical Guide. 2020.

[5] P.W.D. Charles. t-rex-runner. https://github.com/wayou/t-rex-runner, 2013.

[6] Dagmara Dziedzic and Wojciech Włodarczyk. Approaches to measuring the difficulty of games in dynamic difficulty adjustment systems. *International Journal of Human–Computer Interaction*, 34(8):707–715, 2018.

[7] Renê Gusmão, Kennet Calixto, and Caetano Segundo. Dynamic difficulty adjustment through parameter manipulation for space shooter game. 09 2015.

[8] Danial Hooshyar, Moslem Yousefi, and Heuiseok Lim. Data-driven approaches to game player modeling: A systematic literature review. *ACM Computing Surveys*, 50:1–19, 01 2018.

[9] IJsselsteijn, W. A., de Kort, Y. A. W., & Poels, K. The Game Experience Questionnaire. 2013.

[10] Martin Jennings-Teats, Gillian Smith, and Noah Wardrip-Fruin. Polymorph: Dynamic difficulty adjustment through level generation. 06 2010.

[11] Giel Lankveld, Pieter Spronck, and Matthias Rauterberg. Difficulty scaling through incongruity. *Bijdragen*, 01 2008.

[12] RA Likert. A technique for measurement of attitudes. *Archives of Psychology*, 22:1–, 01 1932.

[13] Diana Lora, Antonio Sánchez-Ruiz, Pedro Gonzalez-Calero, and Marco Gómez-Martín. Dynamic difficulty adjustment in tetris. 03 2016.

[14] M. Csikszentmihalyi. Flow: the Psychology of Optimal Experience. Harper Collins.

[15] Nacke, L. E. Flow in games: Proposing a flow experience model. Proceedings of the workshop on conceptualising, operationalising and measuring the player experience in videogames at fun and games. page 104–108, 2012.

[16] Newzoo | Games, Esports & Mobile Market Intelligence. 2020 Global Esports Market Report. 2020. https://newzoo.com/.

[17] Chris Pedersen, Julian Togelius, and Georgios Yannakakis. Optimization of platform game levels for player experience. 01 2009.

[18] Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling player experience in super mario bros. pages 132–139, 2009.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Johannes Pfau, Jan Smeddinck, and Rainer Malaka. Enemy within: Long-term motivation effects of deep player behavior models for dynamic difficulty adjustment. 04 2020.

[21] Mark Riedl and Alexander Zook. Ai for game production. *IEEE Conference on Computatonal Intelligence and Games, CIG*, pages 1–8, 08 2013.

[22] Nigel Robb and Bo Zhang. Empirical evaluation of player experience using a machine-learning approach to dynamic difficulty adjustment in video games. 06 2020.

[23] Robin Hunicke, Vernell Chapman . AI for Dynamic Difficulty Adjustment in Games .

[24] Timo Saari, Marko Turpeinen, Kai Kuikkaniemi, Ilkka Kosunen, and Niklas Ravaja. Emotionally adapted games – an example of a first person shooter. 5613:406–415, 07 2009.

[25] Gabriel Sepulveda, Felipe Besoain, and Nicolas A. Barriga. Exploring dynamic difficulty adjustment in videogames. pages 1–6, 11 2019.

[26] Mirna Silva, Victor Silva, and Luiz Chaimowicz. Dynamic difficulty adjustment through an adaptive ai. pages 173–182, 11 2015.

[27] Mirna Silva, Victor Silva, and Luiz Chaimowicz. Dynamic difficulty adjustment on moba games. *Entertainment Computing*, 18, 10 2016.

[28] Jan Smeddinck, Regan Mandryk, Max Birk, Kathrin Gerling, Dietrich Barsilowski, and Rainer Malaka. How to present game difficulty choices?: Exploring the impact on player experience. pages 5595–5607, 05 2016.

[29] Kalyani Sonawane. Serious Games Market - Global Opportunity Analysis and Industry Forecast, 2016-2023. 2015.

[30] Miron-Shatz Talya and Arthur Stone. Memories of yesterday's emotions: Does the valence of experience affect the memory-experience gap? *Emotion (Washington, D.C.)*, 9:885–91, 12 2009.

[31] Thomas W.Malone. What makes computer games fun? page 258–277, r 1981.

[32] Tim Tijs, Dirk Brokken, and Wijnand Ijsselsteijn. Dynamic game balancing by recognizing affect. pages 88–93, 10 2008.

[33] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi Zaman. Dynamic difficulty adjustment for maximized engagement in digital games. pages 465–471, 04 2017.

[34] Georgios Yannakakis, Roddy Cowie, and Carlos Busso. The ordinal nature of emotions. pages 248–255, 10 2017.

[35] Georgios Yannakakis and John Hallam. Capturing player enjoyment in computer games. 71:175–201, 06 2007.

[36] Georgios Yannakakis and John Hallam. Game and player feature selection for entertainment capture. pages 244–251, 05 2007.

[37] Georgios Yannakakis and John Hallam. Towards optimizing entertainment in computer games. *Applied Artificial Intelligence*, 21:933–971, 11 2007.

[38] Georgios Yannakakis and Héctor Martínez. Ratings are overrated! *Frontiers in ICT*, 2, 07 2015.

[39] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer, 2018. http://gameaibook.org.

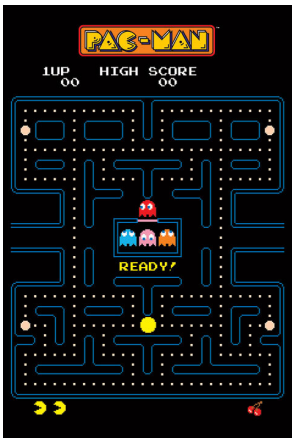
[40] Yannakakis G.N., Hallam J. Ranking vs. Preference: A Comparative Study of Self-reporting. *Lecture Notes in Computer Science*, 6974, 2011.

[41] Mohammad Zohaib. Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018:1–12, 11 2018.

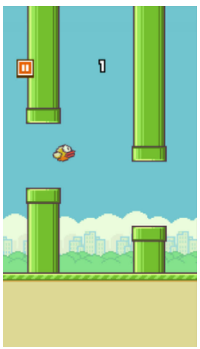
[42] Alexander Zook and Mark Riedl. A temporal data-driven player model for dynamic difficulty adjustment. *Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2012*, 01 2012.

A Game Selection Analysis

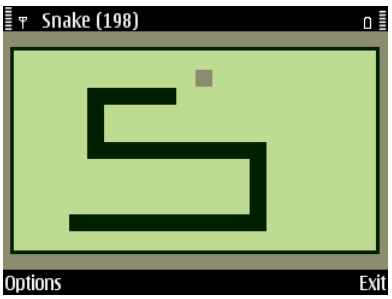
(a) Chrome Dino



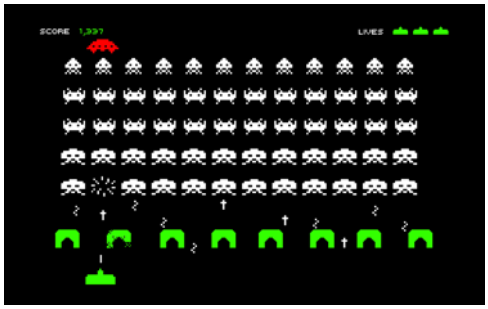
(b) Pacman



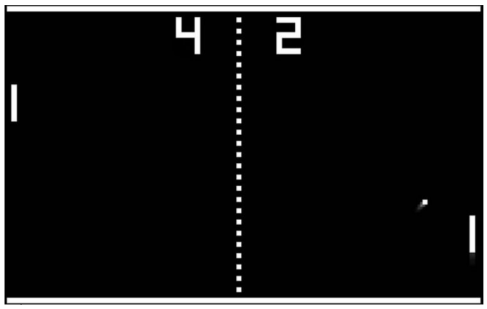
(c) Flappy Bird



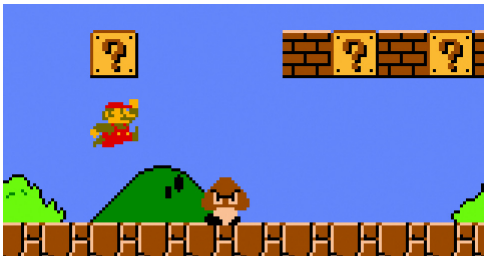
(d) Snake



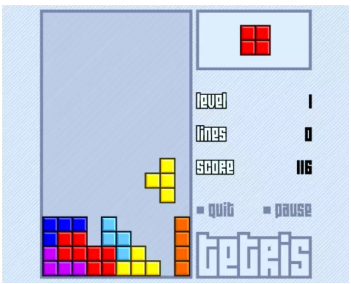
(e) Space Invaders



(f) Pong

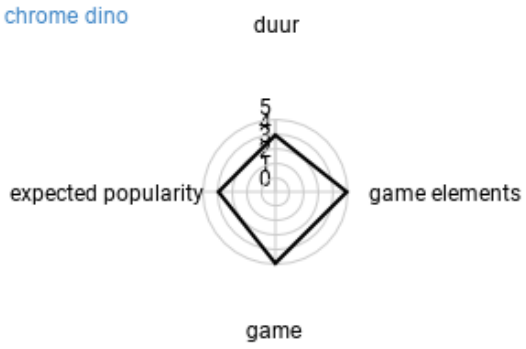


(g) Super Mario Bros

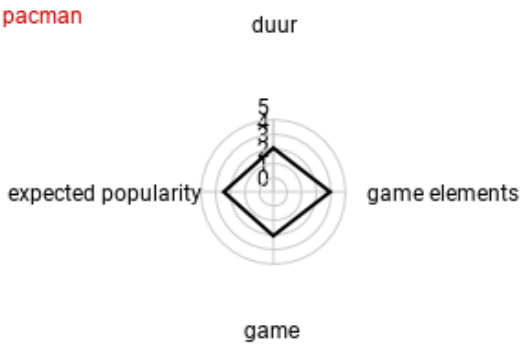


(h) Tetris

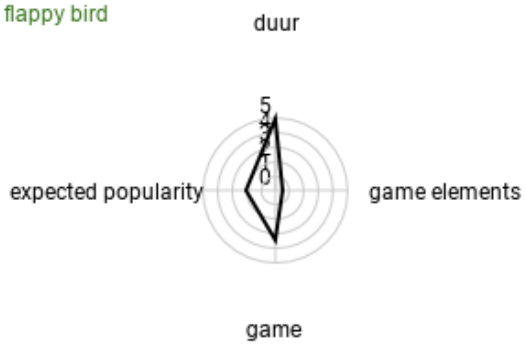
Figure A.1: Video games considered for selection.



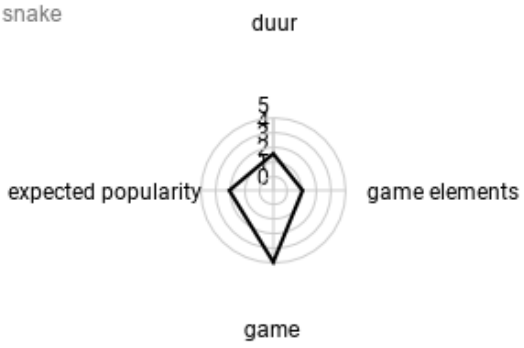
(a) Chrome Dino



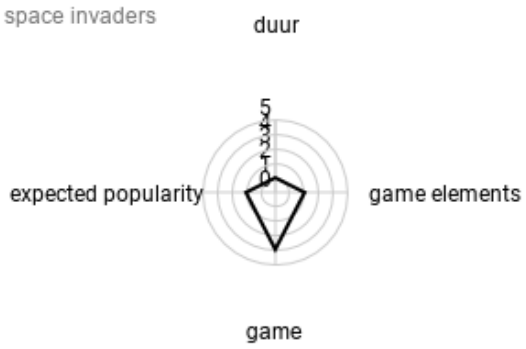
(b) Pacman



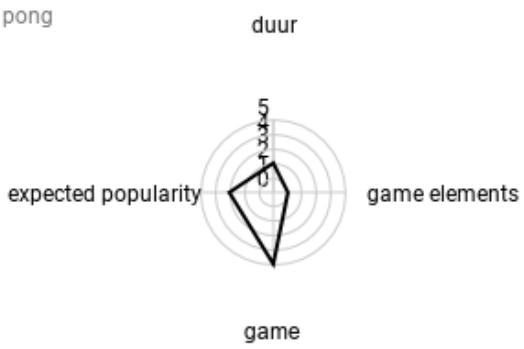
(c) Flappy Bird



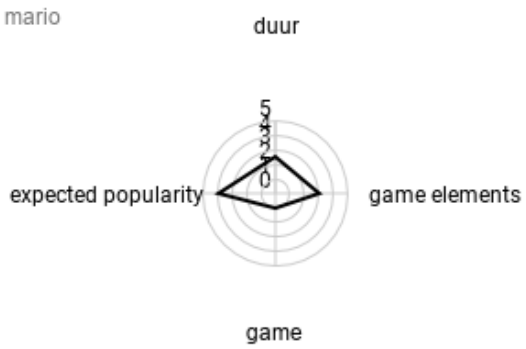
(d) Snake



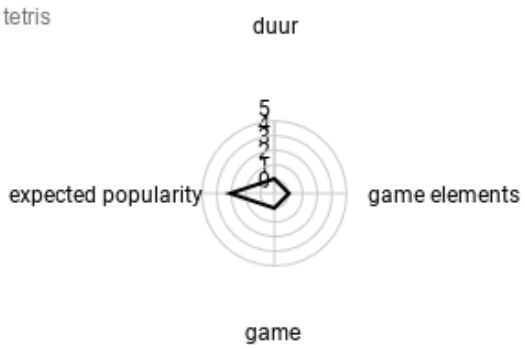
(e) Space Invaders



(f) Pong



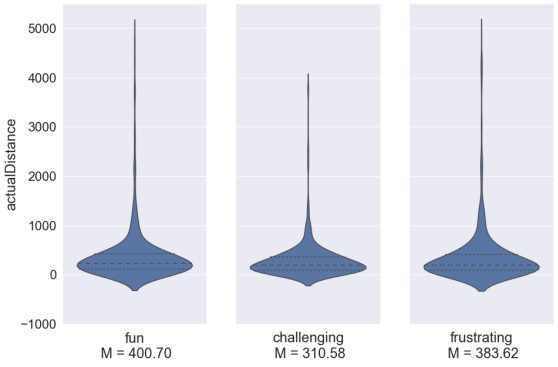
(g) Super Mario Bros



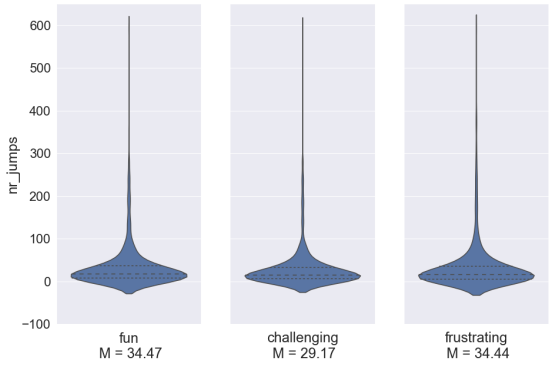
(h) Tetris

Figure A.2: Analysis of the selected video games.

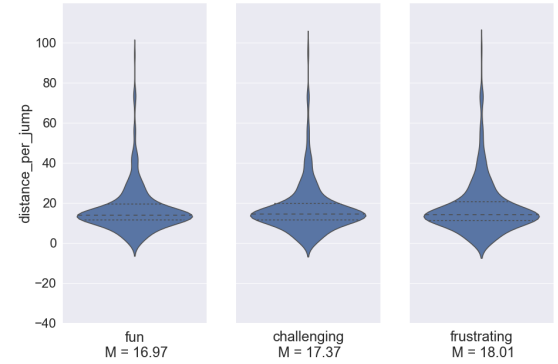
B Global Analysis



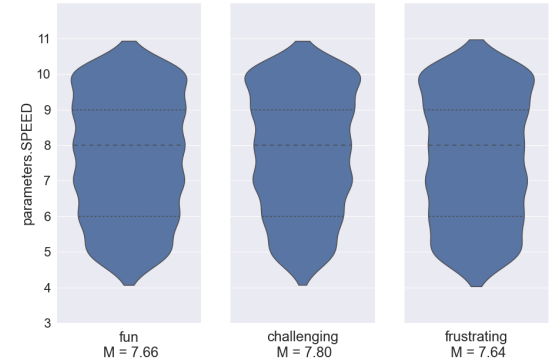
(a) Achieved score



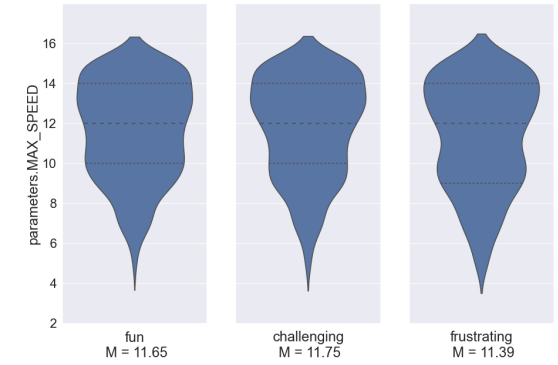
(b) Total number of jumps



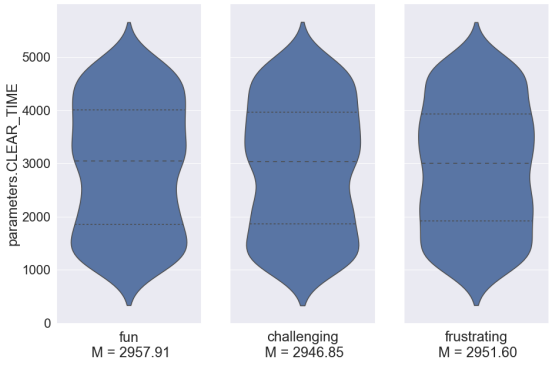
(c) Distance per jump



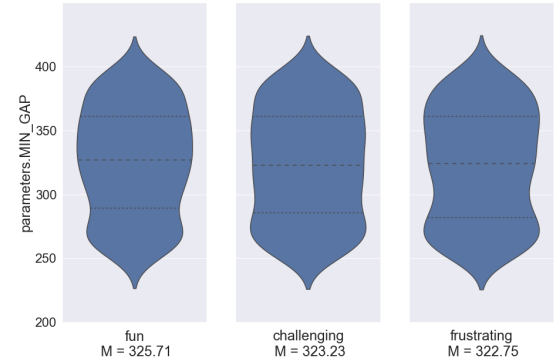
(d) Speed



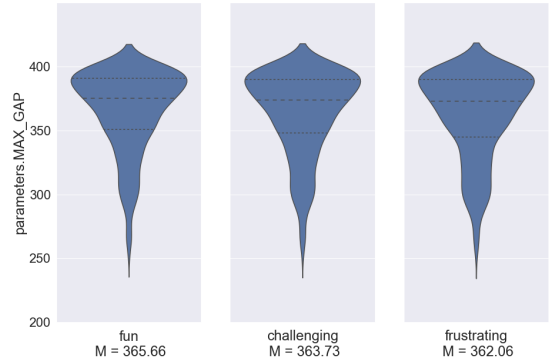
(e) Maximum speed



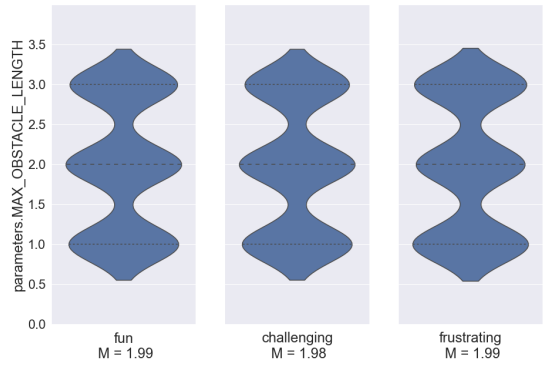
(f) Clear time



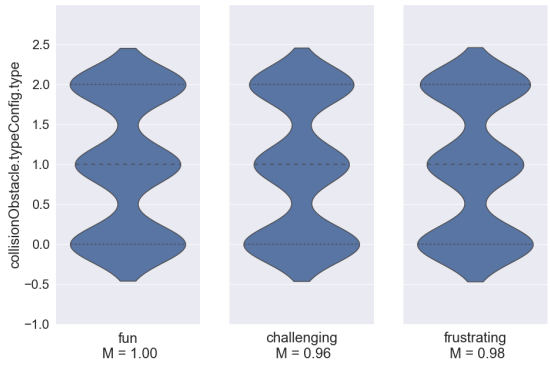
(g) Minimum gap



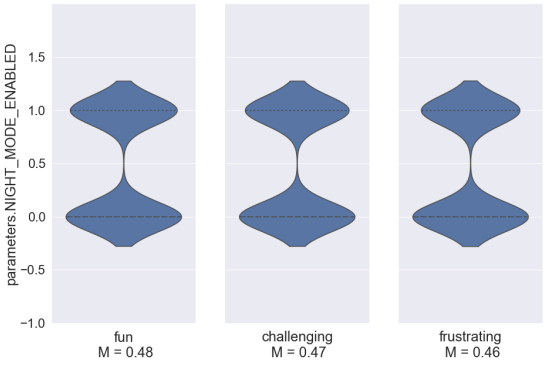
(h) Maximum Gap



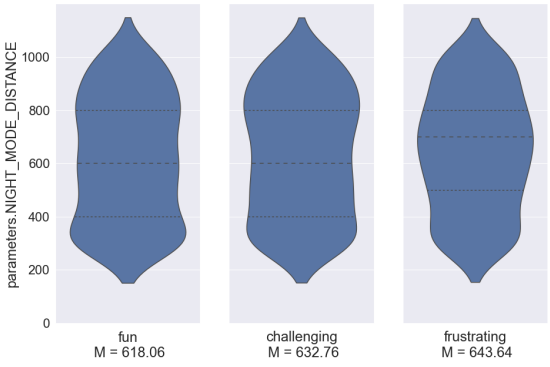
(i) Obstacle length



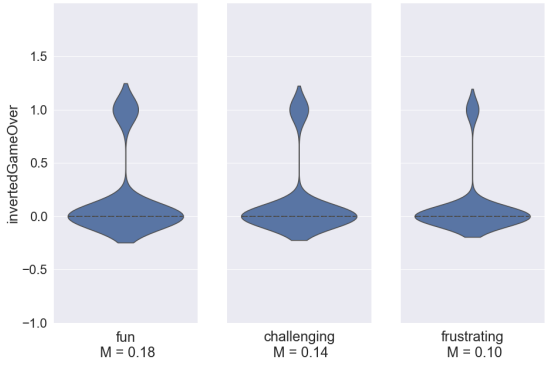
(j) Collision obstacle



(k) Night mode enabled

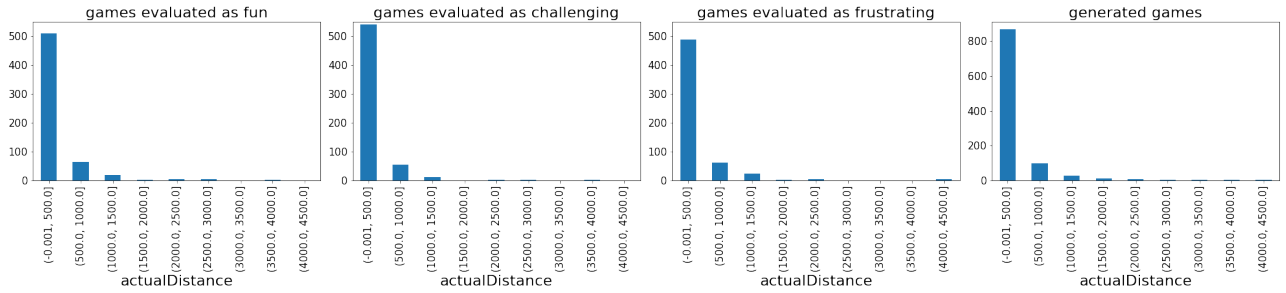


(l) Night mode distance

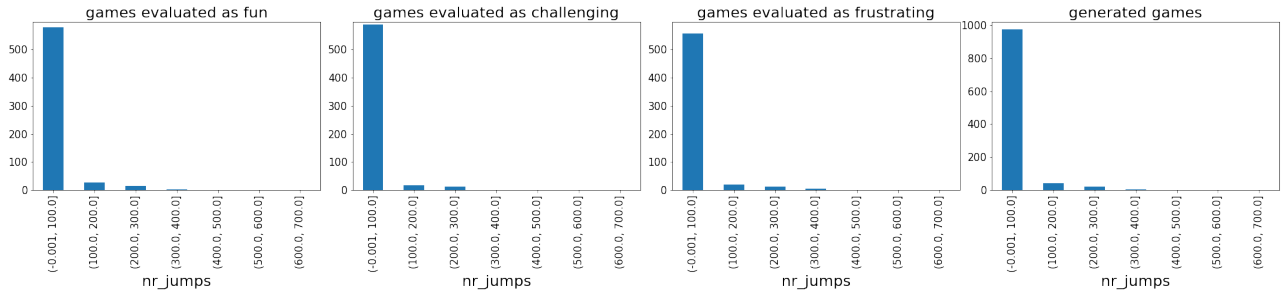


(m) Game over in night-mode

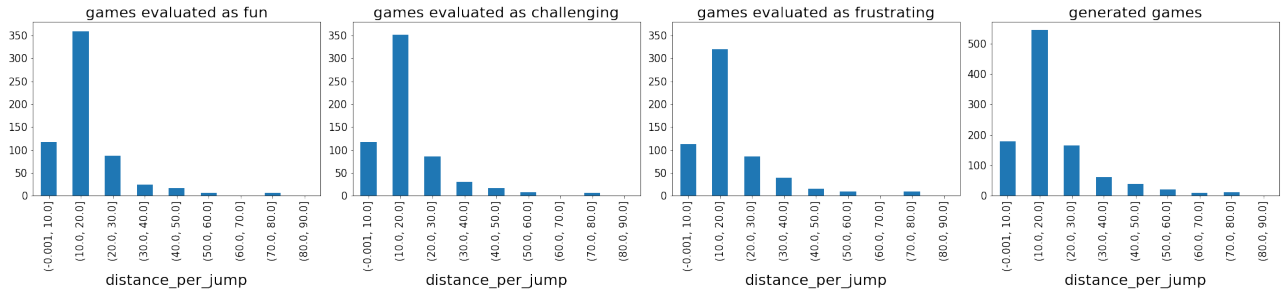
Figure B.1: Violin plots per emotional state for different game parameters and metrics.



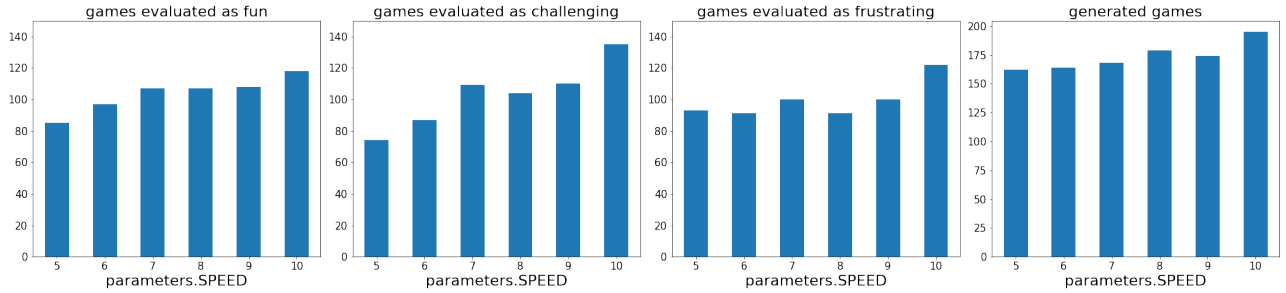
(a) Achieved score



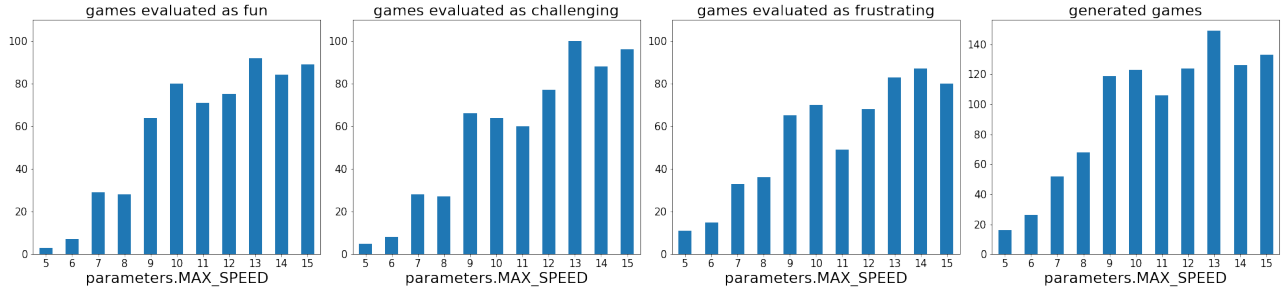
(b) Total number of jumps



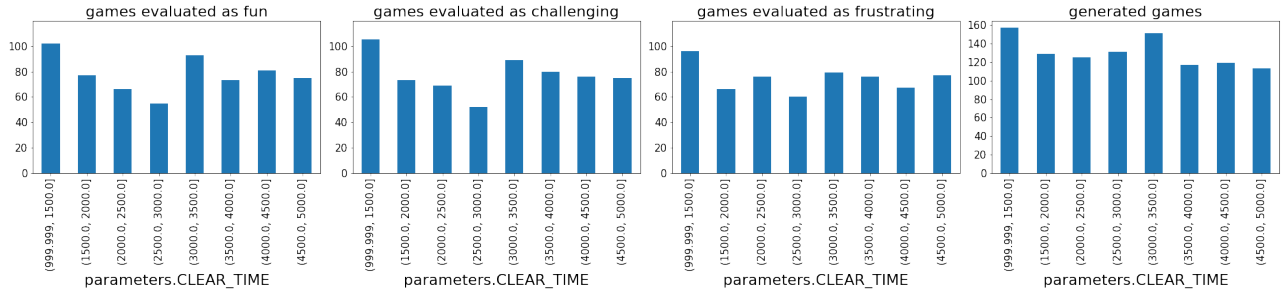
(c) Distance per jump



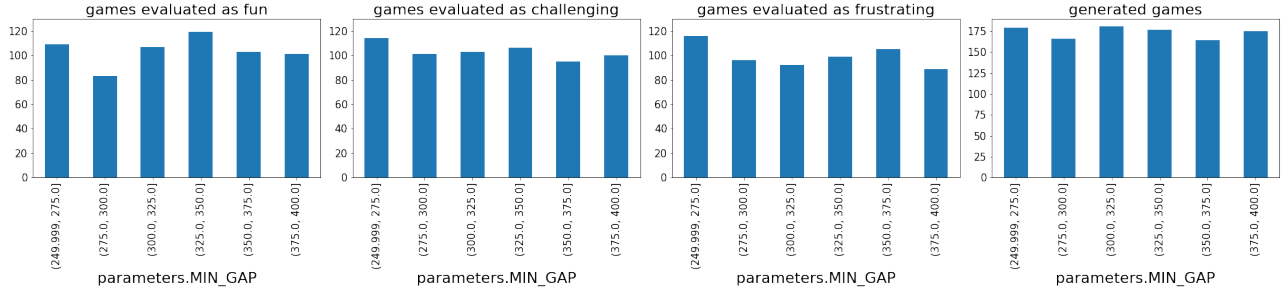
(d) Speed



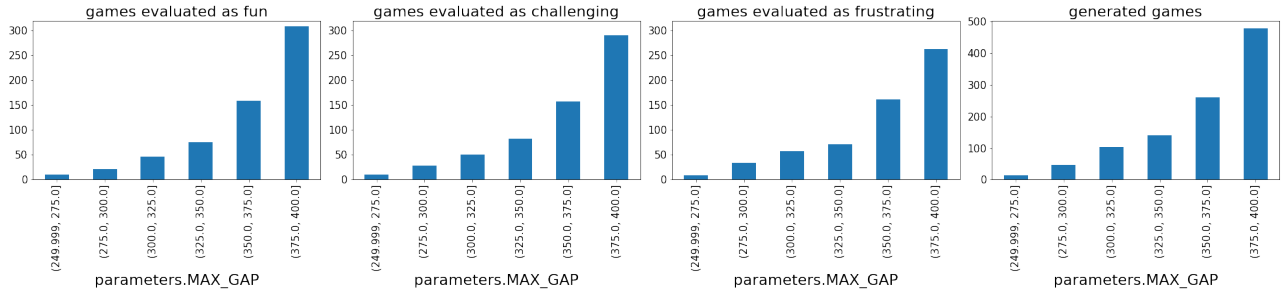
(e) Maximum speed



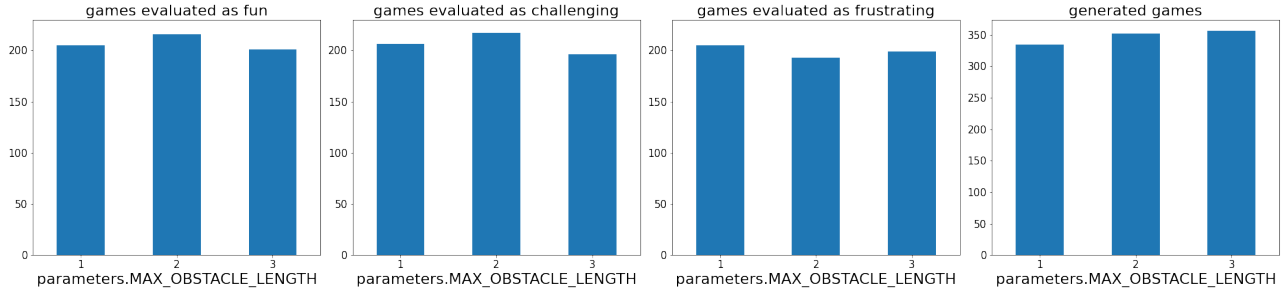
(f) Clear time



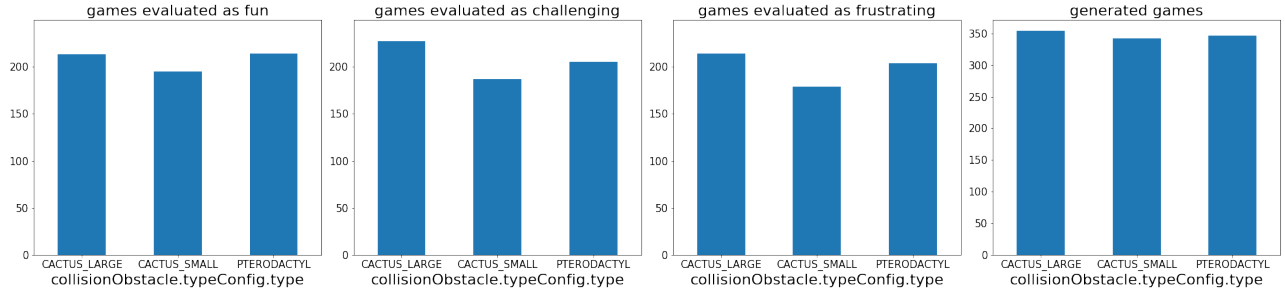
(g) Minimum gap



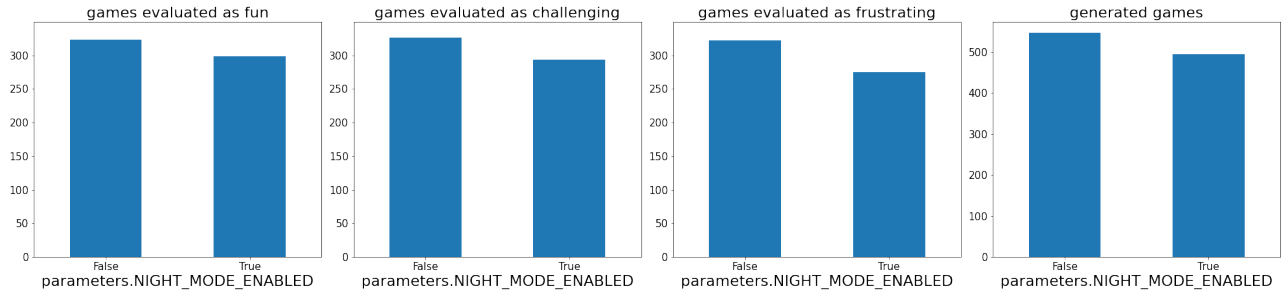
(h) Maximum gap



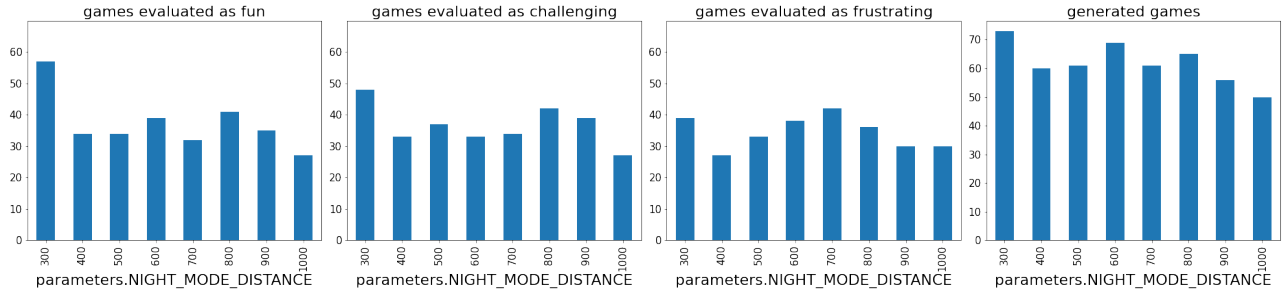
(i) Obstacle length



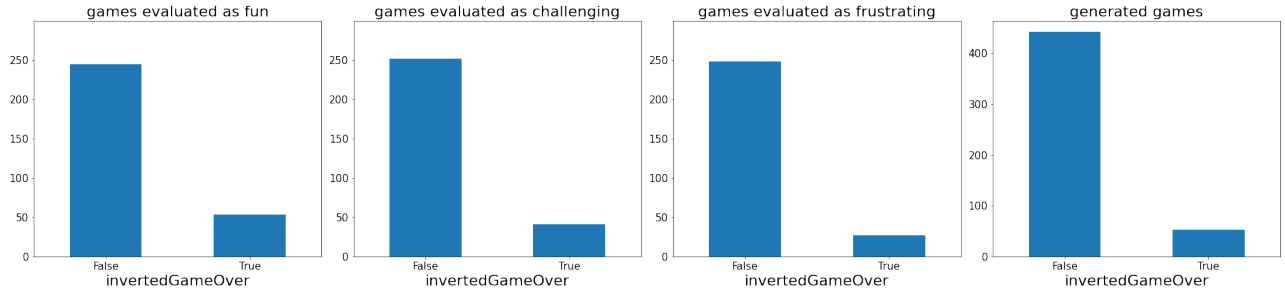
(j) Collision obstacle



(k) Night mode enabled

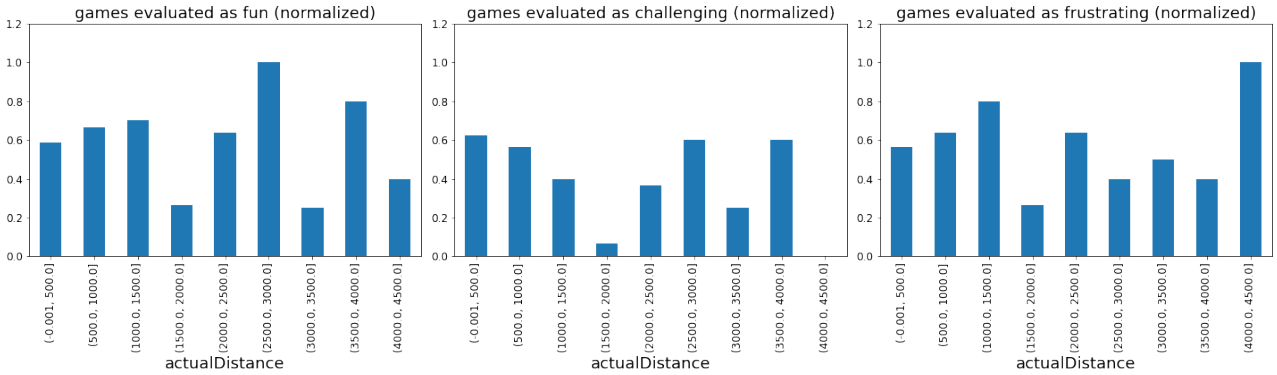


(l) Night mode distance

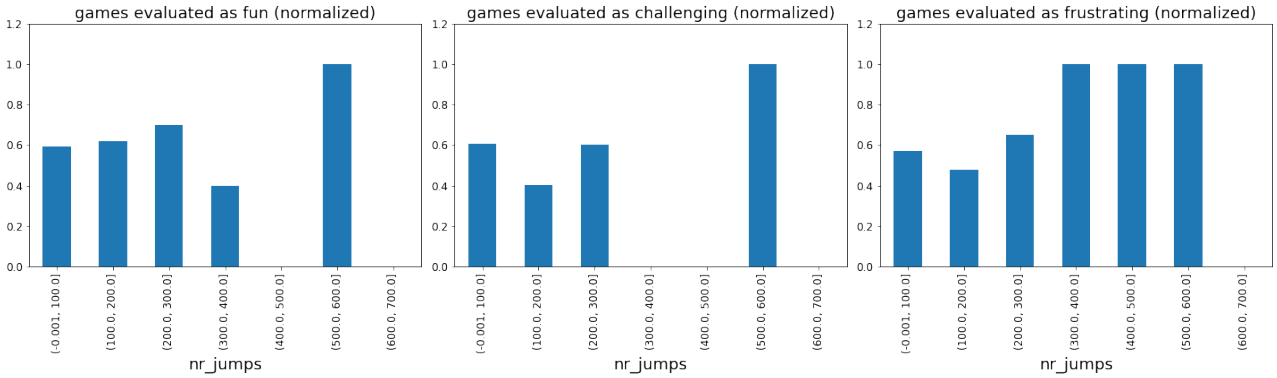


(m) Game over in night-mode

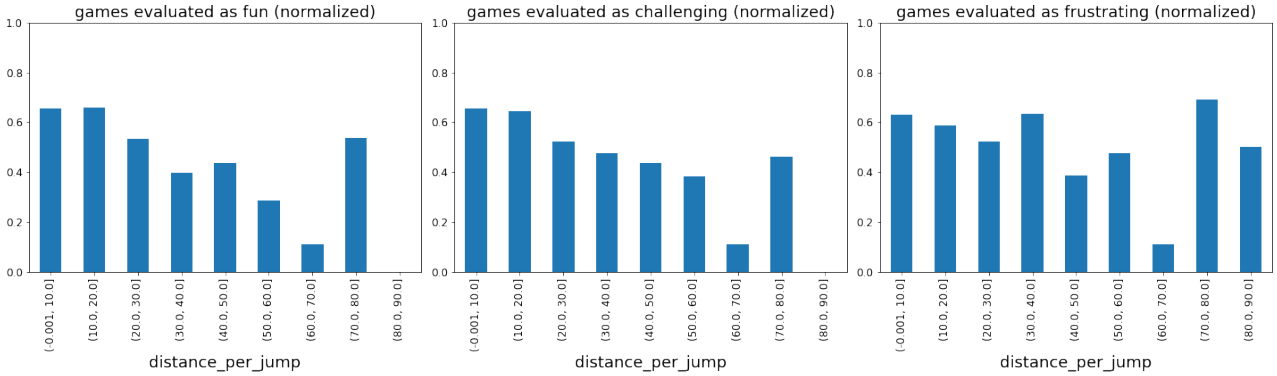
Figure B.2: Bar charts per emotional state for different game parameters and metrics, supplemented with total generated games for specified parameters or metric.



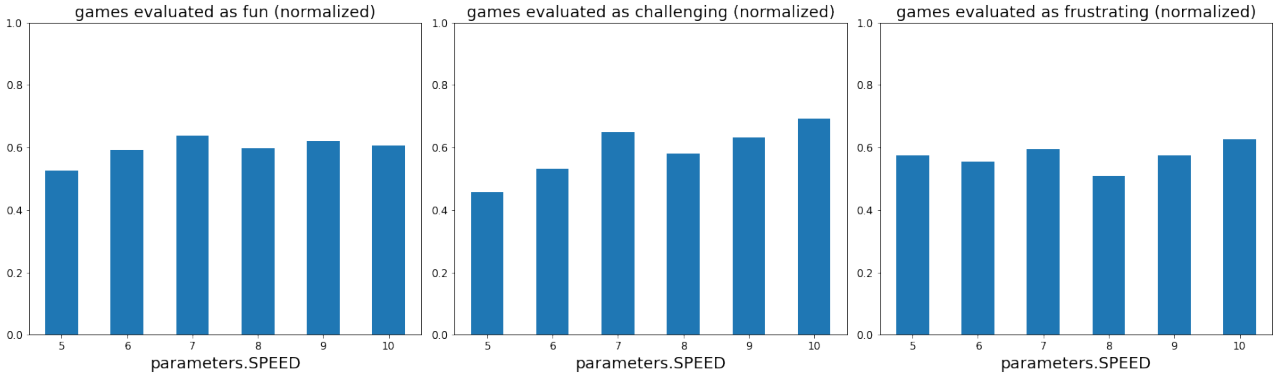
(a) Achieved score



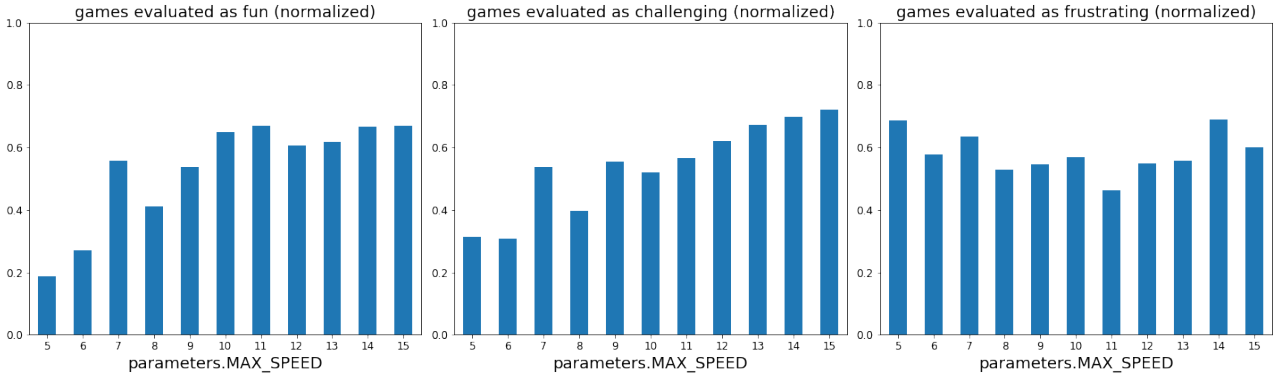
(b) Total number of jumps



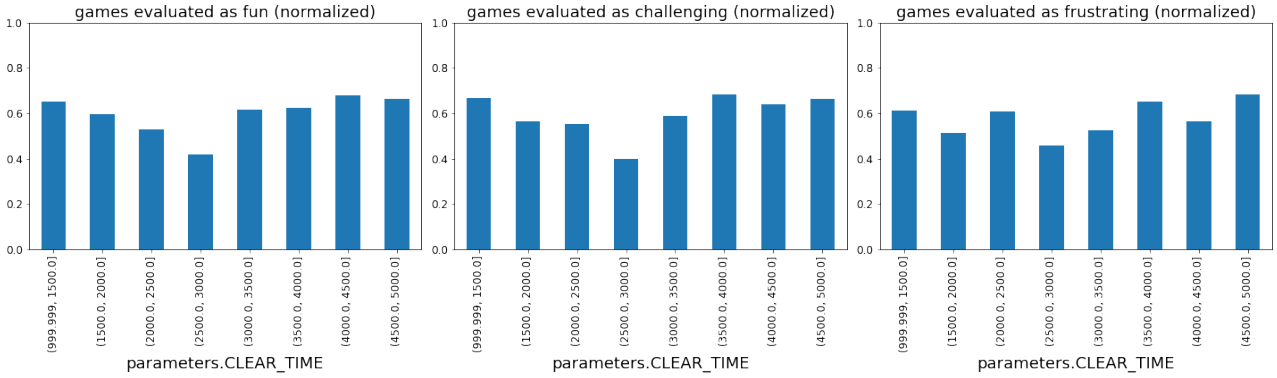
(c) Distance per jump



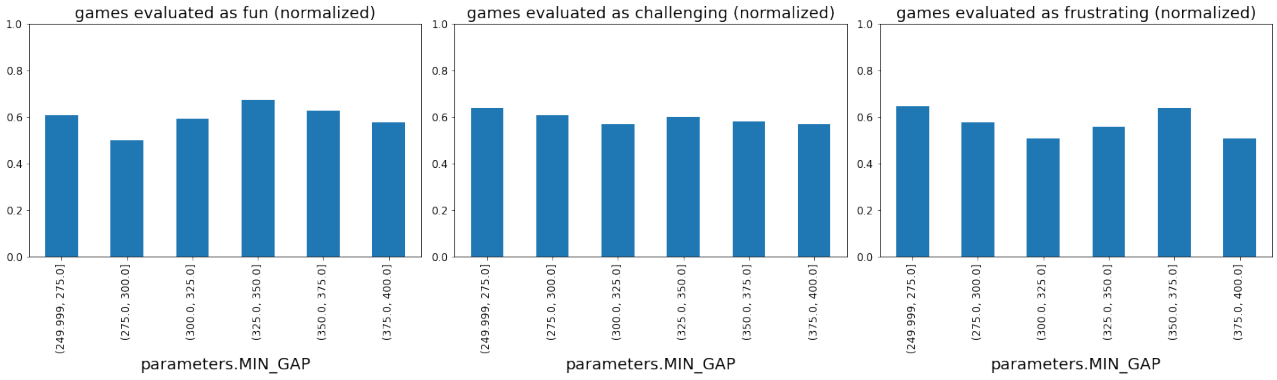
(d) Speed



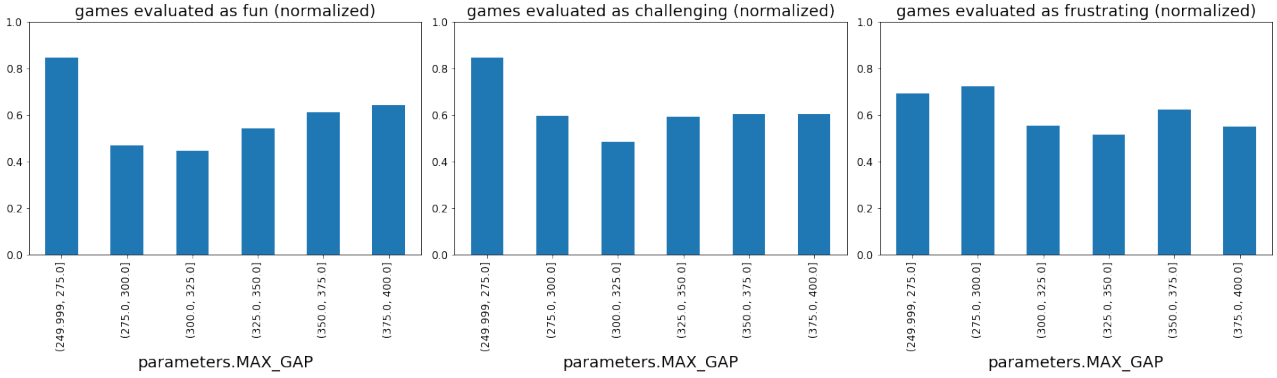
(e) Maximum speed



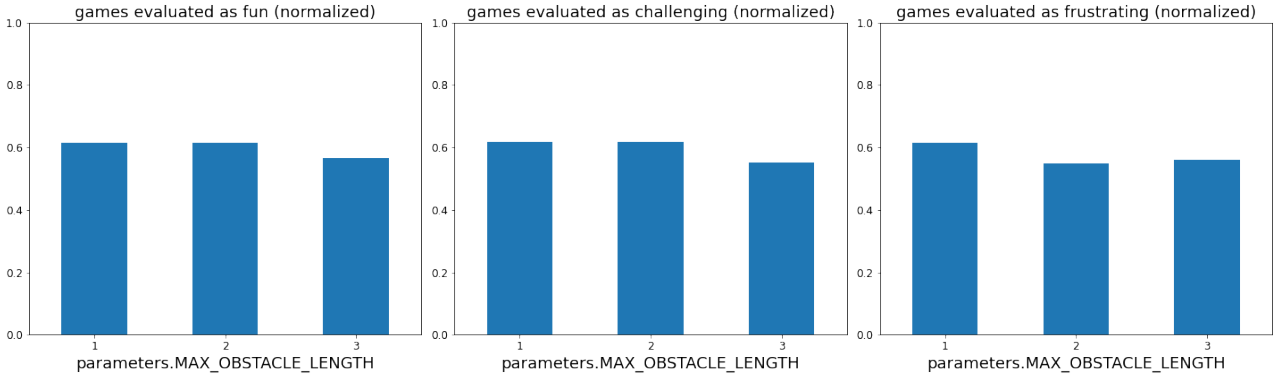
(f) Clear time



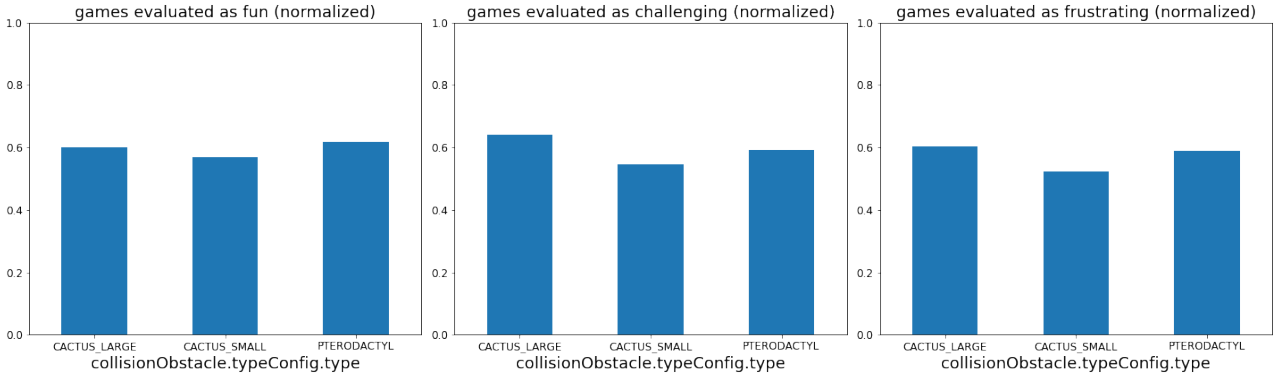
(g) Minimum gap



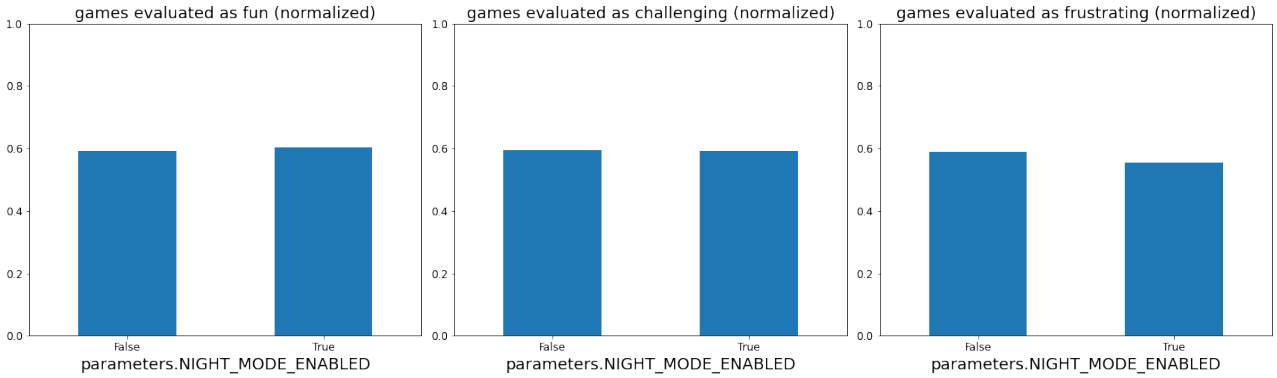
(h) Maximum gap



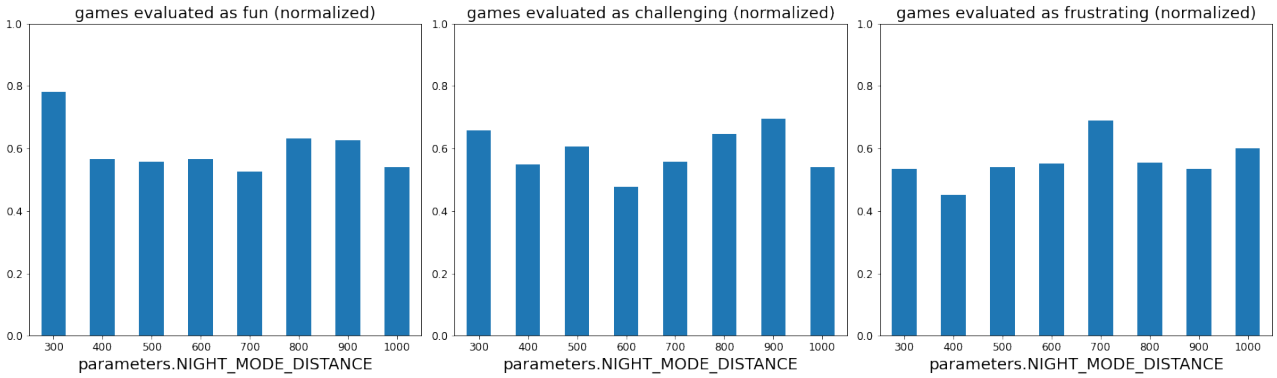
(i) Obstacle length



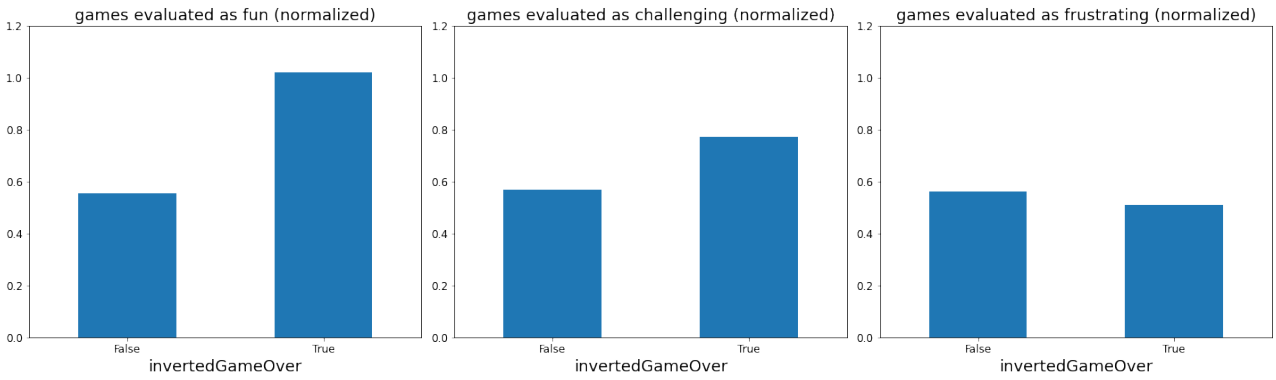
(j) Collision obstacle



(k) Night mode enabled



(l) Night mode distance



(m) Game over in night-mode

Figure B.3: Normalized bar charts per emotional state for different game parameters and metrics.

C UMAP Sweep

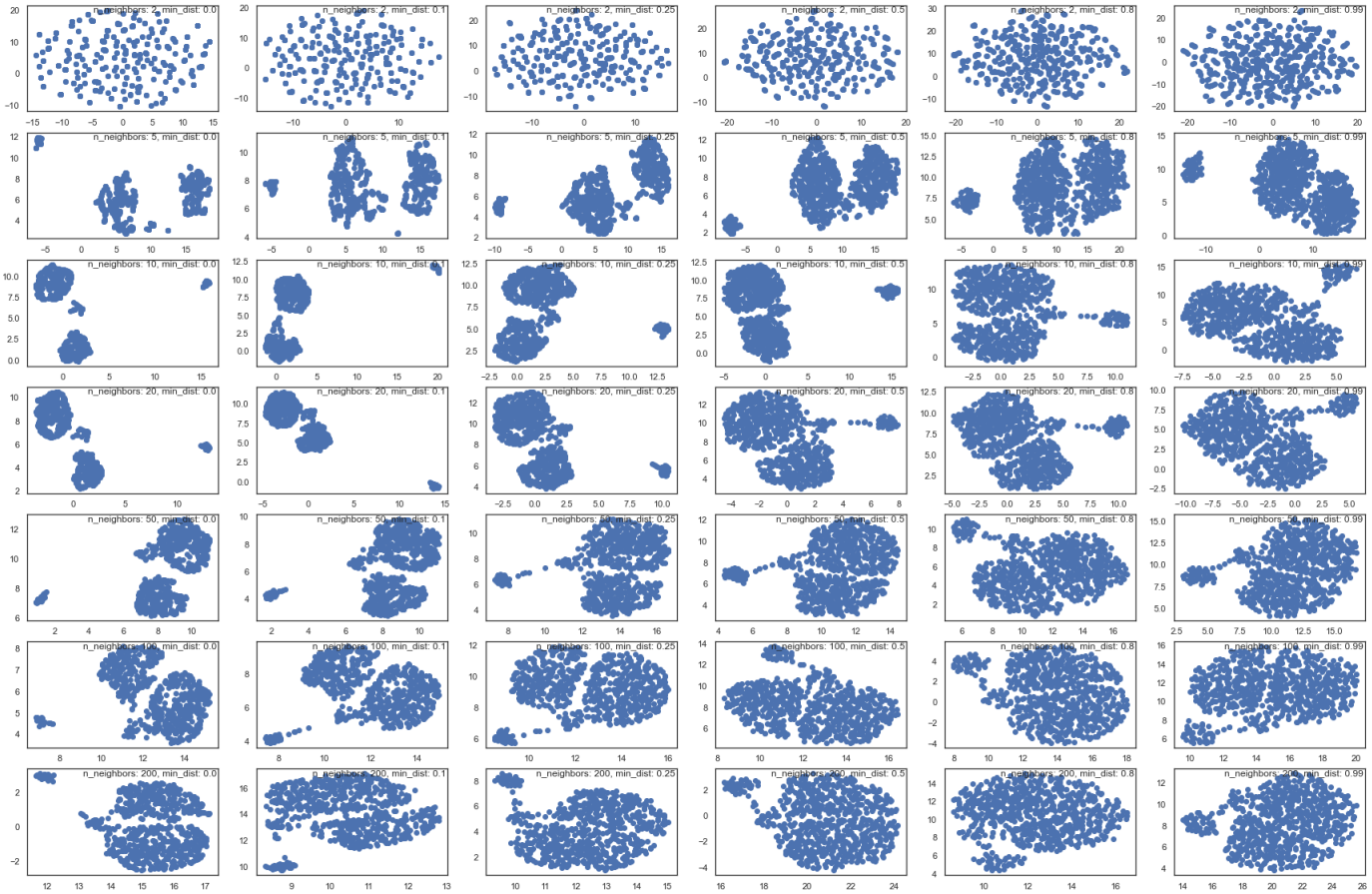


Figure C.1: A sweep through internal UMAP parameters n_neighbors and min_dist.

D Clustering Algorithms

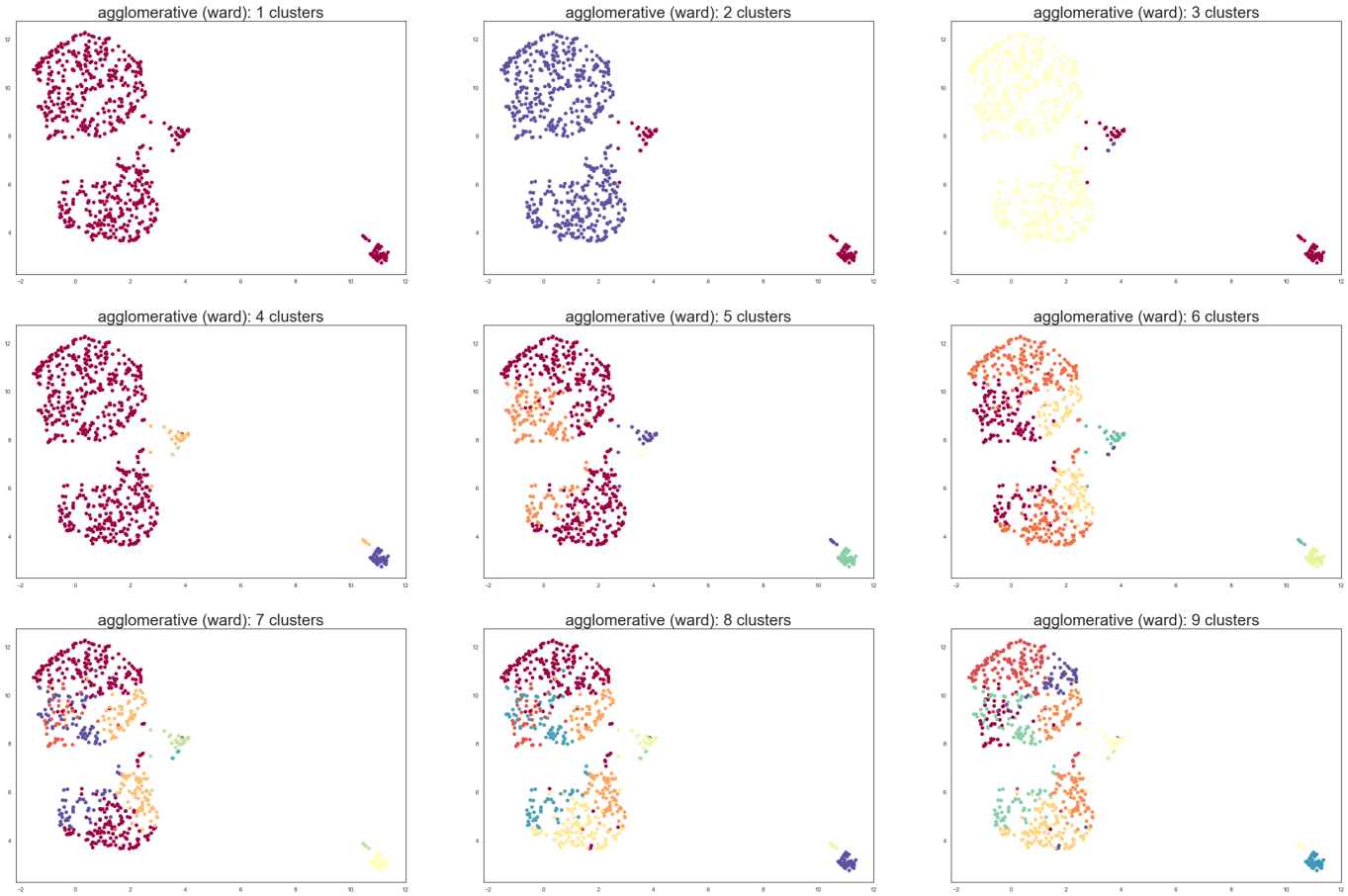


Figure D.1: Agglomerative clustering, ward linkage, with different values for k.

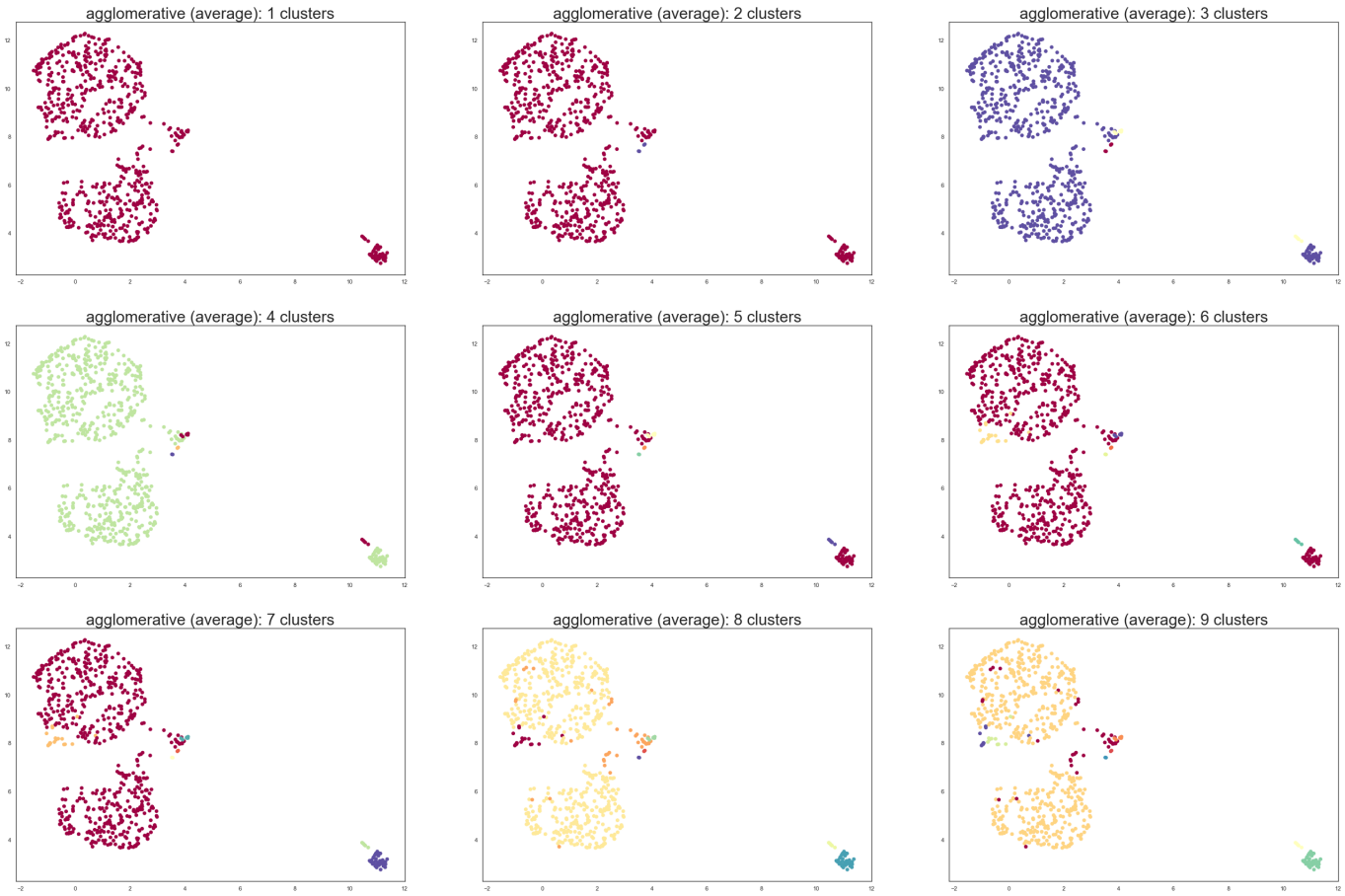


Figure D.2: Agglomerative clustering, average linkage, with different values for k.

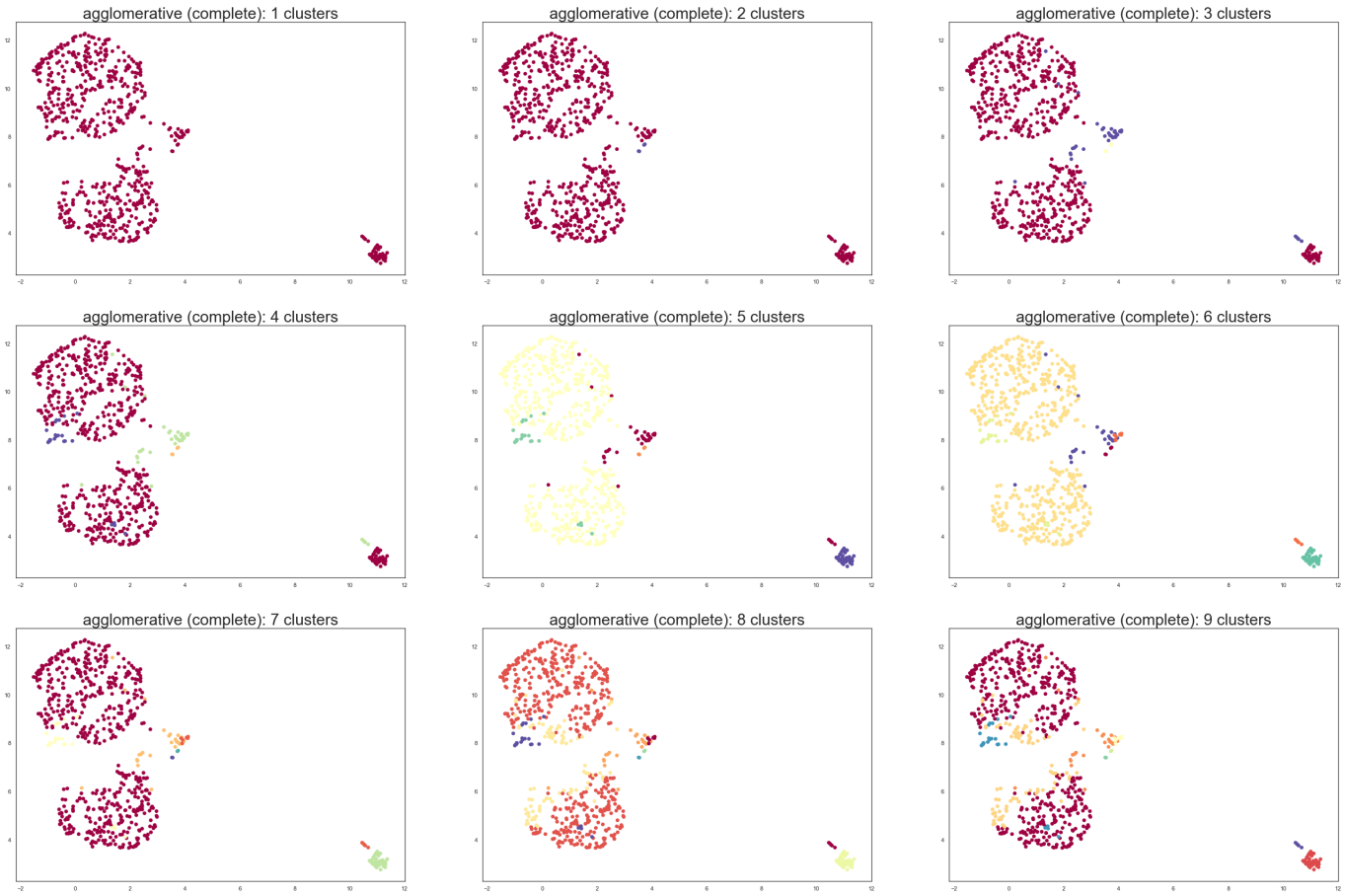


Figure D.3: Agglomerative clustering, complete linkage, with different values for k.

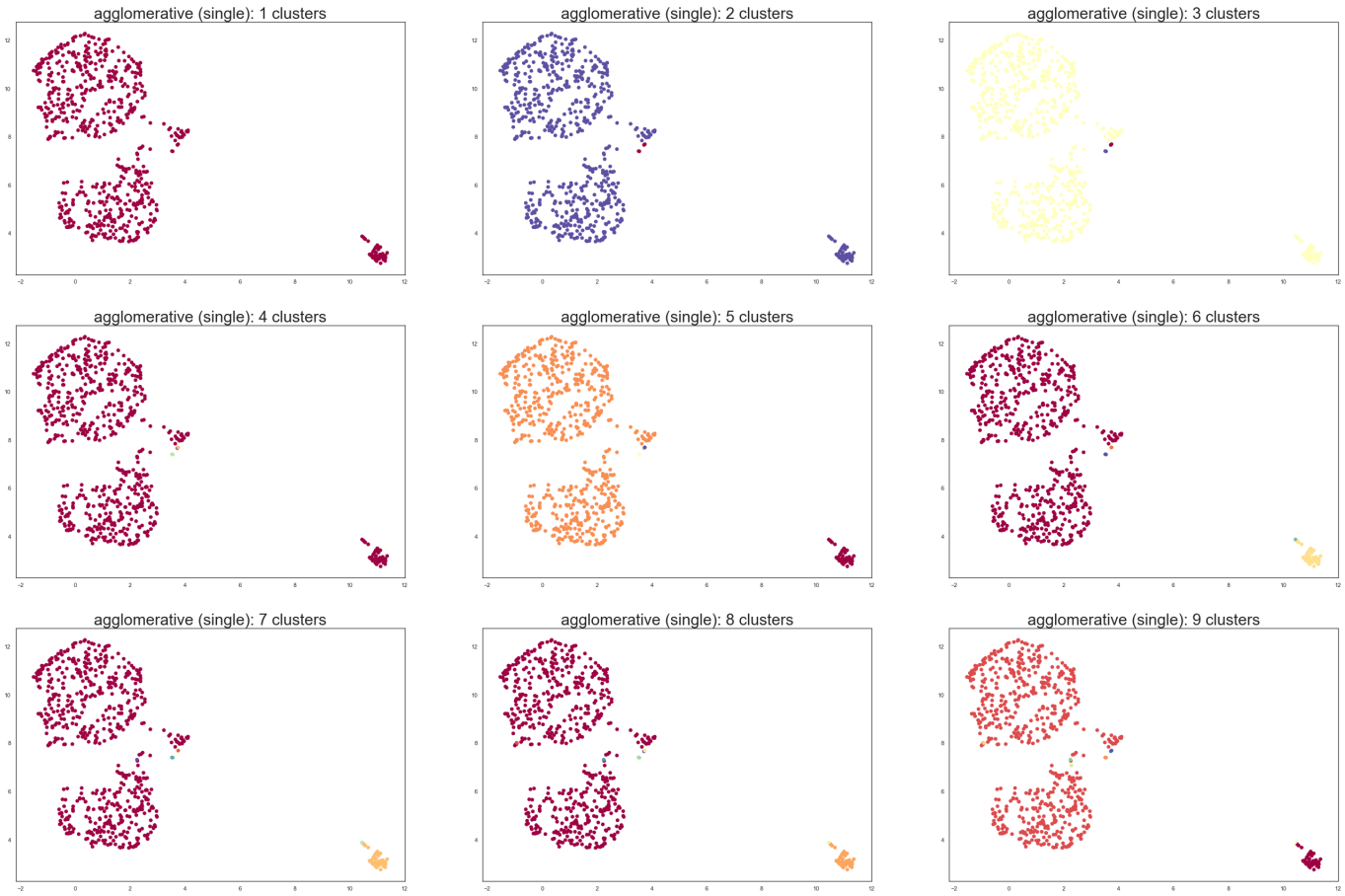


Figure D.4: Agglomerative clustering, single linkage, with different values for k.