Player Preference Extraction From In-Game Behavior

André Pais Borges de Macedo Leite
Instituto Superior Tecnico
University of Lisbon
Lisbon, Portugal
andreleite98@hotmail.com

***Abstract*—Procedural Content Generation is able to generate content tailored to players, but we need to discover the player's preferences to achieve that. This work aims to tackle how to extract and create a machine learning model of the player's preferences from gameplay collected data. To achieve this, a single-player offline game was created, where we placed carefully crafted challenges, based on six of the seven BrainHex classes studied, from which we removed Socializer as it does not fit our type of game, ensuring we matched the players preferences we were trying to measure.**

The player's gameplay data was extracted from the their interaction with challenges and the environment, and parsed to fit our machine learning needs. The parsed data was then used with a variety of machine-learning algorithms, such as Naive Bayes, Decision Trees, and K-Means to predict future players' gaming preferences.

The dataset was replicated six times, one for each BrainHex class, and separately used to train different machine learning models. Even with a very limited sample size of 30, (n=24 for the training set and n=6 for the validation set), our models reported a high accuracy in identifying the BrainHex classes of the players for five of the six datsets. The highest accuracy for each dataset in validation was: 100% for Conqueror, 100% for Achiever, 100% for Mastermind, 83.33% for Survivor, and 66.66% for Daredevil.

***Index Terms*—Player Models; Personality; Machine learning; Data Mining.**

I. Introduction

A. Motivation

With the games industry's constant growth [1], developers are always searching for new ways to make their games stand out in such a crowded environment and appeal to the highest number of potential players.

Choosing what type of game to make can become a problem if the genre ends up being unpopular, giving less prospects of turning a profit. This reduces the likelihood of certain types of game being made, since a player may see the game's genre and automatically assume they won't like it. To solve this problem, we can try and capture the widest range of preferences possible, by making a game that adapts to each player, considerably increasing the pool of player types the game targets. There have been various solutions presented by developers throughout history, with most of them relying on psychological theories to adapt to the player. Using personality models to adapt a game to the player has been shown as a working solution to the problem [1], but multiple people and organizations have tried to find a better way of expressing player's preferences, this being how player type models were born. The development of player type models sparked some research on how to adapt a game to a given player's preferences; however, on the other hand, not much research has been made on how to figure out the player's preferences inside the game environment. This is a crucial step because asking an individual to answer a questionnaire before letting them start playing a game can be not only very intrusive and considered a hassle, but it could also make them give up on playing the game. Another point to consider is that the research which tackles this specific problem is focused on simpler player satisfaction models, which can be too general to use in adapting to a given player's preferences.

Our work will then focus on tackling this idea to extrapolate players' playstyle preferences from data gathered in-game.

B. Problem

The problem we are trying to tackle is how to extract the players' preferences to generate tailored content. In the past, some games have resorted to straight out questioning the player about their likings [2] [3], which is a very invasive approach, while others tried to understand how the players behave by their actions or inactions in-game by analyzing game telemetry data.

The problem is then how to extract the players' preferences from their in-game behavior.

C. Hypothesis

In this work, we propose a methodology for collecting and processing game data to extrapolate the player's preferences for later use in adaptive content. It will explore the hypothesis that we can derive the player's preferences by gathering data from meaningful game data-points, like the player's options and decisions and how they execute them.

[1]https://www.statista.com/statistics/292056/video-game-market-value-worldwide/

[2]Silent Hill: Shattered Memories https://www.konami.com/games/eu/en/products/shsm/
[3]Until Dawn https://www.supermassivegames.com/games/until-dawn

II. Related Work

A. Personality Models

Personality theories/models are taxonomies that try to classify people by the way they interact with and act upon the world. With every person being unique in some way or another, these models categorize people's behavior using one or more categories. We researched two theories, the Myers & Briggs' Type Indicator (MBTI) [2] and the Five Factor Model (FFM) [3], which have been the subject of numerous studies involving games, and many player models have been created based on them.

MBTI [2] is an introspective, self-report questionnaire which classifies individuals according to four psychological preferences relating to how they perceive the world around them: extraversion-introversion, sensing-intuition, thinking-feeling, and judging-perceiving.

FFM [3] also known as the Big Five personality traits, was formed by applying factor analysis to several independent sets of surveys on personality data. This analysis revealed similarities between several different verbal descriptions of personality traits, combining them into five main factors, openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism.

B. Player Models

Player's preferences are reflected by their actions in-game and by the choices of games and content they engage with. These traits are what player type models try to categorize and explain. Some of these player models are based upon the personality models discussed above (BrainHex [4] and Quantic Foundry's Gamer Motivation Profile (GMP) [5]), while others are based on direct observation of player behavior (Bartle taxonomy of player types [6]), and lastly the Marczewski's Player and User Types Hexad [7] which is based on research about human motivation as well as another player model (Bartle taxonomy of player types [6]).

Richard Bartle developed Bartle Player Types [6], one of the first models to classify players according to their preferred gameplay actions. A study conducted to analyze the players' behavior in a MUD (Multi-user Dungeon) created four categories: Achievers, Killers, Explorers, and Socializers. Two main axes of interest, Players-World, and Acting- Interacting, derive the four different categories

GMP [5], initially developed in 2015 by Nick Yee and Nicolas Ducheneaut, categorizes players using twelve motivations inspired by other works like the FFM, grouped in pairs by factor analysis [5]. The twelve motivations identified were: Fantasy, Story, Design, Discovery, Destruction, Excitement, Competition, Community, Challenge, Strategy, Completion and Power.

Marczewski's Player and User Types Hexad [7] is a player and user type model directed to gamification systems. The model talks about six different types of users, Achiever, Socializer, Philanthropist, Free Spirit, Player, and Disruptor, with two of them, Player and Disruptor, labeled as not having very concrete motivations. For the other four the motivations identified were: Mastery for Achiever, Relatedness for Socializer, Purpose and Meaning for Philanthropist, Autonomy and self-expression for Free Spirit, Rewards for player and Change for Disruptor.

BrainHex [4] is a satisfactory player model created by *International Hobo Ltd*, based on studying neurobiological research papers and directly influenced by the Demographic Game Design (DGD)1 survey results (which resulted in the DGD1 model) and the DGD2 survey [8]. The BrainHex model uses seven archetypes, where each one links to a key element in the human nervous system, to define players' motivation and behavior in-game. The model also defines exceptions as the opposites of each class, referring to what the player dislikes the most. The seven classes identified in the BrainHex model are: Achiever, Conqueror, Daredevil, Mastermind, Seeker, Socializer and Survivor. Upon taking the survey, each person will be assigned a score, from a scale of -10 to 20 to each class following their expressed preferences. This means that a given player is not only defined by their main class (highest score on the survey), subclass (second-highest score on the survey), and/or exception (negative score in a given category) but by the score they obtained in each category.

C. Machine Learning Algorithms

Our machine learning process started with the use of the software Waikato Environment for Knowledge Analysis (WEKA) [4] [9], which is a machine learning tool developed at the University of Waikato in New Zealand, including a library of machine learning algorithms and a way to visualize the input data and the results obtained applying the learning algorithms. We separated our data-sets into 70% for training and 30% for testing, and made use of cross-validation to measure the performance of our models in the training phase.

Decision tree is a machine learning algorithm that creates a tree-like structure with nodes and leaves. Each node represents a conditional control statement where a given attribute is tested, usually comparing its value against constant. Each leaf represents the classification of one instance, a set of them, or a probabilistic distribution. Finally, branches signify the connection between different nodes or nodes and leaves of the tree [9].

Instance-based learning is characterized by the memorization of the data rather than learned concepts. To classify a query, we look at the database and compare the input to previously labeled data [9].K-Nearest Neighbors (KNN) classifies the object weighting each K neighbors according to the distance from the object. In a more simplistic form, we can take the most common value among the K nearest examples.

A cluster is a set of data objects where each object is similar to others in the same cluster and dissimilar to objects in different clusters. Clustering is an unsupervised classification algorithm by which we arrange the objects in clusters. In the K-Means algorithm, we start with a training set composed of

[4]https://www.cs.waikato.ac.nz/ml/weka/

N samples, and our goal is, given a value of K, partition the data into a K number of clusters. We end up with K centroids (cluster centers) where each point in a given cluster is always closer to its cluster centroid than to another cluster's centroid. Each centroid is represented by the mean value of all points contained in the given cluster [9].

Naive Bayes classifier is based on Bayes' theorem and requiring substantial attribute independence, is one of the most practical learning methods available. It works by taking as an input a feature vector X to predict the corresponding class Y. This means that given a data point $X = (x1, x2, ..., xn)$, we want to figure out the odds of the class Y being y [9].

III. Methodology

A. The Game Structure

We developed our game, "Fig. 1", on top of the Unity Store asset *"TopDown Engine"* [5], which helped with more low-level systems such as level and character management, and visual assets. The game structure can be divided into two main parts, the "Inside" section and the "Outside" section.

Fig. 1. Look and feel of the testbed game, as seen by the player.

a) "Inside": The "Inside" section corresponds to six different isolated areas, each made in the image of one of the six BrainHex classes approached in this work. The entrances to these locations are present in the "Outside" section, where an Non Playable Character (NPC) will introduce the player to the type of challenge they will face if they choose to enter the correspondent "Inside" section, via a small dialogue. Of the six types represented, two have challenges that require the player to either go to the "Outside" sections to complete (Achiever), or need the player to perform certain actions in the "Outside" section to unlock (Seeker). This happens since both classes are better represented in open type challenges, which measure the players' actions over a big period of time, rather than closed-off type challenges, which are focused solely on the moment to moment interactions.

b) "Outside": The "Outside" section is made of several different "Levels", divided into multiple parts, the "Intersections", the "Paths", the "Level Start" and the "Level End". There might also be some hidden areas that branch of different parts of the "Outside" parts, and some quests for the player to engage with, although these are only present in some "Outside" "Levels". As we can see in "Fig. 2", each "Outside" "Level" starts with a "Level Start" part, which guides the player to the "Intersections". Each "Intersection" has 3 different paths that the player can choose from, each made in the image of one of four BrainHex classes, Daredevil, Conqueror, Survivor and Mastermind, since these four types are easily represented in close-off paths, like discussed previously for the "Inside" challenges. In order to properly inform the player, an NPC is present at each "Intersection", which gives the player a brief description of the 3 different paths they can choose from, based on their corresponding BrainHex class's description. The other two types, Achiever and Seeker, are represented as hidden areas, special quests and other player metrics such as coins and pots collected and secrets found.

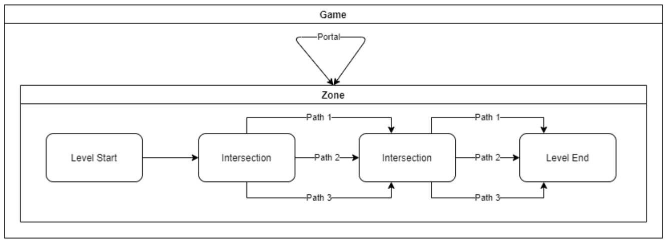


Fig. 2. The structure of the "outside" section of the game.

It is also important to note that in the first "Intersections" of the game, the player is expected to still be exploring and experiencing the content and mechanics of the game for the first time, which means they are less likely to be choosing content strongly aligned with their preferences. This should to be taken into account both while designing the game and when building the predictive models.

B. Player Freedom

The main objective of this work is to see if it is possible to measure the player's preferences through the behaviour they express while playing the game. In order to achieve this it is of the utmost importance to give the player the freedom to engage with whichever elements of the game they want to. This means only forcing the player to complete one path per intersection, with all other elements being completely optional. his means the player, apart from completing one path per intersection can choose to:

- Give up mid-way through any path, go back to the intersection, and start a different path.
- Complete more than one path from each intersection (even all of them).
- Engage with zero Challenges ("Inside" sections) if they want to.
- Enter a Challenge and give up mid-way through it.
- Complete a given Challenge more than once.
- Explore the entire map, or stick solely to intersections' paths.

[5]https://topdown-engine.moremountains.com/

- Collect every item they find, or not collect anything at all.

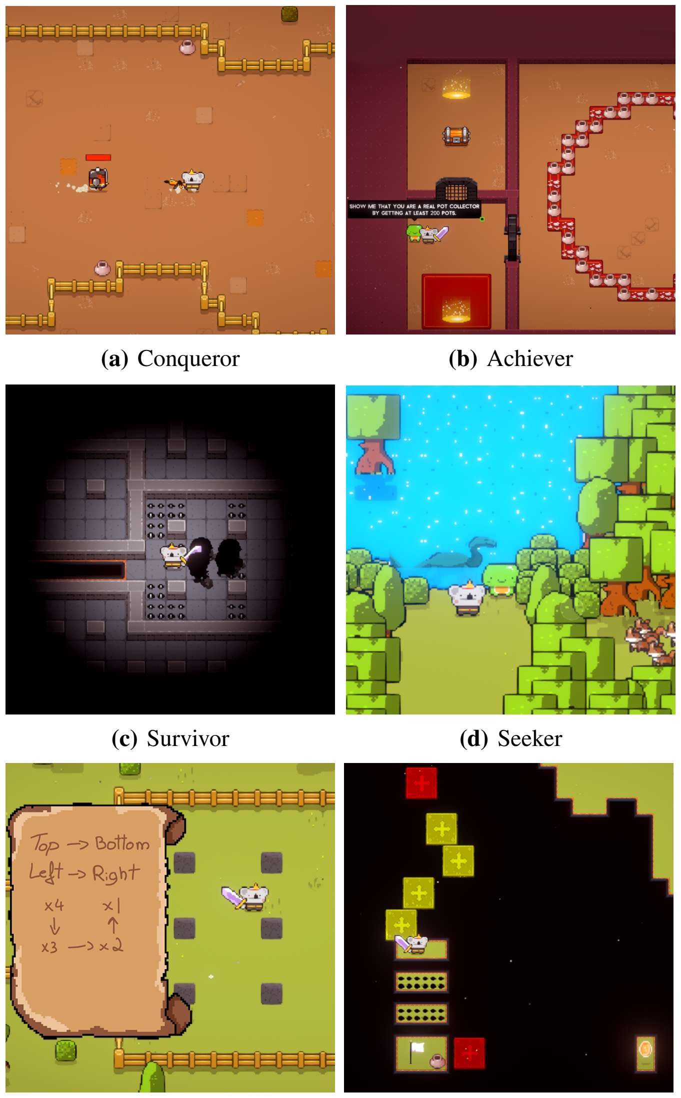
This game design allows the player to explore and engage with the game however they feel more comfortable doing so, hopefully expressing their gaming preferences in line with the BrainHex player model.

C. BrainHex Classes In The Game

In order to properly design paths and challenges representing each BrainHex class, we looked to the official BrainHex website[6], and took into account all the information provided for each BrainHex class, such as what someone who identifies as the given class likes, how they usually behave, their favorite types of games, and their class's relation to other player and personality models. As such, we came up with the following overall descriptions of what should be included in the different types of challenges, paths, and metrics according to their respective BrainHex class:

- Daredevil [7] - The player will need to overcome a challenge filled with moving platforms, trapdoors, and be very precise with their timing.
- Conqueror [8] - The player will need to overcome a series of difficult enemies with increasing difficulty.
- Mastermind [9] - The player will need to use limited resources to solve a puzzle with moving objects and pressure plates, among other things.
- Seeker [10] - The player will need to search for a key to a hidden door that can only be obtained by thoroughly searching the level. The player will find strange and wonderful scenarios.
- Survivor [11] - The player will need to survive trapdoors, spikes, monsters, and other elements that may be deemed as "scary".
- Achiever [12] - The player will try to do everything available to them in the game.

As example of what type of content was designed for each of the BrainHex classes can be seen in "Fig. 3".


SHOW ME THAT YOU ARE A REAL POT COLLECTOR BY GETTING AT LEAST 200 POTS.
Top -> Bottom
Left -> Right
x4
x1
x3 -> x2

(a) Conqueror **(b)** Achiever

(c) Survivor **(d)** Seeker

(e) Mastermind **(f)** Daredevil

Fig. 3. Six different examples of what type of content was designed for each of the BrainHex classes.

D. Data Collection

a) "The Data": In order to perform data analysis and apply machine learning techniques we first need to collect relevant data from the players' in-game behaviour. Since we cannot be sure of what might actually end up being relevant data to the experiment, we decided to log every relevant action the player performs inside the game. This allows us to, if needed, reconstruct the entire play session from the logged game data. We decided to draw a line on what is considered relevant in order to not overload our logged data with irrelevant information. This meant that key presses, mouse movements, or game-pad actions were not recorded, however, every action the player character performs in-game is registered. With this in mind, we recorded the following actions taken by the player character:

- Picked up items, such as coins, health packs, and guns, among others.
- Defeated enemies, such as simple ninjas, bosses, and ghosts.
- Areas entered and exited from, such as intersections, paths, hidden places, and zones, among others.
- Deaths, with where and how they happened.
- Checkpoints reached, and respawn locations triggered.
- Quests started and completed.
- Objects interacted with, such as levers and chests.

On top of all the information described above, we also needed to acquire information on how much the player pro-

[6]https://blog.brainhex.com/
[7]https://youtu.be/Erdug9eO-K4
[8]https://youtu.be/IVHTWJKYtIU
[9]https://youtu.be/lGxheUwJWjs
[10]https://youtu.be/XCHGVLS8q0I
[11]https://youtu.be/mYy47ONc4Eg
[12]https://youtu.be/GYxu0mKxyjo

gressed into a certain path of challenge if they didn't finish it. This information is needed so that, if the player gives up mid-way through a path or challenge, we can score how much they engaged with it. In order to do this, we designed a system where we could use the information from the areas entered and exited, as described above, to initiate a separate logging system. In order to know how much the player engaged with a given path or challenge, we made use of the information in the list above to create an event system, which would progress as the player reached predefined parts of the challenge/path, defeated a certain number of enemies, or collected specific items, adding to their score. Although this system required us to specify every possible action that could contribute to the player gaining score in the challenge/path, it also provided fine-grained control over how the player is scored.

All actions were timestamped, which allows us to know, if we need it, how long the player took to complete any action, quest, or challenge. This metric was discarded later on, since we were made aware by several testers that since the experiment could take more than one hour, some of them decided to take breaks in between, which might change the result. This meant we didn't have concrete mechanisms to do this in an absolutely rigorous way, which could be tackled in future works. This leaves us with a log file composed of separate lines, where each one corresponds to one action taken by the player character, timestamped with the number of seconds elapsed since the game was started.

b) "Data Retrieval": In order to gather the logged gameplay data remotely from the user, a system to send it back to us was needed. We chose to create a system to automatically send an email with the log file as an attachment to an email generated for this specific purpose. This means that upon triggering the end screen, in the background, an email is sent with the attached log file and the corresponding Unique Identifier (UID) given by the tester. The tester is informed about this, happening, before, its occurrence, and can opt-out of the experiment if they choose to do so. Upon receiving the log file with the data, it was matched with BrainHex results, and the final questionnaire answers using the UID.

IV. Evalutation

A. Manipulation Check

To acquire the data needed to verify our design for the challenges we used a questionnaire format, where the user needs to rate each of the six videos in relation to six provided sentences. Each video resembles one of the six different game design philosophies we employed while making the game, described by each inside challenge we created for the different BrainHex classes. The questions were presented in a table, with the rating as columns and the sentences to describe each of the six BrainHex classes as rows. The table was displayed after each video for the user to fill out. The rating for each challenge ranged from "Strongly Disagree" to "Strongly Agree", with the following options in between: "Disagree", "Slightly Disagree", "Neutral", "Slightly Agree", and "Agree". The BrainHex sentences were taken from the official website and represent what each player identified as belonging to a specific class likes to do.

- Daredevil - "You like negotiating dizzying platforms or rushing around at high speed while you are still in control."
- Conqueror - "You like defeating impossibly difficult foes, struggling until you eventually achieve victory."
- Mastermind - "You like solving puzzles and devising strategies."
- Seeker - "You like finding strange and wonderful things, or finding familiar things."
- Survivor - "You like escaping from hideous and scary threats, pulse-pounding risks."
- Achiever - "You like collecting anything you can collect, and doing everything you possibly can."

Since we allow the users to rate each sentence individually in accordance to how much they thought it related to the video shown, the need aroused to come up with a way of validating the challenge with the design intent in mind. With the fact that the challenges presented were all created on top of the same, 2D run and gun base game, we expected some overlap in identifying the correct BrainHex class. With this in mind, we settled for accepting a positive answer if the user's score for the correct class' sentence is greater than the score for all other sentences. As an example if for the video of the Mastermind challenge, the user rates the Mastermind sentence with the highest score of all presented options, then we accept that answer as positive. On top of this, we also observed if the score for the correct sentence was at least "Slightly Agree", to make sure the user didn't just rate the challenge as low for all categories, which is also not a desirable outcome.

The results also showed, as we expected, that most challenges also had some connection to more BrainHex classes other than the main one we were trying to represent. This is attributed to the fact that challenges in a somewhat complex game can't really be created based on a single unblended, isolated class, as there will always be some residual components that can be associated with other BrainHex classes. We argue that as long as the main BrainHex class is identified as the single most important component of the challenge, then it can be used as a challenge for the given class in our game. These may, later on, also help in identifying oddities with our models or data.

With a game scope as large as ours it was not feasible to make pass through the manipulation check procedure all the challenges we designed. With this in mind, and with the main challenges of the game already validated, we decided to create the rest of the mini-quests, hidden areas, and paths, based on these validated challenges and the overall ideas on which we based them.

Although the results show us that, on average, participants correctly identified the BrainHex class, the individual results also showed us that everyone attributed the highest score for each video to the correct class. There was, however, one exception with one participant incorrectly scoring all challenges. We considered this result an outlier, since there were clear indications that the questionnaire had been answered randomly.

B. Final Experiment

a) Demographic: The demographic of the experiment's testers was mostly from people with a gaming background. The link to the google forms questionnaire which started the experiment was distributed among students of Instituto Superior Técnico (IST) via convenience sampling, and other people contacted personally. In total, we got 30 users participating in the experiment, with a 100% completion rate.

Our experiment collected data from 30 individuals, from which 29.0% were female and 71.0% were male, with ages ranging from 18 to 29 years old, with a mean of 23.03 and a standard deviation of 2.30.

As we can see, our coverage of the BrainHex space is pretty diverse, with only Survivor having a slightly more unsymmetrical representation. This helps in the differentiation of the data points, since if we only had one or two data points representing a given class, our results would have been severely skewed.

b) Data Treatment: The data gathered comes in a scattered format and needs to be properly organized to be used in WEKA. This process is performed by an automated script, which outputs the dataset in the form of a Comma-Separated Values (CSV) file format by looking at all entries in the log file and categorizing them, extracting information on what the player did and what they didn't do, and, if needed, attributing a score to their engagement with a given section of the game. The categorization takes the individual events and organizes them into two different types:

- Area events, which record if a given area was reached by the player, and how long they stayed there.
- Instantaneous events, like the player's deaths, kills, picked up items, among others. These are associated with area events.

Afterwards, the script parses through the collected information to attribute a score to the events above and categorizes them in the following variable types:

- Quest completion rates. If the player accepted a quest, and if they completed it. In case the quest was accepted but not completed, a score is attributed based on how far along the player reached into the quest. This is done based on predefined metrics, which we will discuss later in this section.
- Challenges completion rates. Like quest completion, they mark if the challenge was completed or not, and in case it was started but not completed a score is also given to the players engagement.
- Path completion rates. Like the other two completion rates, this tell us if the player started a path or not, and if they gave up mid-way a score is given for how far along they were able to reach.
- Number of coins collected.
- Number of suits of armor purchased.
- Number of pots collected.
- Number of hidden areas found.
- Enemies defeated.

Some examples of metrics used to attribute a score to each of the BrainHex classes' challenges, paths or quests are:

- Seeker Challenge: This challenge requires the player to search for hidden coins throughout one Level, delivering them to a NPC and experiencing a small scenario with a fantastical creature. The progression is first measured by how many coins the player managed to find, with each coin having a separate value depending on how hard they are to discover, capping at 0.45 out of 1. The remaining 0.55 of the score is distributed by the delivering of the coins to the NPC, and the experiencing of the wonderful scenario.
- Conqueror Challenge: This challenge requires the player to fight 3 waves of enemies inside a closed off arena, with a final Boss at the end. The score is distributed by how many enemies the player manages to defeat, with a bonus for finishing each wave.
- Achiever Challenge: This challenge requires the player to find and collect 200 pots throughout the entire game. The score is distributed to the player in relation to how many pots they collected up to a maximum of 0.9. The final 0.1 of the score is only given after they deliver the pots to the NPC.
- Survivor Challenge: This challenge requires the player to find their way through a small cavern filled with traps, ghost enemies they cannot fight against, with a limited range of vision and while listening to a creepy song playing in the background. The score is attributed to the player based on how far along the cavern they managed to get. As with many other challenges, if the player fails and tries again, without ever succeeding, they will receive a small bonus points if their total score does not surpass 0.9.
- Daredevil Challenge: This challenge requires the player to jump across moving platforms, time their movement with the spikes that come up from the floor, while trying to avoid falling out. This requires precise movement and for the player to keep some speed to not fall behind. The score is attributed based on how far along the path they managed to get.
- Mastermind Challenge: The mastermind challenges all consist of different puzzles with different mechanics, which means they all have different scoring systems. If the path has multiple puzzles, each puzzle will be graded separately and the combined score is always equal to 1. Some puzzles do not have a way to measure the player's progress so they either award the full score, or zero.

With the data parsed to a format usable by WEKA, we switched our focus to how we should handle the data for training our models. Taking into account that our dataset has a small number of samples (n=30), if we were to randomly split the dataset into training and testing sets, we might run into a problem of having all "positive" results fall into one of the sets, while the other ends up with none, or vice-versa. This means that we need to make sure both the training and testing sets have the same proportion of positive-negative instances, however another problems arises, which is how to define a positive or a negative instance. Regarding the dataset we decided to make six copies of it, one for each BrainHex class, and split the data into 70% for the training set and 30% for the testing set With this method we can treat each BrainHex class separately, and independently, which lets us have finer control over how we classify the dataset. We can classify each instance as belonging to the corresponding BrainHex class or not, instead of overlapping the classification of all classes. This would also make it easier for evenly splitting the data for the training and testing sets. With this in mind, each copy was treated as a different dataset, and we settled for a cutout of 10 for the positive and negative values. This means that, for example, for the Mastermind copy of the dataset, we looked at the BrainHex scores of each instance and attributed a positive class (belongs to the BrainHex class), denominated by the value "1", if the instance had a BrainHex score higher or equal to 10, and a negative class (does not belong to the BrainHex class), denominated by the value "0", otherwise. The cut-off value of 10 was chosen since it marks the middle point in the BrainHex scale for someone who likes the given BrainHex class. The BrainHex scores vary between [-10,20], with values in the range [-10,0[corresponding to the player having the given class as an exception. We are then left with the value range [0,20] for how much the player likes the challenge and we chose the value 10 as it marks the middle point of how much someone enjoys that specific type of content. The distribution of these results can be seen in "Table. I".

TABLE I
BRAINHEX SCORE COUNTS FOR ALL PARTICIPANT. POSITIVE VALUES INDICATE A SCORE GREATER THAN OR EQUAL TO 10, WHILE NEGATIVE VALUES INDICATE A SCORE LOWER THAN 10.

	Seeker	Survivor	Conqueror	Daredevil	Mastermind	Achiever
Positive	16	22	16	17	16	18
Negative	14	8	14	13	14	12

c) Results: With the six training datasets, we moved onto the training of our models, using 10-fold cross validation. We used five different machine learning algorithms available in our software of choice WEKA, RepTree, RandomTree, RandomForest, NaiveBayes and KMeans. The evaluation for the KMeans algorithm was done via the classes to cluster feature, where WEKA first ignores the class and builds the clusters. It then allocates classes to the clusters during the test phase, depending on the majority value of the class attribute inside each cluster. Finally, depending on this assignment, it computes the classification error which we subtract from 100% to obtain the accuracy, this being the value displayed in the tables below.

The results obtained from the first training of our models can be seen in "Table. II". We then performed feature selection, based on both the results obtained from WEKAs tools for feature selection and our initial understanding, from designing the game, of which features should correlate to which classes. We also discretized the data for Naive Bayes [10], and ran the algorithms again with the modified datasets. The results from the modified datasets can be seen in "Table. III".

TABLE II
MODEL TRAINING RESULTS FOR THE FIVE MACHINE LEARNING ALGORITHMS WITH 10-FOLD CROSS VALIDATION ON THE RAW DATASETS.

Datasets	**RepTree**	**RandomTree**	**RandomForest**	**NaiveBayes**	**KMeans**
Conqueror	62.50%	83.33%	83.33%	75.00%	50.00%
Achiever	66.66%	75.00%	79.16%	75.00%	58.34%
Mastermind	75.00%	75.00%	79.16%	70.83%	66.66%
Survivor	87.50%	79.16%	79.16%	87.50%	62.50%
Seeker	100%	95.83%	100%	91.66%	66.66%
Daredevil	58.33%	58.33%	66.66%	62.5%	62.5%

TABLE III
MODEL TRAINING RESULTS FOR THE FIVE MACHINE LEARNING ALGORITHMS WITH 10-FOLD CROSS VALIDATION ON THE MODIFIED DATASETS.

Datasets	**RepTree**	**RandomTree**	**RandomForest**	**NaiveBayes**	**KMeans**
Conqueror	75.00%	87.50%	79.16%	79.16%	79.16%
Achiever	70.83%	79.16%	83.33	79.16%	70.83%
Mastermind	83.33%	79.16%	75.00%	79.16%	66.66%
Survivor	87.50%	79.16%	87.50%	91.66%	70.83%
Seeker	100%	100%	100%	95.83%	91.67%
Daredevil	75.00%	75.00%	70.83%	75.00%	66.66%

As we can observe form the results obtained in "Table. II" and "Table. III", some classes were overall better identified than others, with Daredevil sticking out as the worst result. We can also see that the Random Forest algorithm, for some classes, performed better in the original raw datasets, in comparison to the modified datasets. With this in mind, we can conclude that either the removed elements were in some way relevant to the identification of the given class by the Random Forest algorithm, or that with our small sample size, some features were being incorrectly used and our model overfitted the data by picking up some noise.

The best models were recorded for each algorithm of each dataset, and used in the final validation performed with the remaining 30% of the data (testing/validation dataset). These results can be seen in "Table. IV".

As we can observe in the table, five out of the six classes had very high levels of accuracy, greater than or equal to 83.33%, which means at least five out of the six instances in the validation dataset were correctly identified. On the other hand, the Daredevil class had very low levels of accuracy which might mean our choice of metrics and challenges for the class were not appropriate. Since WEKA doesn't support test set evaluations for KMeans clustering we are unable to provide validation results for this algorithm.

Regarding the classes with high success rates, Seeker performed especially well both under training and validation.

TABLE IV
MODEL VALIDATION RESULTS FOR THE SIX DIFFERENT DATASETS USING THE BEST PERFORMING MODELS FOR EACH ALGORITHM OF EACH DATASET.

Datasets	RepTree	RandomTree	RandomForest	NaiveBayes	KMeans
Conqueror	100%	100%	83.33%	83.33%	–%
Achiever	83.33%	83.33%	83.33%	100%	–%
Mastermind	83.33%	83.33%	83.33%	100%	–%
Survivor	83.33%	83.33%	83.33%	83.33%	–%
Seeker	100%	100%	100%	83.33%	–%
Daredevil	50%	50%	66.66%	66.66%	–%

Upon closer inspection we could observe that the attributes “seeker_challenge” and “seeker_quest” proved to be excellent indicators of Seeker behavior, as both had clear cut-off values, which perfectly split the dataset, identifying the correct class. Although our dataset had a small number of samples (n=30, with n=24 for the testing set and n=6 for validation), and these cut-offs might not have worked perfectly with a higher number of samples, we can conclude these attributes are very good indicators for identifying the class in question. As for the other three classes with high scores, we can see that they all obtained around the same accuracy in both the training and validation phases, taking into account the low number of samples in our dataset, which restricts the number of possible steps of accuracy that can be obtained.

After finalizing the dataset validation we decided to create a list of the top indicators, and a possible explanation for some of them, for each BrainHex class based on the final Decision Tree models and the feature selection.

Conqueror:

- The four paths that came from the main intersections, which excludes the first path as it was in the exploration section of the game. ´
- The number of enemies defeated.
- The lack of engagement with the Seeker quest.

Survivor:

- The engagement with the Survivor challenge.
- The engagement with the last Survivor path. The first three Survivor paths after the exploration zone where neither good nor bad indicators.
- The non engagement with one of the mastermind paths.

Seeker:

- The engagement with the Seeker challenge.
- The engagement with the Seeker quest.
- The number of hidden zones visited.

Achiever:

- The engagement with the Achiever challenge.
- The number of pots and coins collected.
- The engagement with the Achiever armor collection challenge.
- The number of hidden zones reached. This metric is related to the hidden zones which where visible from the normal paths and the player had to find the entrance to. They had rewards that the player could see (such as chests, coins, or pots, among others), which incentivized the players to go collect them.

Daredevil:

- The Daredevil’s paths were mediocre indicators.
- The engagement with the Daredevil challenge was a mediocre indicator.

Mastermind:

- The engagement with the Mastermind challenge.
- The second and third Mastermind paths after the exploration zone. The other paths were not so good indicators.

V. CONCLUSIONS

From the results obtained in the Manipulation Check, we can conclude the participants correctly understood the design intentions of the challenges. All participants were able to correctly match the challenges presented in the videos to the six BrainHex classes, by scoring the corresponding class as the highest.

From the final experiment, we concluded that, for all BrainHex classes besides Daredevil, we were able to correctly identify the participants’ BrainHex type.

Upon observing the models created, like Decision Trees, and looking at their accuracy, we can conclude that the metrics selected to measure the classes: Conqueror, Survivor, Seeker, Mastermind and Achiever, were appropriate. On the other hand, the Daredevil class could have had more or better metrics used, such as the discarded “time to complete”.

These results corroborate our hypothesis that it is possible to extract the players’ self-reported preferences from their in-game behavior. This allowed for the creation of several machine learning models, using different algorithms, which could identify the players’ BrainHex class, from data collected of their in-game behavior, with a high degree of accuracy. This game could be used as a proxy to identify the dominant dimensions of a player by observing their playstyle, as opposed to having them answer a questionnaire, which would allow for its in-game integration.

A. Future Work

For future research, we suggest the replication of this study, first and foremost, with a focus on obtaining more participants. We also had other ideas on how to improve or expand our work such as:

- Experiment with the methods described here using different player type models, such as GMP, Marczewski’s Player and User Types Hexad or Bartle Taxonomy of Player Types, or even with personality models, such as the FFM or MBTI.
- Exploring the potential relation between different BrainHex classes. We saw this happen in one instance, where the non-interaction with a challenge from the Seeker BrainHex class proved to be an indicator for the player belonging to the Conqueror class.
- Better representation of different classes throughout the dataset. Some classes representation was not perfect, and

with a small dataset like ours, one outlier can more easily skew the results.
- The creation of the Procedural Content Generation (PCG) system for the generation of tailored content using the models created. This is the logical next step in the process of providing a truly tailored experience to the player without the need to fill any questionnaires before playing.

Furthermore, we advise extra care with the design of challenges and the metrics recorded from them, since they will have the greatest impact on the results obtained from the models. In this line of thought, we also advise careful consideration to the freedom given to the player, and to be mindful of player curiosity and exploration during the initial section of the game.

ACKNOWLEDGMENT

I would like to thank my dissertation supervisor, Prof. Carlos Martinho, for his insight, support, guidance, and sharing of knowledge that has made this work possible. Thank you to my entire family and my friends for their support during these difficult times. I would also like to give a special mention to Victor Ribeiro for his patience during our long discussions and finally a special thanks to my friends at Los Cotons for being there with me every day.

REFERENCES

[1] R. Dias and C. Martinho, “inflow: Adapting gameplay to player personality,” 2010.

[2] I. B. Myers and P. B. Myers, *Gifts differing: understanding personality type*. Consulting Psychologists Press, 1995.

[3] R. R. McCrae and O. P. John, “An introduction to the five-factor model and its applications,” *Journal of Personality*, vol. 60, no. 2, p. 175–215, Jun 1992. [Online]. Available: http://dx.doi.org/10.1111/j.1467-6494.1992.tb00970.x

[4] Intenational Hobo. (2008) BrainHex. [Online]. Available: https://blog.brainhex.com/

[5] N. Yee, “The gamer motivation profile: What we learned from 250,000 gamers,” 2016.

[6] R. Bartle, “Hearts, clubs, diamonds, spades: Players who suit muds,” 1996.

[7] A. Marczewski, *User Types HEXAD*, 2015.

[8] L. E. Nacke, C. Bateman, and R. L. Mandryk, “Brainhex: A neurobiological gamer typology survey,” *Entertainment Computing*, vol. 5, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1875952113000086

[9] Witten, I. H., Frank, E., Hall, M. A., &amp; Pal, C. J. (2017)., *Data mining: Practical machine learning tools and techniques. Amsterdam: Morgan Kaufmann.*

[10] G. H. John and P. Langley, “Estimating continuous distributions in bayesian classifiers,” in *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345.