FACULTY OF ENGINEERING
AND ARCHITECTURE

Applying Dynamic Game Difficulty Adjustment Using Deep Reinforcement Learning

Robin Lievrouw
Student number: 01412057

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 Computer Science Engineering

Academic year 2019-2020

GHENT
UNIVERSITY

Word of thanks

First and foremost I would like to thank my promotor prof. dr. ir. Francis Wyffels and counsellor Andreas Verleysen for allowing me to have the opportunity to write a thesis about the interesting field of dynamic difficulties in games. Their feedback and help was always available when necessary and was very insightful.
Secondly, I would like to thank the kind people of The Krook library of Ghent and Imec for allowing me the opportunity to set up two experiments in the library. Finally, I would like to thank my family and friends for supporting me during all these years of studying and helping me to successfully graduate.

Abstract

Gaming today has jumped by leaps and bounds in terms of graphical fidelity and gameplay mechanics. Yet one central aspect of games has remained stagnant, the difficulty. Not all people fit in the typical easy, medium and hard settings that are available in games today. Gamers can find themselves easily bored or frustrated with these traditional static difficulties. To combat this, dynamic game difficulty adjustment can be applied. In this way, the difficulty dynamically adapts itself to the individual playing, matching his skill level and gaming preferences. In this thesis, the focus lies on implementing a form of dynamic difficulty adjustment using deep reinforcement learning on the classic arcade game, Space Invaders. Several experiments are undertaken where the difficulty is adapted with increasing dynamic parameters.

Applying Dynamic Game Difficulty Adjustment Using Deep Reinforcement Learning

Robin Lievrouw

Promotor(s): prof. dr. ir. Francis wyffels & Andreas Verleysen

Abstract—**Gaming today has gone far in terms of graphical fidelity and gameplay mechanics, yet one central aspect remains stagnant, the game difficulty. Games today typically offer static difficulty levels and getting the difficulty right in games is challenging. Every individual has their preference and skill level and thus also a suited difficulty. Gamers can find themselves easily bored or frustrated within the provided difficulty options, as they do not fit their playstyle, skill level or preference of challenge. Dynamic difficulty adjustment attempts to solve this mismatch between the player and the game its difficulty. It does this by dynamically adapting the difficulty to fit the player his needs. In this work, a deep reinforcement learning technique, called Deep-Q-Learning, will be used to apply dynamic difficulty adjustment to the classic arcade game Space Invaders. The proposed method will be used to undertake several experiments with increasingly adaptable difficulties to match artificial players with their preferred difficulty.**

Keywords—**reinforcement learning - dynamic game difficulty - Deep-Q-Learning**

I. Introduction

CReating a game that people enjoy is no easy task. Many different aspects of games have a major impact on the enjoyment experienced. The gaming industry wants a high level of enjoyment to keep people playing and buying their games. Many different techniques and mechanisms like advanced storytelling, online leaderboards and level progression are implemented besides the core game to keep people playing. Elements of gamification like those previously mentioned are finding themselves into non-gaming contexts as well. Think of the level progression one can earn in Google Maps for reviewing a restaurant for example.

The difficulty of a game can also have a major impact on the enjoyment of players. Players can find themselves frustrated at games that are too hard and have a steep learning curve. On the other hand, players are also quickly bored when the game is too easy [3]. More specifically, games that have nothing new to learn offer the player less enjoyment[4]. While games today typically offer only a couple of static difficulty levels, gamers may find themselves in between difficulties with their skill level. A possible solution for this comes through dynamic difficulty adjustment or DDA. In DDA the difficulty of a game adapts itself to the player his skill level and preference of challenge. In this way, the developers of the game do not have to try to fit all players with certain difficulty levels, but rather let the difficulty adapt itself to each player keeping the player engaged.

In this work, dynamic difficulty adjustment will be applied to the game of Space Invaders, using deep reinforcement learning. First, a literature study is given about dynamic difficulty adjustment, player enjoyment and reinforcement learning. Secondly, a large data collection is done to give insight into how player performance relates to the player enjoyment. Afterwards, several experiments with increasing difficulty adaptability are conducted, using artificial players as test subjects. Finally, results are analysed and validated by comparing artificial player preference for certain difficulties with which difficulties get picked for them by the DDA system.

II. Literature

A. Dynamic Difficulty Adjustment

Dynamic difficulty adjustment is a method used to adapt a game its difficulty to match the player, meaning his skill level and preference of challenge. It is a subset of the more wide challenge tailoring where games are adaptable in more ways than the difficulty to suit the player. Typically DDA works by approaching the player his enjoyment and performance by a model and it adjusts the difficulty in an informed way based upon this model. Two categories exist in DDA, active and passive. The first has the player himself choose the difficulty of a game through indirect choices. The other works automatically, trying to remain unnoticed by the player[8]. The ultimate goal then of DDA is to keep the player in the flow zone. Meaning to keep a player engaged and challenged enough such that he or she feels completely immersed inside the game and loses track of time while playing. This concept is explained more in-depth in section II-B. In figure 1 an example is given of a player not fitting inside any of the predefined static difficulty levels typically given in games. The difficulty levels do not suit the needed balance between the player his skill level and need for being challenged and thus cause the player to not enter the flow zone.

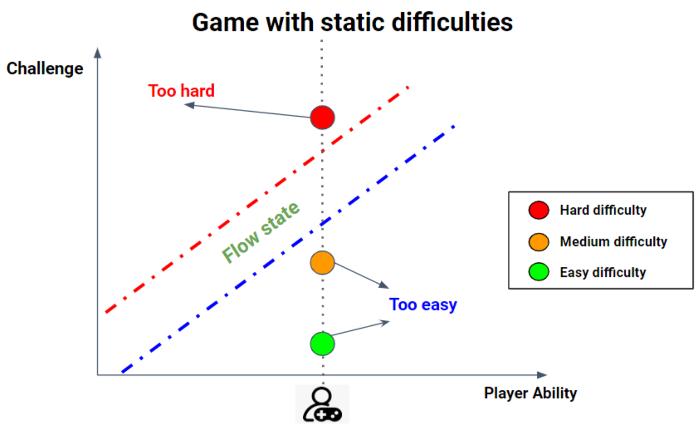


Fig. 1. Concept of being in the flow zone applied to a game with static difficulties, a player finds himself not fitting in any of the available difficulties, not allowing him to enter the desired flow zone.

Figure 2 then, shows an example of DDA being applied. The game adapts the difficulty of the game to suit the player his initial skill level and can cope with improvement or worsening of the player his abilities. It can also cope with the player changing how challenging he wants the game to be.

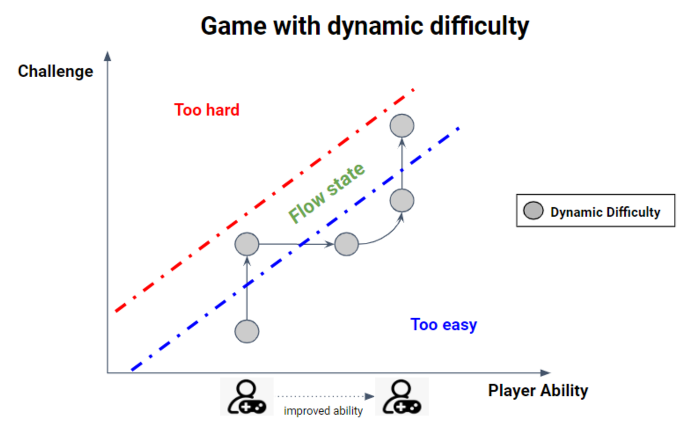


Fig. 2. Concept of flow applied to a game with dynamic difficulties, the game can adapt its difficulty to fit the player.

Nevertheless, DDA is not standard in games. There are multiple problems with implementing it successfully. First and foremost, estimating enjoyment of players, to know when to change the difficulty, is no easy task. Indications that someone is performing well like a high score, does not mean they are enjoying themselves necessarily. One must thus be careful relating player enjoyment to player metrics directly. Secondly, it is also important that the player, for passive DDA, doesn't realise the difficulty is changing. Else, the player might feel punished for performing well and explicitly play worse to get an easier experience.

B. *Player enjoyment*

The concept of being in the flow was thought of in 1975 and is the mental state that one enters when being completely immersed, focused on an activity[1]. Gamers their experience coincides with this description of flow when they are enjoying themselves. This is a more academic definition of why people enjoy playing games. A more simple approach is given by R. Koster [4]. There, fun in games is associated with the learning and mastery of the game. The fun is assumed as a result of the release of endorphins in our brain after players learn something.

Measuring feelings like enjoyment is no easy task. They are a complex state of the human nervous system. Different categories exist for measuring feelings[5][7]. A first category is self-report, where a user is responsible for reporting his feelings, be it verbal on a scale or visual using pictures. It is an inexpensive way of measuring feelings but can be vulnerable to users not being truthful about their feelings. A second category measures feelings by measuring some objective measure like skin conductivity and heart rate and try to match these occurring patterns to certain feelings. This category has the advantage of not being dependent on the truthful self-report of a human, but the results require a lot of knowledge to interpret correctly. The measurement equipment can also be obtrusive to the user experience and expensive.

C. *Reinforcement Learning*

Reinforcement learning is a machine learning technique that uses reward and punishment to shape the behaviour of an agent. That agent is operating in a certain environment. By rewarding certain actions and punishing others, a desired behaviour from the agent can be obtained. The basic concept of reinforcement learning is again illustrated in figure 3. A state can be described as the observation an agent receives from the environment. Using that state, the agent decides which action it performs and is subsequently rewarded or punished by the environment. Finally, the agent arrives in a new state.

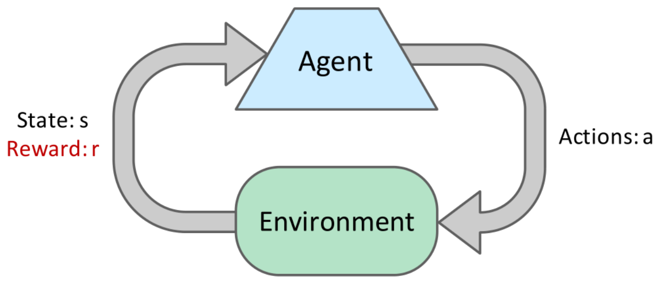


Fig. 3. Basic concept of reinforcement learning [2]

A combination of a state and action is called a state-action pair. Each of these state-action pairs has an associated value, called a Q-value. This Q-value is a measure of how much expected reward the agent will obtain by performing a certain action in that state. Recent developments in reinforcement learning have started to incorporate a neural network in their architecture, using it as a function approximator for the Q-values associated with the state-action pairs. This is called deep reinforcement learning[6]. Approximating these Q-values allows an agent to perform well even in environments with an enormous state-space.

III. Methodology

The motivation for this research is to apply DDA successfully to the game of Space Invaders using deep reinforcement learning. A trade-off will be made between how many difficulty parameters of the game can be adjusted compared to the amount of necessary data. The system its accuracy will be researched and validated.

A. *Space Invaders*

Space Invaders is a classic arcade game where a player must try and survive as long as possible against waves of slowly approaching enemies. The player can kill the enemies by shooting them, gaining him score. The game gives opportunity for different playstyles and has many difficulty parameters that can be adjusted. Examples are the enemy movement speed, bullet speed and the number of lives a player gets. Players also need to learn a couple of aspects of the game before they can advance further, like dodging enemy shots and hiding behind cover. Space Invaders can get boring quickly, as the game is slow in the beginning. There is therefore opportunity for DDA to improve this.

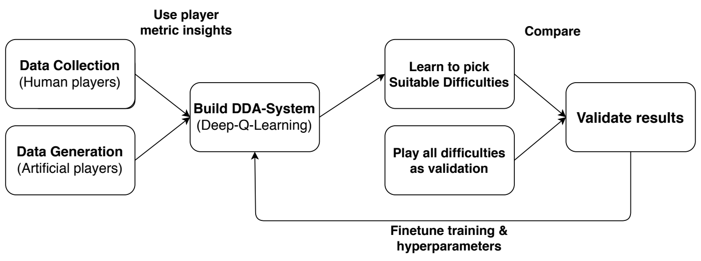

Use player metric insights
Data Collection (Human players)
Data Generation (Artificial players)
Build DDA-System (Deep-Q-Learning)
Learn to pick Suitable Difficulties
Play all difficulties as validation
Compare
Validate results
Finetune training & hyperparameters

Fig. 4. Methodology used for building a DDA system for Space Invaders using Deep-Q-Learning

B. Data Collection & Analysis

To gain insight into what game metrics relate to game enjoyment. Some data collection was to be done. Both human and artificial data will be collected, where a relationship between game data and enjoyment will be researched. Human players will be asked to indicate their mood before and after playing using the Pick-A-Mood tool [7]. Artificial players will have a like condition programmed into them that needs to be fulfilled. The human data is more interesting but also riskier, as it could be less reliable due to humans not truthfully filling in their mood. The artificial data is more reliable but less reflective of actual human players.

C. DDA System

The DDA system will be using Deep-Q-Learning to learn to select an appropriate difficulty for the player. It will use the player his performance on a certain difficulty as a state representation and will estimate what difficulty fits best to that particular player. Based on the player his likings and dislikings of difficulties, the system can over time learn what type of difficulty fits what type of player.

D. Analysing Results

Different experiments will be designed with the number of adaptable parameters in the difficulty increasing. The system its accuracy for picking a suitable difficulty will be validated against players playing all difficulties multiple times to know how likeable a difficulty is. An accurate system will thus serve the same difficulties as the players most liked difficulties. Besides that, a percentage liked games over the last hundred games served by the DDA system will be calculated for each experiment and persona respectively.

IV. Data Collection

The motivation for data collection is to find possible relationships between objective game metrics and player enjoyment. These insights can help with the design of our DDA architecture.

A. Human Data

For the human data collection, two setups were made in Ghent, Belgium. The two setups had different audiences, one primarily young children and the other adults. In both setups, players are asked to indicate their mood before and after playing one game of Space Invaders. While they are playing, several game metrics like score, keys pressed are logged at every game tick. The data collection resulted in over two thousand games being played, most of them coming from the youth library setup.

Analysing the datasets resulted in no clear relationship between enjoyment and game metrics. Averaged score and accuracy across different mood categories such as excited or bored, showed no large difference in mean value over time or spread. Besides that, the metrics were also differentiated in the first order, trying to perhaps uncover a pattern of behaviour related to moods. Again, no clear relationship was found. A big cause of this is the assumed inaccurate mood recordings. A large percentage of moods recorded stem from the default selected option in the game, introducing lots of noise in the dataset. The need for a more controlled setting of play is clear.

B. Artificial Data

To generate artificial data, three different artificial player personas were created, each playing the game in their style. The beginner AI plays much like a new player to the game, dying quickly and not accumulating much score. The safe AI uses a safe play-style that involves hiding behind cover and peeking out to shoot. Finally, the advanced AI plays riskier, not using the cover much. This persona can dodge enemy bullets well and accumulates score quickly. Each of these personas also has its specific condition for liking a game. The beginner persona has to shoot at least ten enemies and obtain a score of 500. The safe AI then likes a game when he survives two rounds of the game and obtains a score of at least 3000. Finally, the advanced AI likes a game when he obtains a score of 2000 and loses at least one life while playing. Letting these different personas play over six hundred games each on random difficulties provided metric analysis that show clear differences in performance between the personas. The average obtained score, for example, was 492, 1900 and 2855 respectively for the beginner, safe and advanced player. The different preferences of each persona along with the clear differences in-game metrics gives opportunity to optimise the game difficulty for each persona.

V. Results

A. Varying the enemy shoot interval

In a first experiment, the DDA system was tasked with finding a suitable difficulty for each AI persona, with three difficulties available. In these difficulties, the enemy shoot interval is varied, determining how much time there is between enemies shooting. A bigger interval means more time between enemies shooting and thus an easier difficulty.

As a validation, each of the AI personas first played 300 games on each difficulty, to see what difficulty is preferred by what persona. The results of this can be found in figure 5. It shows the beginner and safe AI liking the easiest difficulty, while the advanced AI has the biggest preference for the middle difficulty, providing some challenge. Afterwards, the DDA system was tasked to learn what difficulty fit what type of AI best. The average estimated Q-values are plotted in figure 6. The system can correctly associate the highest estimated Q-values with the most liked difficulty for each persona.

Looking at the like rates for the last hundred games played

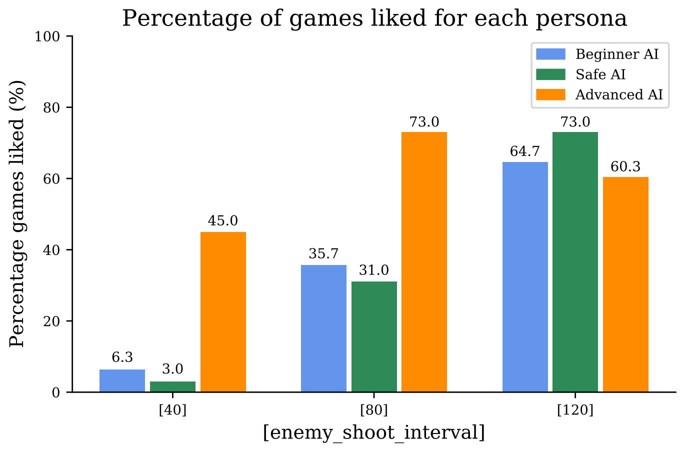


Fig. 5. Results of each difficulty being played 300 times by each persona with one adaptable difficulty parameter, ordered by increasing enemy shoot interval, meaning decreased difficulty. Show the beginner and safe AI preferring the easiest difficulty, the advanced the middle one

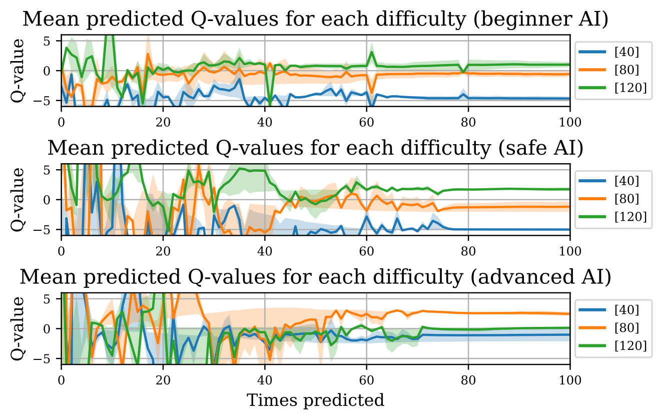


Fig. 6. Mean estimated Q-values for each persona with one adaptable difficulty parameter. Correctly reflects the likeability of figure 5 of each difficulty for each persona respectively

in table I, the difference between the percentage of last hundred games liked and the highest like percentage of any difficulty for each persona is small. Meaning that the personas are indeed presented with games that match their skill and have a high probability of liking them.

B. *Varying four different difficulty parameters*

In another experiment, four difficulty parameters are varied at the same time. These are the enemy shoot interval, amount of player lives, the horizontal movement speed of the enemies and the movement interval of the enemies. The number of lives a player has, determines how many times a player can get shot by the enemies before the game ends. The last two varying parameters affect how fast the enemies horizontally move and thus how fast the enemies approach the player as well. Low enemy movement intervals and/or a high movement speed result in fast-moving enemies. Each AI persona was made to play each difficulty one hundred times as validation.

These adaptable parameters resulted in eighty-one different

	Served by DDA	**Validation**
Type AI	**Like % (last 100 games)**	**Highest likeability % of any difficulty**
beginner	64.6	64.7
safe	71.4	73.0
advanced	70.7	73.0

TABLE I
PERCENTAGE OF LAST HUNDRED SERVED GAMES BY DDA SYSTEM LIKED BY EACH PERSONA WITH ONE ADAPTABLE DIFFICULTY PARAMETER AND THE MOST LIKED DIFFICULTY FROM THE VALIDATION RESULTS. SHOWS THAT THE DDA SYSTEM IS PERFORMING ADEQUATELY AS THE DIFFERENCE BETWEEN THE TWO METRICS IS SMALL.

difficulties that are available to the action space of the DDA system. Table II, displays the top three of the highest Q-values estimated for each difficulty when having the beginner persona play. It shows the beginner AI its preference for the easy difficulties, which corresponds with the validation results of highly liked difficulties. The difficulty column deals with the values of the following difficulty parameters in this order: player lives, enemy shoot interval, enemy movement speed intervals, enemy movement speed(horizontal).

Beginner top 3		
Difficulty	**% liked**	**Q-value (mean)**
[5, 120, [90, 60, 30, 15], 10]	99.0	4.86
[5, 120, [60, 40, 20, 10], 10]	98.0	4.62
[5, 120, [60, 40, 20, 10], 50]	99.0	4.58

TABLE II
TOP THREE MEAN Q-VALUES ESTIMATED FOR EACH DIFFICULTY FOR THE BEGINNER AI WITH FOUR ADAPTABLE DIFFICULTY PARAMETERS. SHOWS THE BEGINNER ITS PREFERENCE FOR EASY DIFFICULTIES.

The bottom three lowest estimated Q-values are also displayed in table III. The high dislike for the beginner AI for hard difficulties, meaning a low enemy shoot interval combined with high movement speed is displayed. This again corresponds with the low like percentages of these difficulties.

Beginner bottom 3		
Difficulty	**% liked**	**Q-value (mean)**
[3, 40, [40, 27, 13, 7], 30]	3.0	-5.0
[5, 40, [40, 27, 13, 7], 30]	9.0	-4.94
[4, 40, [40, 27, 13, 7], 30]	7.0	-4.9

TABLE III
BOTTOM THREE MEAN Q-VALUES ESTIMATED FOR EACH DIFFICULTY FOR THE BEGINNER PERSONA WITH FOUR ADAPTABLE DIFFICULTY PARAMETERS. SHOWS THAT HARD DIFFICULTIES CORRECTLY GET A HIGHLY NEGATIVE VALUE ATTRIBUTED TO THEM.

Similar results for the beginner AI are obtained for the safe AI as well, the preference for the easiest difficulties and dislike for the hardest ones. The preference of the advanced AI is different. It prefers difficulties that provide some challenge and are not the easiest, as seen in table IV. This is visible through the enemy shoot interval not always being the easiest, along with faster movement speeds and movement intervals. The bottom three difficulties for the advanced AI are similar to the beginner and safe personas. They correspond with the hardest difficulties.

Advanced top 3		
Difficulty	**% liked**	**Q-value (mean)**
[5, 80, [40, 27, 13, 7], 30]	92.0	4.39
[5, 120, [90, 60, 30, 15], 50]	92.0	4.1
[5, 80, [90, 60, 30, 15], 50]	83.0	4.0

TABLE IV
TOP 3 MEAN Q-VALUES ESTIMATED FOR EACH DIFFICULTY FOR THE ADVANCED PERSONA WITH FOUR ADAPTABLE DIFFICULTY PARAMETERS. SHOWS THE ADVANCED AI ITS PREFERENCE FOR DIFFICULTIES THAT PROVIDE SOME CHALLENGE AND ARE NOT THE EASIEST.

A summarising table that shows the like rate of the last hundred games served by the DDA system and the highest like percentage of any difficulty from the validation results is shown in table V. Performance from the DDA system is thus good, even with a sizeable action space.

	Served by DDA	**Validation**
Type AI	**Like % (last 100 games)**	**Highest likeability % of any difficulty**
beginner	93.9	99.9
safe	76.7	92.0
advanced	88.3	97.0

TABLE V
PERCENTAGE OF LAST 100 SERVED GAMES BY DDA SYSTEM LIKED BY EACH PERSONA WITH FOUR ADAPTABLE DIFFICULTY PARAMETERS. A COMPARISON CAN BE MADE TO THE HIGHEST LIKE PERCENTAGE OF ANY DIFFICULTY FOR EACH PERSONA. SHOWS THAT THE DDA SYSTEM IS PERFORMING ADEQUATELY AS THE DIFFERENCE BETWEEN THE TWO METRICS IS SMALL

VI. CONCLUSION

In this work, dynamic difficulty adjustment was applied using a deep reinforcement learning technique, called Deep-Q-Learning. A data collection was performed for human player data showing no clear link between player metrics and player enjoyment. Artificial data coming from rule-based AI personas was done as well, revealing more differentiating player characteristics. These personas were then used as test players for the DDA system. The system was able to pick suitable difficulties for each artificial player persona and was able to deal with an increasing amount of variable difficulty parameters, demonstrating scalability. Results show a high percentage of games being liked that are served by the DDA system for each of the experiments.

VII. FUTURE WORK

This work has multiple options for future work. A first being the use of a single network for all different AI personas, as currently a separate neural network was used for each. Preliminary experiments with a single network were performed but showed lacking performance. Multiple factors could be the cause of this, such as the need for a more complex neural network, a better input state representation as the system has to deal with different scaling numbers.

Research for using a continuous action space, rather than a discrete one, could be done as well. This allows the DDA system to be more accurate in serving the best difficulty for each player.

Finally, the human player dataset could be explored more as well. Clustering algorithms could identify clusters of player characteristics and find new insights in the relation between enjoyment and player metrics.

REFERENCES

[1] Csikszentmihalyi, Mihaly, *"Flow: The psychology of happiness*, 2013

[2] *"CS188: An intro to AI"* http://ai.berkeley.edu/home.html Berkeley AI Materials

[3] Hunicke, Robin, *"The Case for Dynamic Difficulty Adjustment in Games*, 2005, Association for Computing Machinery, 10.1145/1178477.1178573, Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, p. 429–433

[4] Koster, Raph and Wright, Will *"A Theory of Fun for Game Design"* Paraglyph Press, 2004, p. 42

[5] Mauss, Iris and Robinson, Michael, *"Measures of emotion: A review"*, Cognition & emotion, 2009, 02, p. 209-237

[6] Mnih et al *"Human-level control through deep reinforcement learning"* Nature 518, february, 2015, p. 529-33

[7] P.M.A. Desmet and M.H. Vastenburg and N. Romero, *"Mood measurement with Pick-A-Mood: Review of current methods and design of a pictorial self-report scale"*, "2016", "Journal of Design Research"

[8] Zook, Alexander and Riedl, Mark O *"A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment"* AIIDE 2012.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ANS	Autonomic Nervous System
DDA	Dynamic Difficulty Adjustment
DP	Dynamic Programming
DQN	Deep-Q-Network
MC	Monte Carlo
RL	Reinforcement Learning
TD	Temporal Difference

1 Introduction

The gaming industry today is a huge market[6]. More and more players are entering the market, with mobile gaming quickly becoming one of the most profitable segments of it. Gaming companies are always striving to maximise profits and player enjoyment, as the two go hand in hand. The more people enjoy their game, the more people will buy it.

One might ask then what the key part is of creating a game that has people going back to it after playing. The gaming industry applies many techniques to maximise enjoyment and retention of gamers. Design elements such as earning points for progress, online leaderboards and engaging storytelling are all techniques that help keep the players their engagement and retention. Many of these design elements have found their way into non-gaming contexts as well, a concept named gamification. Think of the for example of the badges one can earn on Google Maps for reviewing a certain place to eat or the avatar one unlocks for making progress in a certain app.

Creating a game that people enjoy is not easy though, even with gamification elements. People quickly get bored and go on to the next one. A big part of this is players getting bored with game is that is too simple, meaning it has not much to learn or master[7]. Games that are frustratingly hard, with a steep learning curve, are not enticing to pick up again as well[8]. How hard or easy a game is, is called its difficulty level. Getting the difficulty right in a game is challenging, because of the wide gaming audience. There is a wide spread in individual skill levels, be it experienced and new gamers as well as a spread on people their preference for being challenged. A game its difficulty has a big impact on potential enjoyment from players.

Traditional games apply one or multiple static difficulty levels, assumed by the game designers to be a wide enough range of difficulties to fit most players. This leaves some gamers frustrated, as these preset difficulties do not match their individual skill level and learning curve. This can occur when a player improves beyond the difficulty level he or she is playing on or changes playstyle from trying hard to playing relaxed for example. This is where Dynamic Difficulty Adjustment or DDA comes into play. With DDA, a game can dynamically adapt its difficulty to the individual playing the game and it can do this for any type of player and his accompanying skill level and learning curve. In general, it does this by building a model of the player to relate player enjoyment to some kind of other collected data, like for example game data or a player enjoyment survey. It then makes an informed decision about how to adapt the difficulty to the individual playing. There are different approaches to apply DDA such as using machine learning techniques like reinforcement learning, deep learning and more[9]. Reinforcement Learning or RL works by having an agent learn through punishment and reward. Through this reward system, certain desired behaviours can be stimulated. In the DDA case, this means that we reward an agent for picking a suitable difficulty for a player. Many flavours of RL exist for different purposes.

Applying DDA is not easy though, it comes with some challenges that remain difficult to address and are different for each game. This mainly stems from the fact that relating subjective game enjoyment and objective data is no easy task. Determining what approach suits a certain game is a necessary step. The DDA system also needs to be flexible and subtle enough such that a player does not notice huge changes in gameplay.

The motivation of this research then is to apply DDA to the classic arcade game of Space Invaders, an arcade shooter game, using a deep RL technique. It will be researched if it is possible to effectively apply DDA using deep reinforcement learning to correctly set the difficulty to the players their liking and ability. In order to gain insights on how enjoyment and player game data relate to one another, human player data will be collected, as well as artificial player data.

The contents of this thesis can be summarised as follows: In chapter two, a literature study discussing dynamic difficulty adjustment, player enjoyment and reinforcement learning is given. This is followed by the methodology chapter, where the different steps of the approach are laid out. Chapter four then deals with the collection and analysis of player data. After this, the architecture of the DDA system is explained, followed by a chapter discussing the results of several experiments. A conclusion is given in chapter six, which is followed finally by a chapter for future work.

2
Literature

2.1 Dynamic Difficulty Adjustment

Dynamic Difficulty Adjustment is a technique that is used in games to allow a game to adapt its difficulty to a player his characteristics hoping to improve the player his enjoyment. It is a subset of the more broad Challenge Tailoring (CT) where the game adaptations are not limited to changing difficulty, but also changing the game its content to optimise the player experience, on -and offline[10]. Besides these, the concept of challenge contextualisation deals with adapting the game its storytelling to fit the players its characteristics. An example here can be to present the player with a certain quest dependent on his emotions.

Both CT and DDA try to solve the mismatch that can occur in games where the player his ability does not match with the challenge the games is providing. The application of DDA is illustrated in figure 2.1. There, you can see a fictitious game having three static difficulties and one player, with a certain characteristic flow zone[11]. The flow zone is a state that a player can enter when he is truly focused on the game and loses track of time. This state is often accompanied by the most enjoyment while playing[12]. To reach this optimal flow zone, the player his abilities and the game its challenge must match, otherwise, the player can easily get frustrated if the game is too hard, or vice versa get bored because the game is too easy. The concept of flow is explained more in the player enjoyment section2.2.

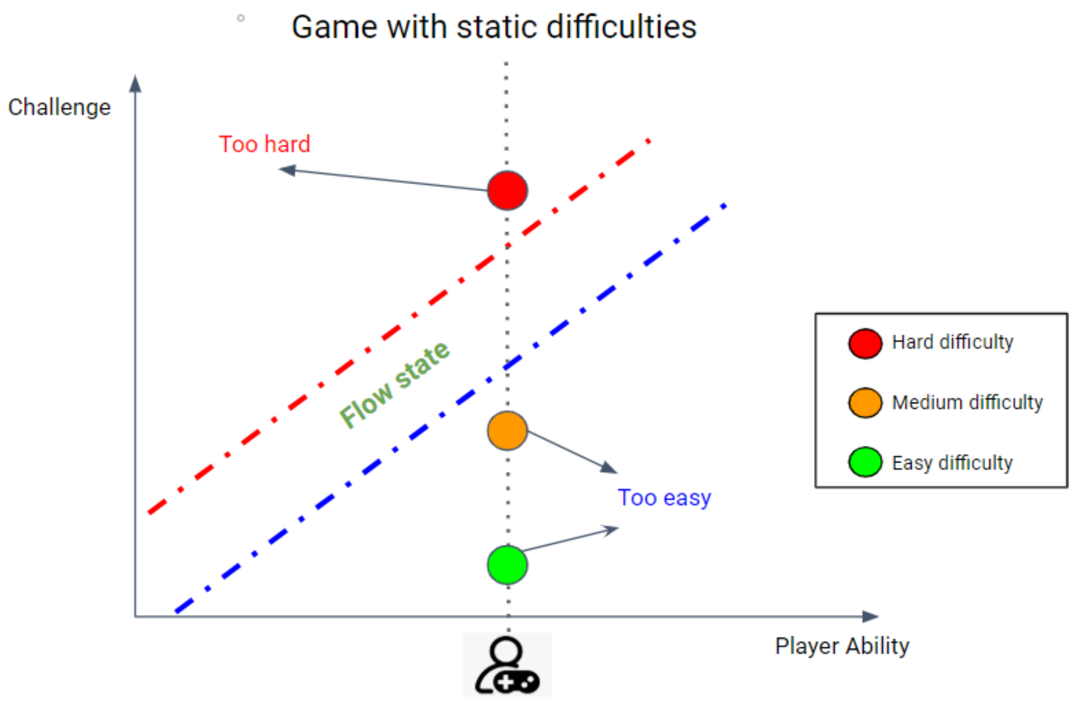


Figure 2.1: Concept of flow applied to a game with static difficulties, a player can find himself not fitting in any of the available difficulties.

DDA tackles this by allowing the difficulty to adapt over time while the player is playing. In this way, the game can try to keep the player engaged by, for example, increasing the difficulty when it seems the player is not engaged and performing too good and vice versa, making it easier if the player is performing badly. DDA can also account for the learning curve of the player, progressively making the game harder as the player improves his abilities. This as opposed to static difficulties becoming harder in the game based upon beliefs about how quick players will generally improve while playing the game. This is illustrated in figure 2.2 below.

There are two types of DDA, active and passive. Active DDA directly involves the player influencing the game its difficulty, this can, for example, be like choosing an easy or hard level in a game or some free a. Passive DDA, on the other hand, does this automatically, based on data that models the player his characteristics and current enjoyment.

While DDA is a suitable tool for maximising game enjoyment, it requires modelling of the player enjoyment. This information is necessary to make an informed decision about changing the difficulty. Ways of capturing and measuring this enjoyment are explained in the section about player enjoyment. Besides the player model, some sort of algorithm is also needed to change when and how much to change the difficulty of the game at certain points in time. This algorithm also needs to take into account how the player responds to the changes made in difficulty. A learning algorithm is beneficial here, as the algorithm can then adapt itself to

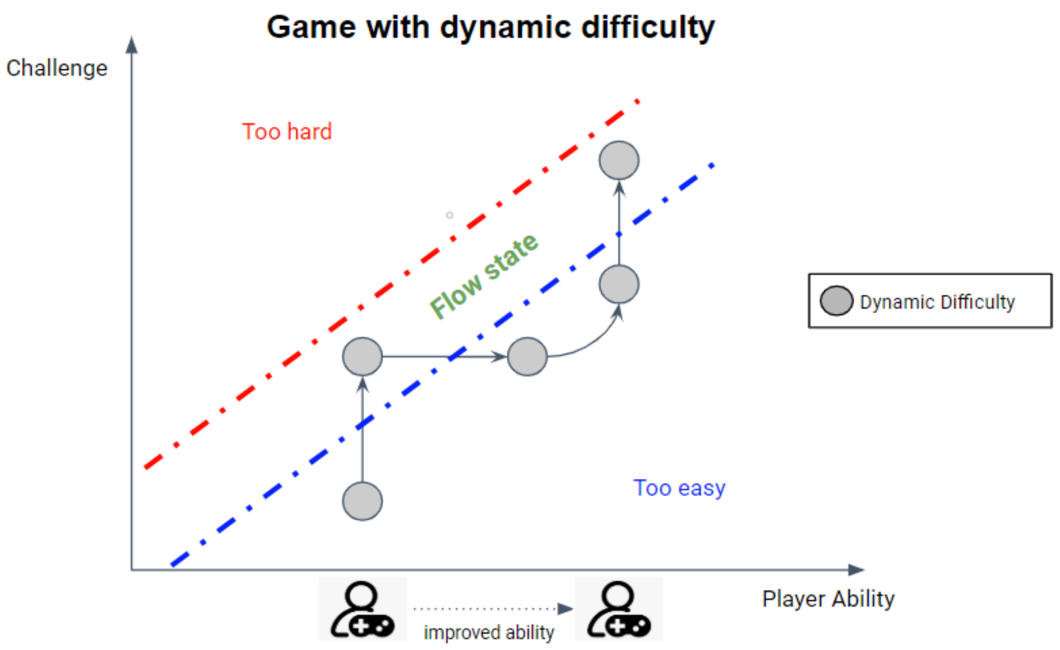


Figure 2.2: Concept of flow applied to a game with dynamic difficulties, the game can adapt its difficulty to fit the player.

each player his play style and use knowledge of previous instances to better predict future ones. There exist many approaches for applying DDA, a review is given in[9].

Applying DDA to games is not the industry standard, as it is difficult to apply correctly. This is because most applications make assumptions on the players their enjoyment, a subjective feeling, based on objective game data alone. **Game data that indicates a player is performing well, e.g. a high score per minute, does not necessarily indicate the player his enjoyment.** In passive DDA, it also important that the player has no sense that the system is changing the difficulty because the player could take advantage of this by playing worse to make the game easier. The player may also think the system is unfair because the game is more difficult the better he plays. This means that rigid changes in difficulty are not recommended, careful design is thus necessary to ensure this.

An example of DDA successfully being applied to a popular game is Left for Dead[13]. It is a multiplayer zombie survival game where, an AI director is used to determine the amount and type of zombies to fight the players, based on their performance, teamwork and pacing the makers of the game wanted to create. Another popular example is Mario Kart, a famous Nintendo racing game. Here DDA is applied to help bad performing players catch up with others, by giving them better power-ups than people at the front of the race.

2.2 Player Enjoyment

2.2.1 Cause of enjoyment

The concept of flow was named in 1975 and is thought of the mental state that one enters when being completely immersed, focused on an activity and possibly enjoying oneself while performing it[14]. One can think of this as the feeling of losing track of time while doing some sort of activity. The research identified the major components of an activity that can achieve the flow state. They are listed as follows:

- Requiring skill and challenge
- Clear goals
- Immediate direct feedback
- Sense of control
- Loss of self-consciousness and time

Not all components are necessary to give a person the experience of flow.

Gamers their experience when being completely immersed or focused on a game coincide with the experience of flow. Being highly focused in the flow zone, allows the player to maximise his performance and his enjoyment [12]. Each individual has its flow zone, which makes it hard to allow each player to achieve the flow state by the typical static difficulty experience provided by games.

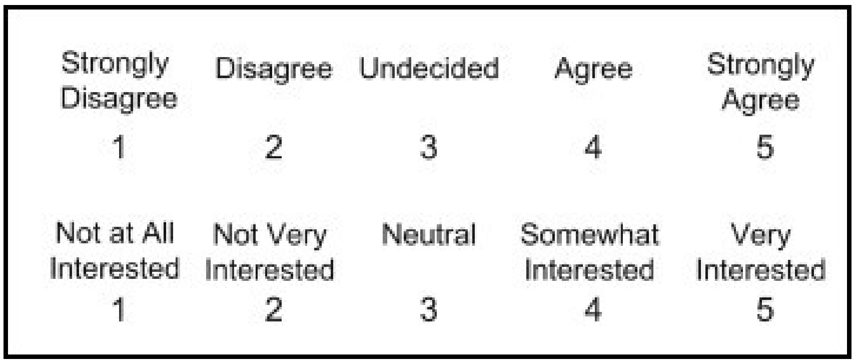
The above concept is a more academic definition of explaining why people enjoy playing games, a more simple approach is the one by R. Koster[7]. There, the fun of gaming is assumed to be coming from the mastery of learning of the game, because endorphins are released in the brain. Games are thus seen as exercises for our brain. It helps explain why some games are boring quickly. It means that the game has nothing more new to offer and the player does not learn new things, thus no endorphins get released and no fun is experienced. This also goes for games are too hard to master, where players have a difficult time learning the necessary techniques or strategies required to perform adequately because of there being too much information. A game thus requires a tricky balance between not overloading the player with new information and also being stimulating enough such that a player remains engaged. A simple example here is the game of Tic-Tac-Toe. As players learn the game, they quickly know that every game ends in a tie when played optimally. This means that there is not much to learn in the game, leaving players bored.

2.2.2 Measuring enjoyment

Measuring emotions is no easy task, as they are a complex state of the biological nervous system, comprised of thoughts, feelings and behavioural responses. The enjoyment people have while playing a game can be a good indication to a game developer to know whether his game is enjoyable. Different ways of capturing emotion and thus enjoyment exist. An overview of measuring emotions or feelings is given in [15]. Two main categories exist here, one where objective measures are used to represent certain emotional states or moods. The other type relies on some sort of self-report to this[16].

Self-report

A first, perhaps most simple one would be self-reporting. In self-reporting, the user is responsible for reporting his emotions. This can be done by using either a verbal or visual form of self-report. A verbal self-report measure usually consists of a checklist and questionnaire where a person must report his feelings on a scale. Responses typically consist of some adjective followed by some type of agreement or feeling. An example of such a scale found in questionnaires can be found in figure 2.3.



Strongly Disagree	Disagree	Undecided	Agree	Strongly Agree
1	2	3	4	5
Not at All Interested	Not Very Interested	Neutral	Somewhat Interested	Very Interested
1	2	3	4	5

Figure 2.3: Example rating scale found in verbal self-report questionnaire[1]

Visual self-reporting, on the other hand, does not rely on using text adjectives to measure some dimension of affect like mood or emotion, but rather uses pictorial scales. These have the advantage of requiring less effort of the user, are more intuitive and are not vulnerable to other interpretations through translation like with verbal scales.

Self-report is an inexpensive way of measuring emotions but has some caveats. Firstly, a person might not be honest about his feelings for a variety of reasons. The feelings could be socially unacceptable or he or she could rationalise what they should be feeling and report that instead. Self-report also works best when done as soon as possible after the emotions have been

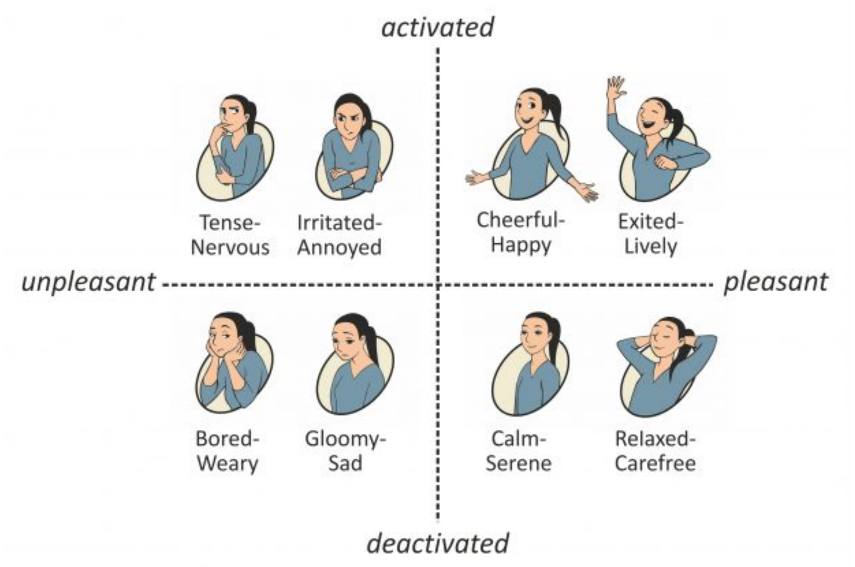


Figure 2.4: Example self-report tool. Pick-a-mood characters with corresponding mood quadrant[2]

experienced. Finally, self-report can also be vulnerable when not supervised, as people can just randomly fill in the report as quickly as possible.

An interesting self-report measure that mitigates some of the downsides of this category is called pick-a-mood. It is a character-based pictorial scale used for reporting moods. It is the first pictorial scale that captures distinct moods. It consists of different characters that each express a certain mood. Four categories exist, activated-pleasant, activated-unpleasant, deactivated-pleasant, deactivated-unpleasant. Four moods are available per each respective mood category. Tense and irritant for activated-unpleasant, excited and cheerful for activated pleasant, calm and relaxed for calm-pleasant, sad and bored for deactivated-unpleasant. These are displayed in the figure 2.4 below.

The pick-a-mood tool makes a distinction between emotion and mood. They refer to moods as being a low-intensity, diffuse feeling that can last a long time, while emotions are more short-term and high-intensity feelings that have a clear cause and direction. Mood and emotion also have different manifestations and while they do make the distinction between them, it is also stated that they have a dynamic interaction with each other. A combination of multiple different emotions can lead to certain moods.

One of the major strengths of this tool its simplicity and speed. No long explanation is needed before a user can correctly use the tool. Users only have to select one character to report their mood, which helps to avoid annoying users with overly long surveys and keep them motivated. Selecting this mood is also intuitive because the facial expressions and body language of the characters are clear and expressive. The tool also has been properly validated for its accurateness and has been used successfully as a tool in different research area's.

Autonomic nervous system

An alternative approach to self-report is to measure physiological measures of the autonomic nervous system or ANS of a human and measure things like skin conductivity and heart rate. Studies have shown that emotional states are linked with specific patterns of ANS activity[17]. One of the advantages of such an approach is having a continuous measure of the user his state in real-time, without any interference of the user himself. Besides that, these measures are not biased by the user himself, be it for social or cognitive desirability.

A downside is that measuring these factors can be intrusive to the user and expensive to implement. Modern wearables have alleviated some of this, as they provide an inexpensive and natural way of having physiological measures. Still, the results of these measurements require a great deal of effort, knowledge and time to interpret correctly, as they can easily be disturbed by other influences in the surroundings of the user. However, these ANS approaches fail to capture the distinction between mood states, as existing technologies only can capture the amount of arousal or affect.

Behavioural Characteristics

Another option besides ANS, also of the objective category, is using behavioural characteristics of humans as a way of representing emotions. There exist devices that can measure certain characteristics such as facial expressions and body posture which then correspond with certain emotions. Other characteristics such as eyelid closure patterns, vocal expression and interaction behaviour, are also used.

The main advantages of this approach are similar to those of ANS. Meaning it is an objective, continuous measure. Besides that, they can sometimes even be used without the user knowing he is being monitored which also helps avoid cognitive or social biases. These systems are also generally inexpensive, besides hardware cost of measuring the characteristics. A big downside is that to have a reliable measure, each system has to be individually trained for each individual. Meaning it can require a lot of time and effort to map patterns of characteristics to emotional states for each user.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique that uses reward and punishment to shape the behaviour of an agent such that it solves a certain problem by achieving a goal or maximising a certain score metric. It handles problems where actions are correlated with a delayed reward, where other machine learning methods can have a more difficult time with understanding which actions lead to a desirable outcome.

RL differs from more traditional machine learning techniques like supervised and unsupervised learning. Unsupervised learning mostly works by trying to recognise similarities in a given dataset, so it can perform clustering or anomaly detection for example. Supervised learning, on the other hand, requires a labelled dataset for the algorithms to learn. Acquiring such a dataset is a lot of work and thus expensive. Reinforcement learning has the agent itself acquiring experiences in the environment, these experiences can be thought of the labels in a supervised learning environment. This is then coupled with delayed and sparse feedback.

Reinforcement learning problems are most often represented as Markov Decision Problems (MDPs).
These kind of problems consist of a:

- Finite set of states:
$$s \in S \tag{2.1}$$
- Finite set of actions:
$$a \in A \tag{2.2}$$
- Set of transition probabilities between states given an action and state:
$$T(s,a,s') = Pr[S_{t+1} = s' \mid S_t = s, A_t = a] \tag{2.3}$$
- Reward function
$$R(s,a,s') = E[R_{t+1}|S_{t+1} = s', S_t = s, A_t = a] \tag{2.4}$$
- Discount factor
$$\gamma \tag{2.5}$$

Being a Markov process also means that the transitions between future and past states are conditionally independent. This means that all information about the future is encapsulated in the current state the agent finds himself in.

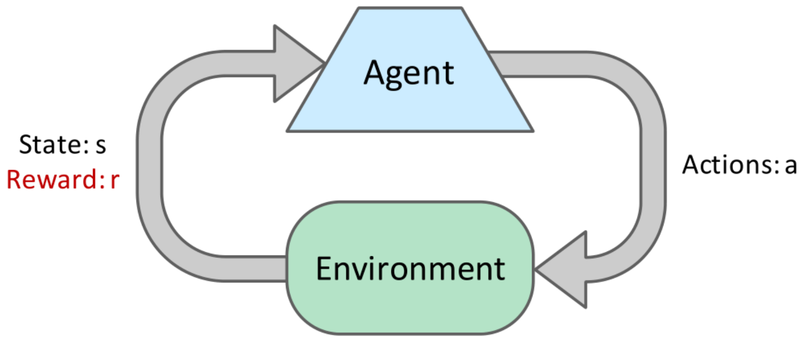


Figure 2.5: Basic concept of reinforcement learning[3]

Reinforcement learning problems differ from MDPs though, as the agent does not know the reward function and transition model beforehand. The agent must perform actions himself to learn.

The basic idea in RL is, that an agent which currently residing in some *state* $S_t \in S$ at time step t, can choose an *action* $A_t \in A(s)$ in the environment. Once the action has been performed, the agent receives a reward *reward* from the environment, $R_{t+1} \in R$ and resides in a new state S_{t+1}. These rewards are then used by the agent to shape its behaviour in the future to maximise the obtained reward. A discount factor allows the agent to focus on short or long term rewards, by discounting rewards that are further away in time. These key concepts are again illustrated in figure 2.5.

Finally, the agent needs to learn a control policy $\pi_t(s)$ which maps a certain state to a certain action. This policy is what allows the agent to solve the actual problem at hand. Each policy has an accompanying value function. This value function describes the value (quality) of a state under a certain policy. It thus is the expected cumulative discounted reward an agent receives while operating under a policy, starting from a state.

$$V^{\pi}(s) = R(s) + \gamma \sum_{\forall s'} T(s, \pi(s), s') V^{\pi}(s') \tag{2.6}$$

An optimal value function or V^* is the best value function across all policies. It thus states what action maximises the total expected reward when in state s.

$$V^{*}(s) = R(s) + \max_{a} \gamma \sum_{\forall s'} T(s, a, s') V^{*}\pi(s') \tag{2.7}$$

Using this optimal value function, an optimal policy can also be described. This policy gains the agent the biggest expected cumulative reward when starting from state s.

$$\pi^*(s) = arg\max_a \gamma \sum_{\forall s'} T(s,a,s') V^* \pi(s') \tag{2.8}$$

These three above equations are also known as the bellman equations and are key for finding an optimal policy.

Many flavours of reinforcement learning exist[18]. A first distinction is the learning of a model. A model in RL is the knowledge of the different transition probabilities that exist between states when performing a certain action and the knowledge of the reward function. RL flavours where the agent attempts to learn a model capturing the reward function and transition probabilities are called model-based.

Value iteration is such a model-based technique. Value iteration consists of two sequential steps. In the first step, the agent starts with a value function that returns zero for every state. This function is then repeatedly improved according to the Bellman optimality equation found in equation 2.7.

$$\pi^*(s) = arg\max_a \gamma \sum_{\forall s'} T(s,a,s') V^* \pi(s') \tag{2.9}$$

Then, the optimal policy extracted from this optimal value function, again using the above bellman equations.

Model-free methods, on the other hand, use the trial-and-error experiences of an agent directly to build a policy and do not try to capture a model of the world. This is advantageous for problems where there is no knowledge of the transition function and/or reward function. In model-free learning, the agent experiences the world through episodes, each one containing a state, action and reward triplet. Similar to the value function of a state under a certain policy, a function that describes the quality (Q-value) of a state-action pair is defined.

In Monte Carlo methods of model-free learning, the agent uses these experienced episodes to build an approximation of the expected return of each state, the expected return is simply the mean reward observed by the agent over all episodes and is updated after each episode. This means that MC methods can only be applied to episodic problems. This can easily be extended to state action pairs by counting the number of times the agent visits a certain state and performs a certain action. The agent finds the optimal policy by greedily selecting the action that maximised the expected reward for each state-action pair each episode and eventually will obtain an optimal policy. This approach explores non-optimal actions because it is optimising a near-optimal policy that still performs some exploration. It is an example of an on-policy approach, another distinction in RL methods, meaning the target policy and the one being followed, are the same.

In off-policy methods then, multiple policies can be followed at once. With the policy being learned called the target policy, and the followed one the behavioural policy. The agent can perform a balance between exploration and exploitation by selecting a sub-optimal action, thus following the behavioural policy, with some small probability , where a balance must be struck between exploring and exploiting, controlled by this value. Off-policy methods are more powerful, as they have a lower chance of getting stuck in local optima, but also more complex since they require additional effort to implement and can be slower to converge.

Temporal Difference (TD) learning methods are another form of model-free learning. On the one hand, TD learns directly from trial-and-error experiences without a model, like MC methods. However, TD has the advantage of being able to update the value of state at each time step, instead of episodically like MC. On the other hand, TD updates estimates based on other estimates like in DP, meaning they build a guess from another guess. Luckily, TD can still guarantee convergence of the optimal value function under a policy and is in practice, faster than MC methods[18]. An important off-policy TD algorithm, called Q-learning, learns the Q-values associated with state-action pairs. It is defined by equation 2.10.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}; a) - Q(S_t, A_t) \Big] \tag{2.10}$$

It thus tries to approximate the optimal q function, q*. Q-learning is guaranteed to converge to this optimal q function. When dealing with a simple problem, meaning a small action and state space, q-learning can be applicable. This changes as complexity goes up. For real-world problems, state-spaces can be infinitely large. It thus quickly becomes infeasible to maintain a Q-value for each possible state-action pair. One solution for this problem is to use approximate Q-learning, where states are represented by a linear combination of handcrafted features, reducing the state space by approximating it. An example of this is, for example, using the number of queens, kings, ... on a board of chess to represent its state. Creating these features for approximation is not easy and they are hard to come by.

2.3.1 Deep Reinforcement Learning

Deep reinforcement learning is the combination of two fields, reinforcement learning and deep learning. Deep learning is a subset of machine learning as is reinforcement learning, which both are a subset of artificial intelligence. All give the possibility to let machines perform a task that otherwise would require human knowledge. The deep in deep learning refers to the use of multiple hidden or deep layer of a neural network, that is used for learning a task.

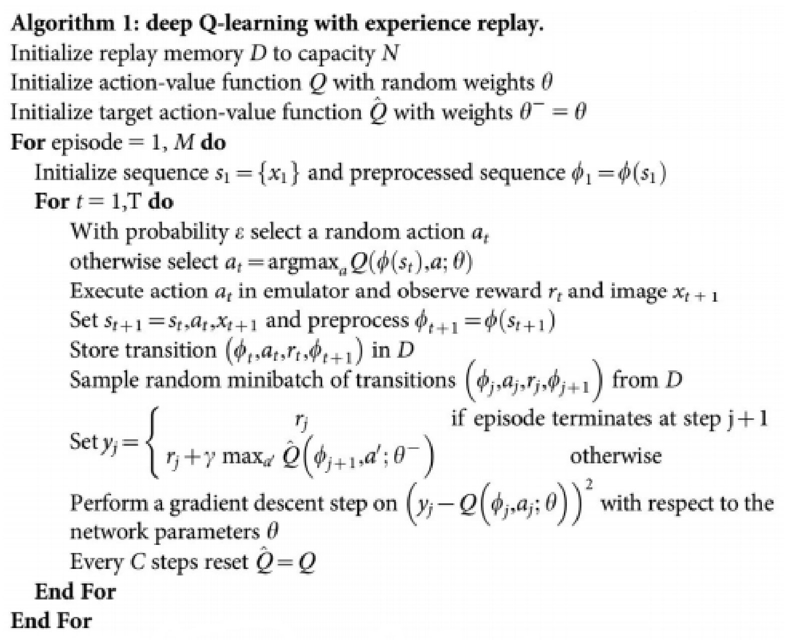
An artificial neural network or ANN, is a collection of neurons, simple processing units, that are densely interconnected with each other and is vaguely based on the neural network occurring in the brain of a human being. A neuron has several weighted inputs, stemming from other connecting neurons, that are summed together and used as an input for a certain activation function, that defines the output of the neuron. Each of these processing units, has a collection of inputs, coming from other processing units, and a collection of outputs. Each of these connections has an associated weight with them. By adjusting these weights in a certain manner, neural networks can be trained to perform a task. These ANNs are the means to recognise patterns in all kinds of data, be it sensory data like images or time-series of scalar numbers. ANNs are often used to perform classification tasks, like object recognition, but can be used for many other purposes.

The network is comprised of three types of layers. An input one, responsible for accepting input data, such as an image for example. The input layer sends this information to the hidden layer, where the black box aspect of neural network happens. Here, the network is trying to recognise different kinds of patterns in different hidden layers. As an example, an object classifying ANN might try to recognise edges in a first hidden layer, a more elaborate pattern in the next and so forth. Finally, the hidden layer connects to an output layer, which can assign a probability to a certain outcome or label. Neural networks are in this way a function approximator, they map an input to a certain output, and this function is what is learned inside the neural network.

The use for neural networks in reinforcement learning is to use it as a function approximator for a value or policy function. Meaning the ANN can map states to values, or it can map state-action pairs to Q-values. An ANN can thus be used to decide which action to take in a certain state, by for example outputting the Q-values associated with each action for the input stake and selecting the action that is associated with the highest Q-value. The feedback loop from obtaining a reward from that action allows the neural network to improve its prediction of expected reward and received reward as the agent acts in the environment. This is the same approach as for neural networks that use supervised learning, however, supervised learning has a set of labelled examples available to it, whereas in reinforcement learning the agent has to acquire this for himself.

The use of a non-linear approximator can be very helpful in environments with a big action and state space, as it quickly becomes infeasible, to for example for Q-learning, keep a tabular Q-value for each possible state-action pair. Combining reinforcement learning with non-linear approximators comes with some challenges. In combination with Q-learning for example, a model-free technique, it could lead to the Q-network diverging and thus an unstable network. Recent developments have tried to mitigate these by introducing some new mechanisms to deep reinforcement learning[4].

Using nothing but raw pixel values as an input, RL agents can learn to play complex games like



```
Algorithm 1: deep Q-learning with experience replay.
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
For episode = 1, $M$ do
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t$ = 1,T do
     With probability $\varepsilon$ select a random action $a_t$
     otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
     Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
     Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
     Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
     Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
     Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
     Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the
     network parameters $\theta$
     Every $C$ steps reset $\hat{Q} = Q$
  End For
End For
```

Figure 2.6: Deep-Q-learning algorithm by DeepMind[4]

Starcraft II, a real-time strategy game developed by Blizzard and even dominate professional players[19].

Deep-Q-learning

Deep-Q-learning is a deep reinforcement learning technique first introduced in 2013 by V. Minh et al to learn an AI agent to play Atari 2600 arcade games like Space Invaders and Breakout. It is an off-policy, model-free RL technique and is a variant of Q-learning where a convolutional neural network (CNN) network is used as a function approximator. This CNN uses raw pixel data of the game as an input and outputs a value function that predicts the future rewards obtained from different state-action pairs. The weights of the network are updated using RMSprop[20], an adaptive learning rate method that is suitable for working with mini-batch updates. It was successfully applied to seven games where it was able to outperform a human expert on three of them[4]. The complete algorithm can be seen in figure 2.6.

The algorithm introduces two mechanisms to enhance stability and prevent divergence of the Q network. Deep-Q-learning uses experience replay. It is a biologically inspired mechanism to help some of the divergence problems that stem from using neural networks as value functions for Q-learning. It works by storing an agent his experiences (state, action, reward, next state) in a replay memory D with capacity N. This replay memory is then randomly sampled, for a certain fixed batch size. Each of the transitions in a sampled batch is used in the learning process of the neural network, to better predict future values for state-action pairs. In this way, the learning process and experience gathering are logically separated. This allows the network to more efficiently learn from experience, as experience can be used multiple times for learning. Besides that, using randomly sampled batches of data instead of the sequential samples helps avoid training on correlated data, stimulating convergence of the network.

Deep-Q-learning also introduces the use of fixed Q targets in the learning process. This is done by having two Q-networks in the algorithm. One network is used to obtain the predicted Q-values for each state-action pair, the other learner network is constantly updated with each batch. Only periodically are the updated weights from the learner network synchronised with the predictor network. This keeps the target function from changing too quickly, which can cause instability in training.

3 Methodology

The goal of this thesis is to research whether or not DDA can be successfully applied to the game of Space Invaders. Deep reinforcement learning will be applied using the player metrics as an input for it. A trade-off between how many difficulty parameters can be adjusted compared to the amount of necessary data and system accuracy will be researched.

In what follows is an explanation of the methodology followed for applying DDA in the game of Space Invaders.

3.1 Game Mechanics of Space Invaders

In the game Space Invaders, a player controls a space ship, which he can fly to the left and right. In each level, the player has to fight enemy aliens, who spawn in formation at the top of the screen. The player can kill the enemies by shooting them, gaining him score.

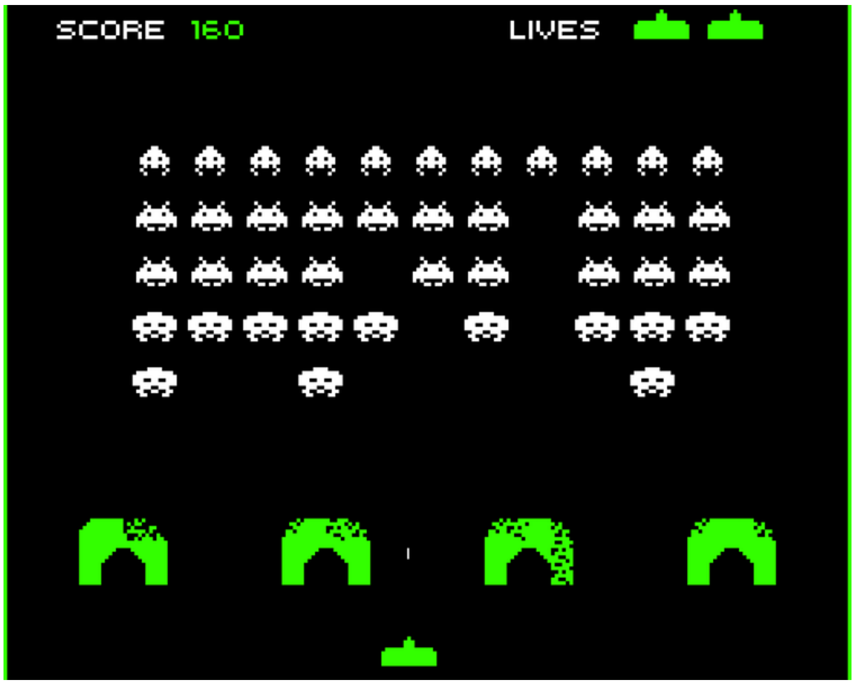


Figure 3.1: Original gameplay of the arcade game Space Invaders[5]

The enemies also move, iteratively to the right and left, with a small movement downwards, each time an alien hits the respective edge of the screen. The more enemies a player kills, the faster they will move. When a player kills all enemies displayed on the screen, he advances to the next level, where the same amount of enemies are presented, but start lower on the screen. This gives the player less time to kill them, leading to a slightly more difficult experience. The ultimate goal is then to achieve the highest score, as the game is never-ending until the enemies spawn directly on top of the player. This is because whenever the player his ship collides with an alien, the game is immediately over. All the while, enemies also shoot back at the player his ship, with the player losing one life if he gets hit. If the player does not have any more lives, the game also ends. The player can use some, destructible cover as well. These covers absorb enemy and player shots and exist of multiple layers. Finally, a random mystery ship also occurs at random times in the game, the player can shoot this ship as well, earning him a lot of score. The original game was built for arcade machines. a modern clone of the game was made by Lee Robinson[21] and is used for this work.

The game gives opportunity to different play-styles. A player can opt to play safe and mainly use covers. On the other hand, a player can also choose to play risky and try to kill as many enemies as quickly as possible by positioning himself in the centre mass of the enemies. A beginner player thus has the opportunity to learn these different styles, and alternate between them as the game progresses, as they improve the chances of obtaining a high score. The DDA system will try to give opportunity for both playstyles to thrive.

The game has a wide range of difficulty parameters, like the number of player lives, for example, that can be adjusted. A full overview of adaptable difficulty parameters in the game is given in 4.2. These parameters can have a great impact on the game its experience, as they can either make the game frustratingly hard, too easy, or just right.

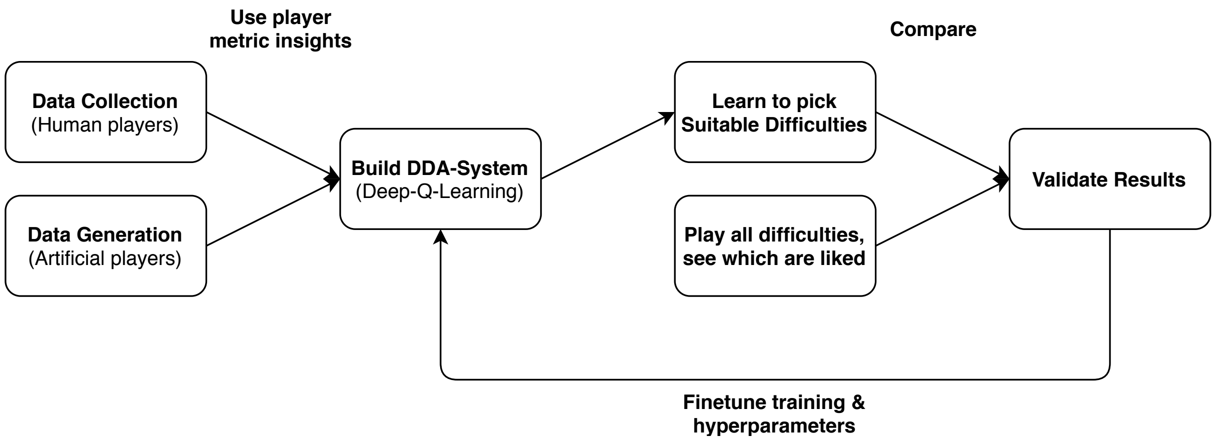

Use player metric insights
Compare
Data Collection (Human players)
Data Generation (Artificial players)
Build DDA-System (Deep-Q-Learning)
Learn to pick Suitable Difficulties
Play all difficulties, see which are liked
Validate Results
Finetune training & hyperparameters

Figure 3.2: Methodology for building a DDA system for Space Invaders using Deep-Q-Learning

3.2 Data Collection and analysis

To apply DDA, player game data is to be collected. This is necessary to give insight into what game metrics can relate to the enjoyment of the game and thus the necessary information for our DDA system to make an informed difficulty adjustment. Both human and artificial data will be collected. For human data, moods before and after playing will be recorded using the pick-a-mood tool, to try to find correlations between enjoyment and objective game data. The precise application of this tool will be explained in the data collection chapter. As for the artificial data, different AI personas will be created representing a type of player, with certain characteristics, and games will be generated respectively. Having human player data is more interesting than artificial data, but, it can also be less reliable and noisy. The artificial data then is reliable, but less reflective of actual human players. Collecting and analysing both helps alleviate the risk of working with bad data.

3.3 DDA System

The DDA system will be approached using a deep reinforcement learning technique, more specifically Deep-Q-Learning, as discussed in section 2.3.1. The Deep-Q-Network or DQN will be responsible for picking the best suitable difficulty for the player. Instead of using an image as input, the DQN will be fed metrics that describe a players performance on a certain difficulty. Based on this performance, the DQN will have to estimate what difficulty is suitable for this player by predicting the corresponding Q-values of each state-action pair, with one action being one certain difficulty. The DQN learns, over time, what difficulties are liked by the player by keeping track whether or not through player indication after each game. The reward or punishment a DQN receives for each difficulty picked depends on this indication.

3.4 Analysing Results

Different experiments will be designed with increasing complexity, thus increasing the number of difficulty parameters that can be varied, where hypotheses will be made about what type of difficulties will be liked by what type of player.

For each experiment, each difficulty will be played a numerous amount of times, to know how likeable a difficulty is for each player. These like percentages will be used as a validation tool when compared to the Q-values estimated for these difficulties. Highly liked difficulties should obtain corresponding high estimated Q-values. On the other hand, highly disliked difficulties should obtain corresponding low estimated Q-values. Besides, a like rate over the last x amount of games will be shown as well. To validate that the network is indeed picking suitable difficulties for the player.

4
Data Collection and Analysis

4.1 Motivation

To apply DDA to Space Invaders, player data was to be collected. The goal of this collection was to find possible relationships between player game data and player enjoyment. These relationships can help the DDA system to know when to make the game difficulty harder or easier.

4.2 Types of data

The first type of data to be collected were player game metrics. A summary of all the different game metrics that are logged is given below in table 4.1. These metrics are logged at every game tick. All these metrics are dumped in a file after playing the game. An advantage of logging at every game tick, because of the deterministic nature of the game, is that is possible to completely replay a game with only a log file. This allows looking into interesting games where a player might have gotten very far or vice versa where they died very quickly. It also validates the logging system as replayed and played games must have the same outcome.

The second type of data collected was the perceived mood of the player. Before playing a game,

	Difficulty Parameter	Type	Description
1.	game_ticks	Integer	Amount of game ticks elapsed
2.	time_elapsed (ms)	Integer	Amount of time elapsed
3.	ticks_behind_cover	Integer	Amount of ticks a player has spent behind cover
4.	n_keys	Integer	Amount of time a player has kept one or more keys pressed while playing
5.	n_keys_up	Integer	Amount of different keys a player has pressed
6.	enemy_bullets_killed	Integer	Amount of enemy bullets killed by a player
7.	enemies_killed	Integer	Amount of enemy aliens killed by a player
8.	n_mysteries_killed	Integer	Amount of mystery ships killed by a player
9.	enemies_seen	Integer	Amount of enemy aliens seen by a player
10.	enemies_left	Integer	Amount of enemies left to kill when the game is over
11.	enemies_shots_fired	Integer	Amount of shots fired by enemy aliens
12.	score	Integer	The player his current accumulated score.
13.	round_reached	Integer	Player his current round reached
14.	dead_by_bullet	Boolean	Indicates if the game was ended if the player died by an enemy bullet (0 lives left)
15.	dead_by_invasion	Boolean	Indicates if the game was ended by an alien reaching the player
16.	lives_left	Integer	Amount of lives left when the game is over (when player is dead by invasion)
17.	distance_between_aliens_and_player	Integer	Amount of distance between a player and the closest enemy (upon game over by a bullet)

Table 4.1: Summary of different captured game metrics.

players were asked to indicate their current mood. This was done using a circular pattern of the 9 pick-a-mood characters, rotating between the male, female and robot version of the characters each game. The tool was used without labels for each corresponding mood, as was suggested in the manual of the tool[22]. After the game ended, players were again asked for their mood, using the same figure. Screenshots of this process are available in 8.

Each game was initialised with a random difficulty, with difficulty parameters having a range of allowed values each. This difficulty data is also logged. Using a random difficulty for each game was done to get a wide spread on presented game difficulties and thus also on experienced moods while playing. A summary of all difficulty parameters that were randomly initialised can be found below in table 4.2.

4.3 Collecting human player data

A first data collection was done for human player data. The game was set up at two locations, the first being the youth section at the Krook library in Ghent, Belgium. Here an arcade box, which was kindly provided for use by imec, was used to make the game available to the public. The game was controlled using one joystick to move left and right and one button to shoot. Users used the joystick and the button to select a mood on the circular mood palette that was displayed. A poster with some explanation about the game was also attached to the arcade

	Difficulty Parameter	**Type**	**Description**
1.	nr_enemies_rows	Integer	Amount of enemy rows
2.	nr_enemies_cols	Integer	Amount of enemy columns
3.	dual_bullets_after	Integer	Score which a player has to reach before getting the dual bullets upgrade
4.	enemy_shoot_interval	Integer	Time (ms) between enemies shooting
5.	enemy_movement_speed_x	Integer	Movement speed in the horizontal direction (left or right) of the enemy aliens
6.	enemy_movement_speed_y	Integer	Movement speed in the horizontal direction (towards the player) of the enemy aliens
7.	enemy_starting_height	Integer	Determines the starting vertical position of the highest row of aliens
8.	enemy_bullet_speed	Integer	Vertical speed of enemy bullets
9.	regen_blockers	Boolean	Determines if cover blocks regenerate after each round
10.	enemy_move_interval	Integer	Time (ms) between enemies moving
11.	players_lives	Integer	Total amount of lives of a player

Table 4.2: Summary of difficulty parameters that were randomly initialised for each game played

box. The game was also set up in Dutch as opposed to English. The main visitors of the youth library are young kids, sometimes accompanied by their parents. In figure 4.1 the actual setup is shown.

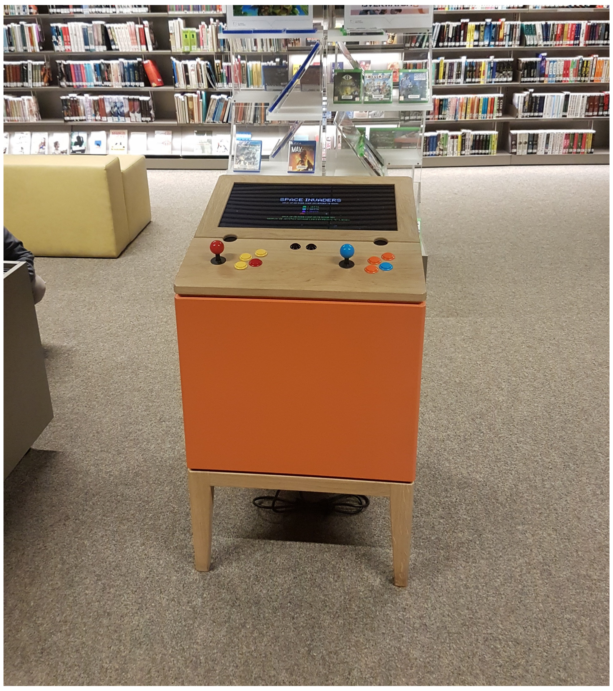


Figure 4.1: Setup for data collection of people playing Space Invaders at youth Library of De Krook.

A second location was also set up, again at de Krook, but now targeting a more adult audience inside an office with over fifty people working. Here, a normal desktop computer was used where users played with a keyboard to move and shoot. Users indicated their mood by pressing a number corresponding to the mood on the pick-a-mood palette that was displayed. Other than the type of input used by the players and language of the game, no differences between the two setups existed.

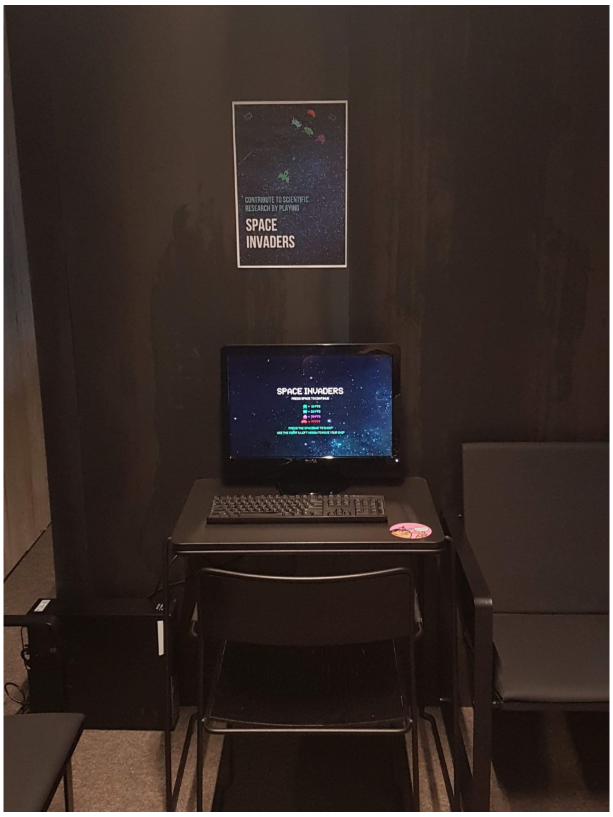


Figure 4.2: Setup for data collection of people playing Space Invaders at offices of UGent and Imec at De Krook

A setup for a third location, at the iGent Tower in Zwijnaarde, was also planned and set up. It could not be continued because of the ongoing corona pandemic. No data was thus captured and analysed from this setup. The full game flow a player goes through for playing one game can be found in the appendix8.

The goal of having two locations was to be able to compare the results between the younger and more adult audience to see if there were any noticeable differences in game and mood metrics. With the assumption that people were to pick their moods more responsibly and thoughtfully at the adult setup, it could also validate the distribution of moods recorded at the youth library setup. Finally having more data is always helpful, as more data gives more opportunity to interesting conclusions and could be useful for our DDA system as a validation tool.

4.3.1 Analysis

The two setups, running for approximately a month and a half resulted in a total of 2123 games being played, 2067 of them were played in the youth library and fifty-six at the offices at the fourth floor. Games that were played, but while playing left behind, were not recorded. Due to the low amount of games for the second setup, the analysis will not have strong statistical strength, nevertheless, it can still show interesting results. In a first step, some preprocessing is done to the datasets, after which the recorded moods post-game are analysed. Then, game metrics are analysed by use of violin plots and other graphs.

Preprocessing

Before the first analysis, the datasets needed some preprocessing. This is because the setups were not monitored at all and people could thus, pick a random mood to get in a game quickly, which is probable considering the young kids present at the youth library. This was verified by the fact that the two default selected moods at the selection screen had the highest overall number of records. The choice was thus made to remove games from the datasets that had a before-and-after mood picked that were the default selected ones. This led to a reduction of 370 games for the youth dataset. This abundance of default selected moods was not present for the dataset coming from the office so no further preprocessing was needed.

Mood before and after playing

The first point of interest in the data was the recorded moods using the pick-a-mood tool before and after playing the game. Figure 4.3 shows a histogram of the recorded before and after moods for the youth library dataset. It shows that irritation increases significantly, from five % of moods before the game to fifteen % after playing. Tenseness also increased by seven % after playing. The rest of the recorded moods dropped in frequency after playing, the biggest decrease was seen in excitedness, a six % drop. The same histogram for the second dataset shows a similar story, with irritation increasing and other moods decreasing.

The moods were categorised and plotted with their respective mood category and are shown in figure 4.4. Unpleasantness goes up, pleasantness goes down, both for activated and deactivated moods.

These findings go hand in hand with the fact that the games uses a static difficulty, as people are having either too hard of a time, or find the game non-engaging because of the non-existent challenge.

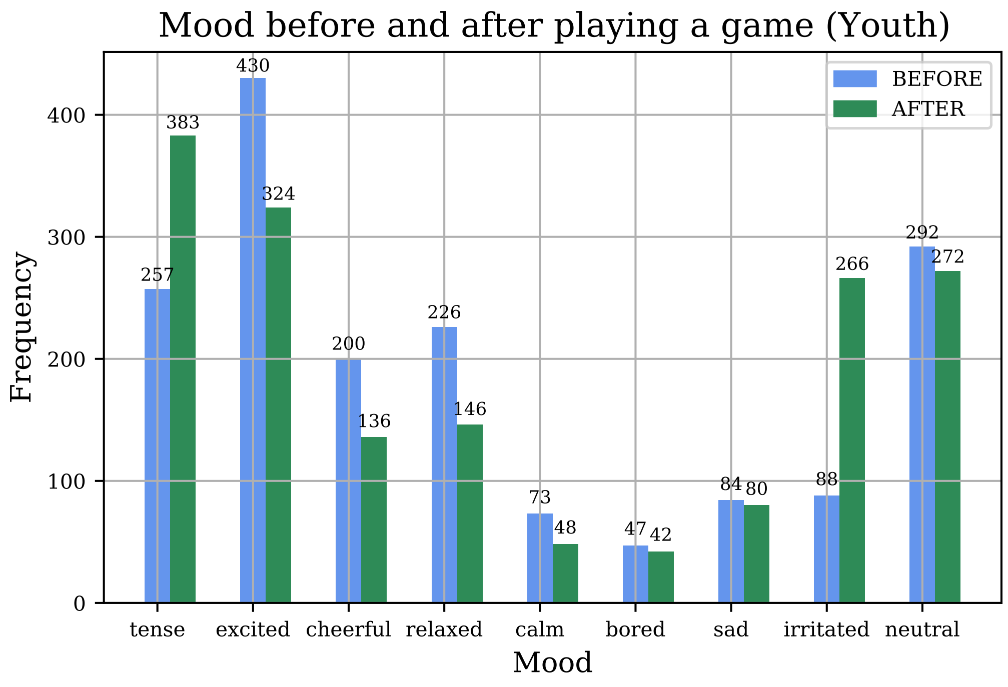


Figure 4.3: Histogram of recorded moods before and after playing Space Invaders at the youth library setup

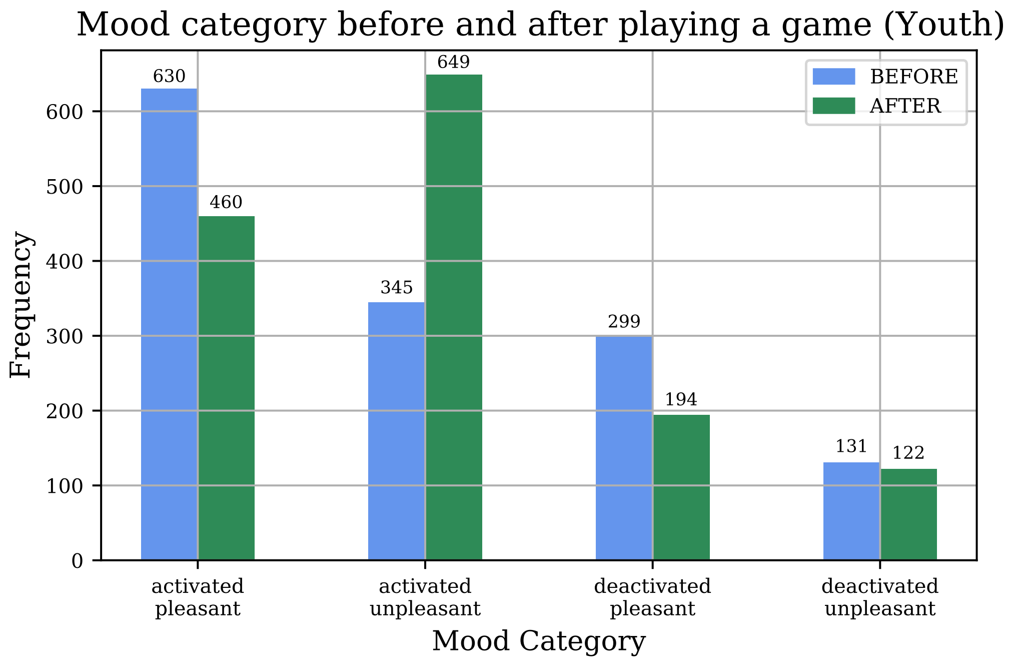


Figure 4.4: Histogram of mood quadrants before and after playing Space Invaders at the youth library setup

Game metrics

Besides mood, game metrics were also analysed. In a first analysis effort, histograms of each metric at the end of the game were made for each mood. Meaning that for each recorded game, the values of the metrics at the end of the game were used and linked with the corresponding pick-a-mood quadrant. These histograms were not easy to compare at a glance, thus the choice for violin plots was made. An example of such a violin plot for achieved scores can be seen in figure 4.5. In this plot, the distribution of different scores obtained is shown.

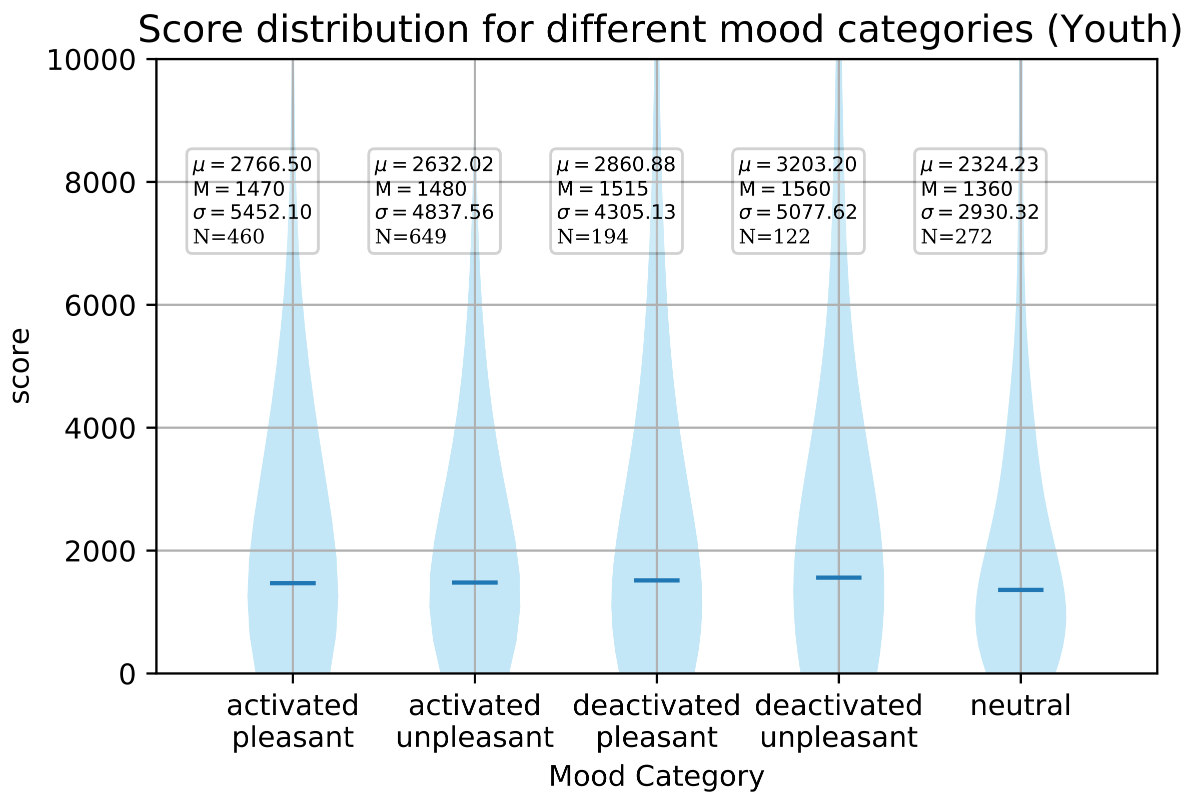


Figure 4.5: Distribution of score obtained in Space Invaders for different mood quadrant at the youth library setup

It is observed that the different quadrants do not have a big difference in distribution and have a very large spread. However, mean and median scores are a little higher for deactivated people. This is presumably due to deactivated people having easier, boring games in which they can easily get a high score without much trouble. This goes in hand by the fact that the mean and median amount of enemies killed is also higher by about five percent.

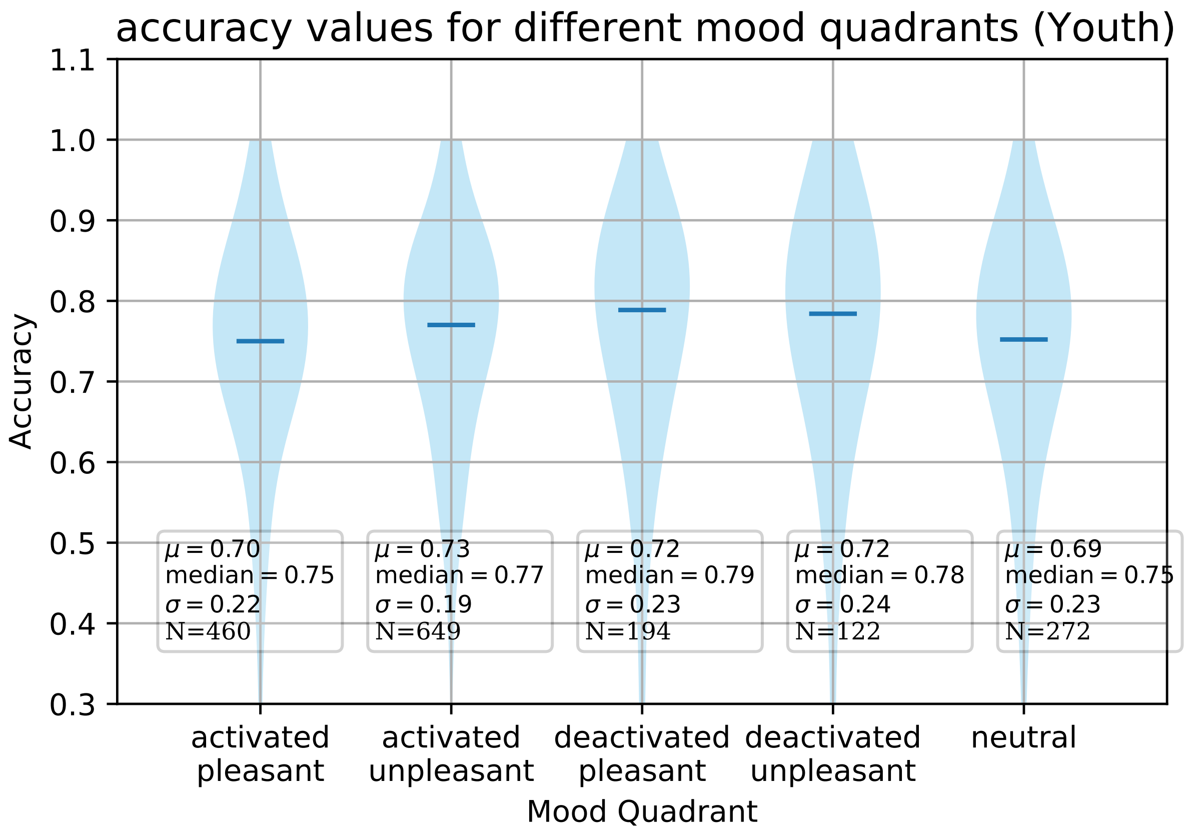


Figure 4.6: Distribution of accuracy (enemies killed / shots fired) obtained in Space Invaders for different mood quadrant at the youth library setup

Looking at the mean accuracy for the youth dataset for the different mood quadrants, similar results are obtained. No clear differentiation between quadrants exists, due to the large spread on these values.

These metric analyses do not show a relationship between metrics and recorded moods only some minor differences were observed. More metric analyses were done on other metrics such as average shots fired by the player, average lives lost while playing, round reached and more. All of them displayed no clear differences between different mood quadrants.

In a second analysis step, the metrics were thus looked at over time, too see if any patterns existed that were characteristic of a certain mood or quadrant. Perhaps bored players would have a similar pattern to activated players at first, but then lose interest as the game progressed. Nevertheless, similar results were obtained. Quadrants had similar mean and spread on different game metrics over time.

These metrics analysis lose a lot of detail of player characteristics due to each category having a wide spread of characteristic values for metrics over time. A third analysis approach was thus done that differentiated these metrics in the first order, over time and then averaged. To see how quickly the metrics changed over time, instead of their actual value. This could show certain patterns of activity for each mood category. An activated player, for example, could have a more consistent score gain, while deactivated players may more often have times of inactivity. A graph of such characteristics can be found in figure 4.7 and 4.8, showing the differentiated score for different mood categories.

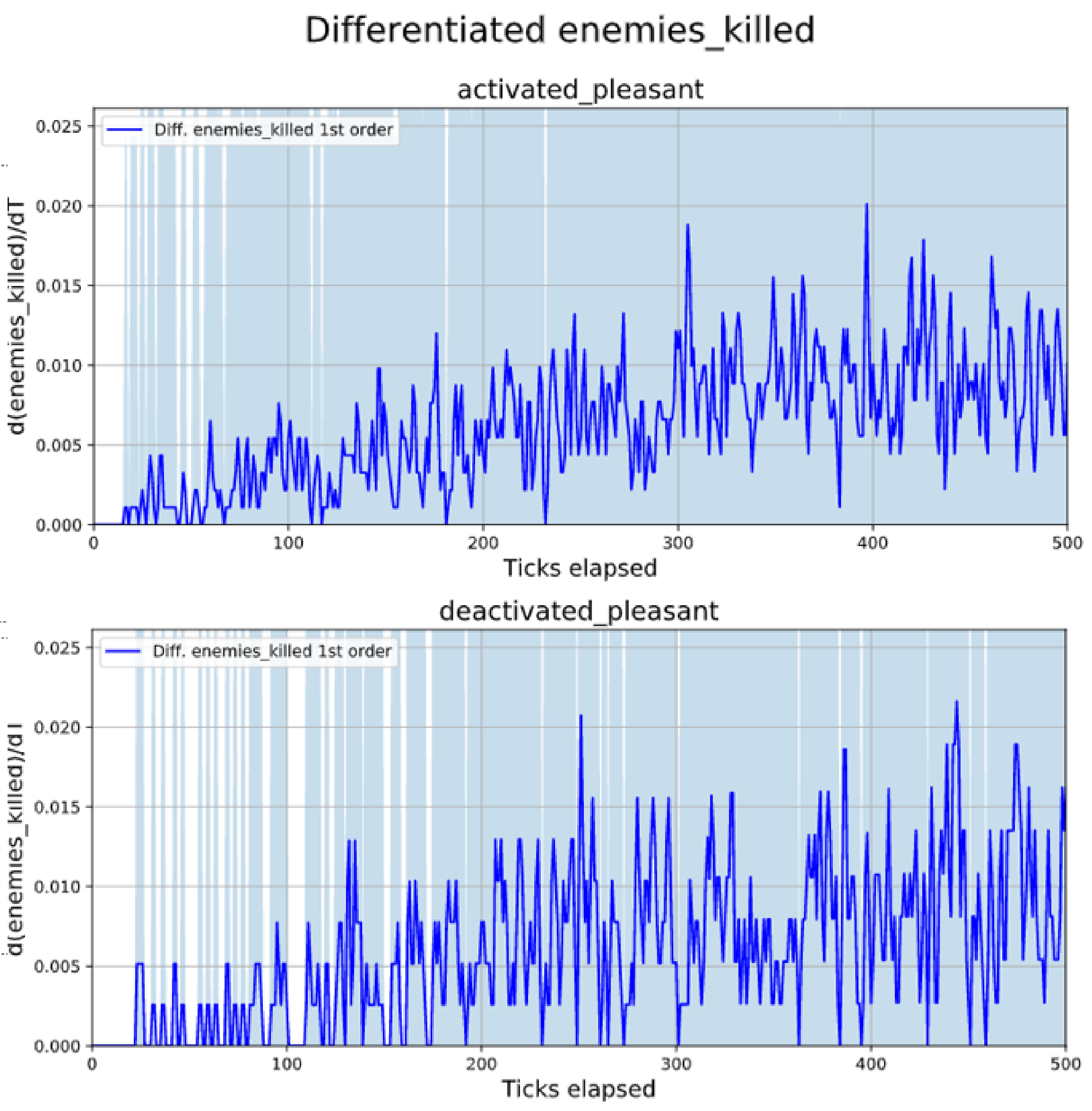


Figure 4.7: Mean differentiated amount of enemies killed for activated and deactivated pleasant mood categories

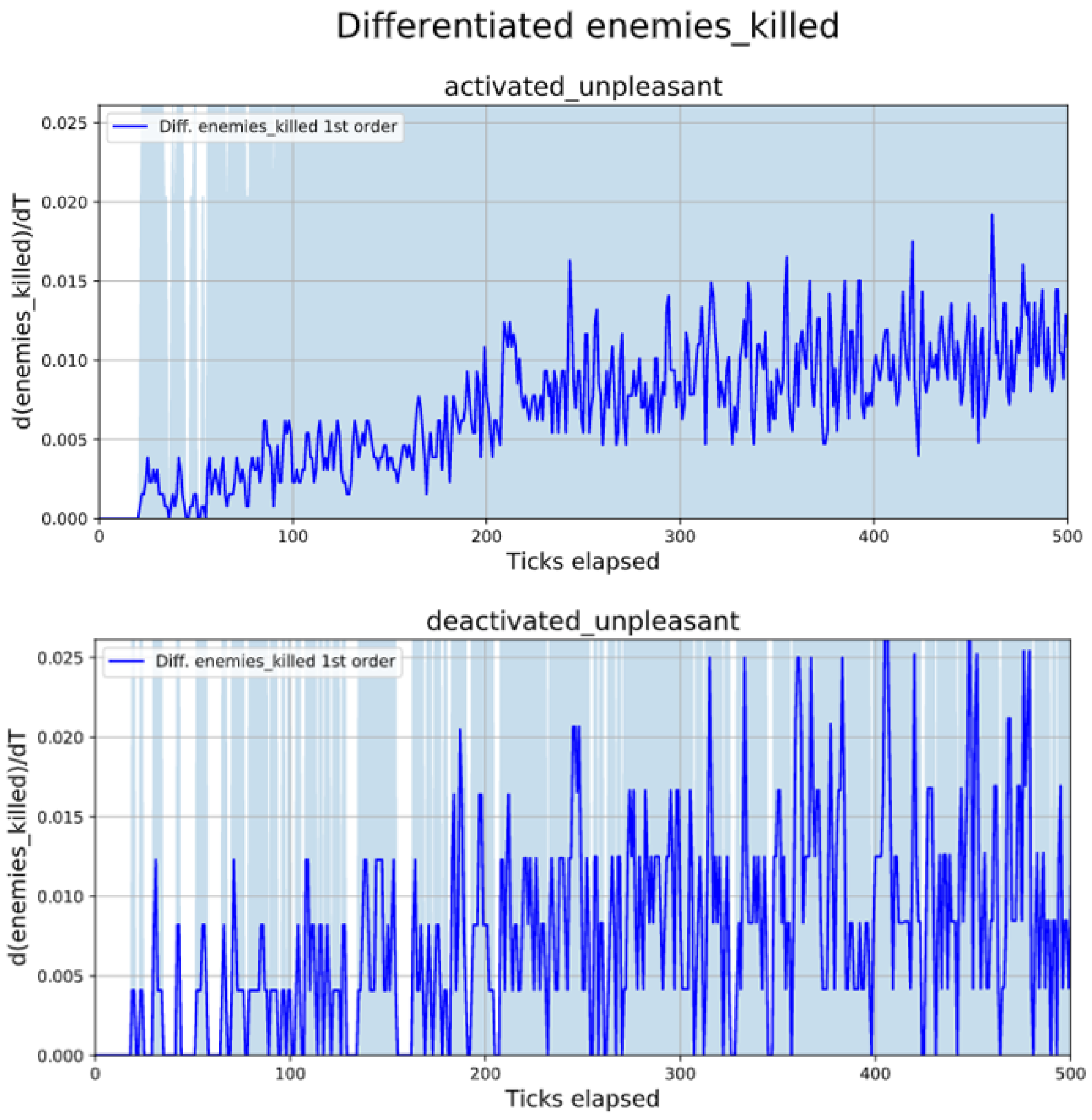


Figure 4.8: Mean differentiated amount of enemies killed for activated and deactivated unpleasant mood categories

Looking at the mean value for differentiated enemies killed, for each category, some differences can be observed that comply with the above hypothesis. However, the spread on the values is again large and noisy.

4.3.2 Conclusion

The overall conclusion of the human data collection is that capturing subjective emotions like a mood in an unsupervised environment is difficult. Besides that, trying to relate game data to these noisy moods proved even more difficult and showed no significant results. Even though participants were only asked three questions that are answered with one press of a button, lots of noise on the answers still existed. This was made apparent by the large portion of games having their recorded moods being the default selected ones. Even with some preprocessing clear relationships between recorded post-game mood and player game data failed to show.

4.4 Artificial Data

Generating artificial data from artificial players can be more reliable and can allow a relationship between game metrics and player enjoyment to exist, as different types of AI's will perform quite differently and can have different preferences programmed into them.

4.4.1 AI personas

Each AI persona is a rule-based player that has its characteristics, play-style and preference for certain game difficulties. Instead of having a before and after playing mood like human players, the AI personas will determine if they like or do not like a game according to their own programmed rules. These rules are general, meaning they can be achieved with any type of difficulty, as long as it suits their condition of liking it. Each persona, beginner, safe or advanced has their condition of liking game.

Beginner

The first type of AI that was implemented was the beginner persona. This rule-based AI is simple. It looks for a random enemy and tries to shoot it, not looking for the closest one. The beginner AI is ignorant towards using cover to for enemy shots and does not know to dodge enemy shots that are heading towards it. It is hypothesised that this AI will have the lowest scores among all personas because of its simplistic nature.

For the beginner AI, the condition to like a game was set to killing at least ten enemies and achieving a score equal or higher than 500.

Safe

A second type called the safe player was also implemented. This AI represents a more experienced player, one who actively tries to use cover in the game. This agent tries to find the nearest cover and hide behind it. Following a found cover, the agent peeks out of the cover, fires a shot and goes back. The safe player has a high probability of trying to dodge enemy bullets.

The safe AI has more work to fulfil its condition for liking a game than the beginner AI, it will like a game when: the game is won, achieving at least a score of 3000 points. The score condition is to prevent the AI from liking games that are too easy in future experiments, as the DDA system could generate games with few enemies for example. The reason for this condition overall is to represent a player who likes to win his games consistently for enjoyment.

Advanced

A final type, the advanced player, was implemented. This persona represents a player that has an aggressive, risky play-style, trying to kill as many enemies as quickly as possible, without trying to use covers. This player has a high probability of dodging an enemy bullet and will also avoid shooting his cover.

The advanced AI has a rating that is geared towards excitement. The AI will like a game if he has lost a life while playing and achieved a score of at least 2000 points. In this way, the advanced AI wants to avoid playing games that are too easy for him.

Links to a short video displaying each type of AI playing are available in the appendix 8.

4.4.2 Types of Data

The same data was to be collected as for the human data, except the mood recordings that are replaced by the like or dislike for a difficulty. An overview of the difficulty and player game metrics is given in the previous section 4.2.

4.4.3 Analysis

Each AI was running for 600 games each with a randomly initialised difficulty in the same manner of the human player data. Violin plots were made for each game metric. A violin plot of the mean achieved scores overall 600 games, per persona is shown in figure 4.9. The advanced AI has the highest overall score, followed by the safe player, followed by the beginner player.

This was not what was expected. This is because a player gets a dual-shot, firing two bullets with each shot, upgrade after a certain amount of time. This synergizes well with the risky play-style, as the AI has a greater chance of hitting two enemies at once then the safe AI player who peeks out the cover. The rather large spread on values is due to some of the randomised difficulties being very hard to play and some very easy to play.

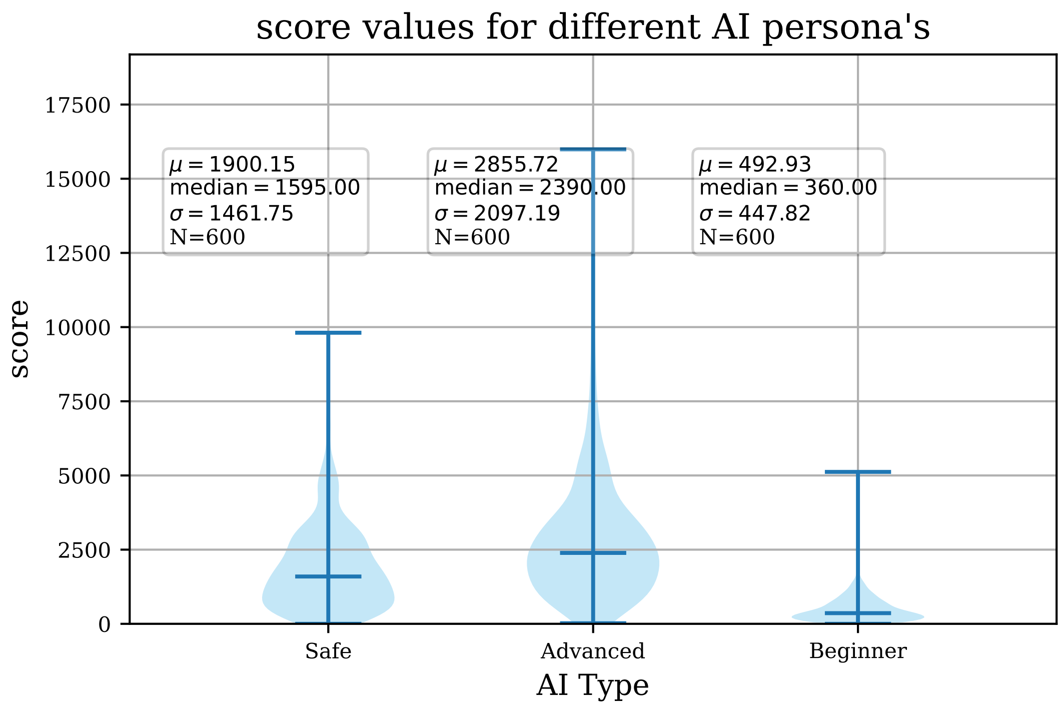


Figure 4.9: Distribution of score obtained in Space Invaders for the AI personas

When looking at the accuracy of the different personas a similar story is told, again the advanced performs best, then the safe AI, followed by the beginner. This gives more insight into why the advanced AI performed better, its risky play-style of avoiding the use of cover, to shoot more accurately, pays off more.

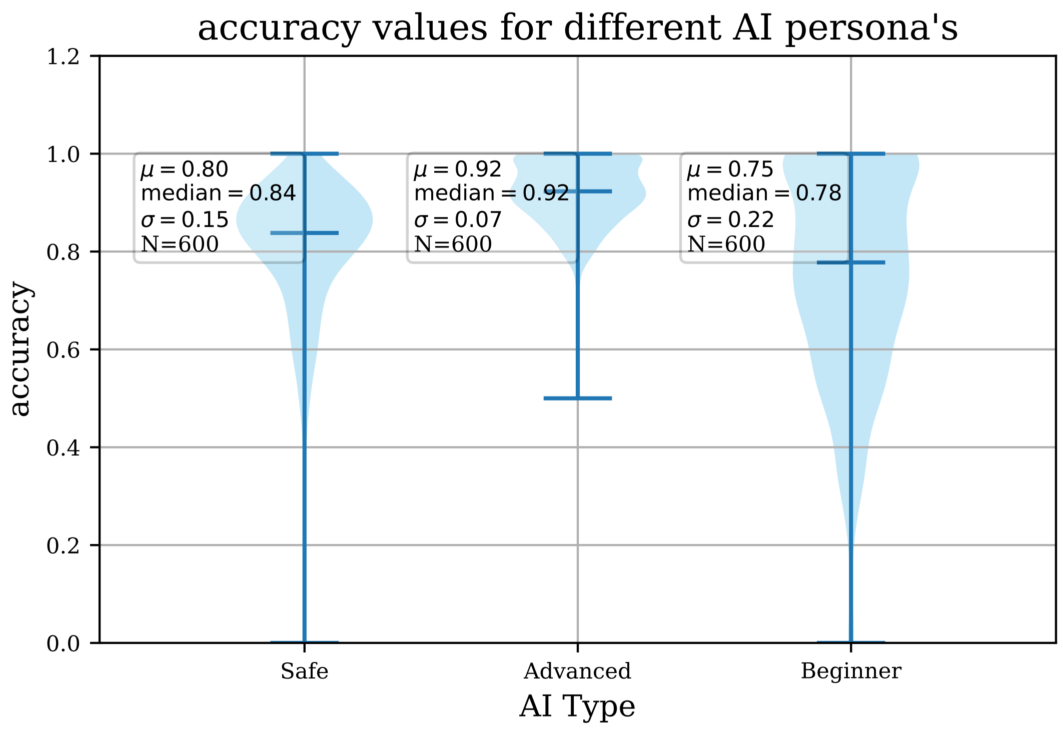


Figure 4.10: Distribution of accuracies (enemies killed / shots fired) obtained in Space Invaders for the AI personas

4.4.4 Conclusion

The analysis of the artificial data shows the clear distinction between the implemented AI personas. Each persona has certain characteristics such as mean achieved score and accuracy. Spread on these values is far lower than the human counterpart, as is expected because the AI's act consistently. This gives a clear opportunity to try and optimise the game difficulty for each persona and validate our DDA system in this way.

5 DDA System

In what follows is an explanation of the general architecture used by the DDA system.

5.1 Architecture

Diagram 5.1 summarises the architecture. To start, a player is playing a game of Space Invaders on a certain difficulty level. This player produces game metrics while playing, that are dependent on this difficulty. Playing well will give metrics that indicate good performance and vice versa. These metrics along with the played difficulty is passed to the DQN and that DQN is then responsible for estimating the best suitable difficulty for this player. Then, a certain exploration policy is used to determine whether or not the system tries a random difficulty or the current best option. Trying random difficulties has the goal of finding an even better fitting difficulty that has not been tried yet or explored enough.

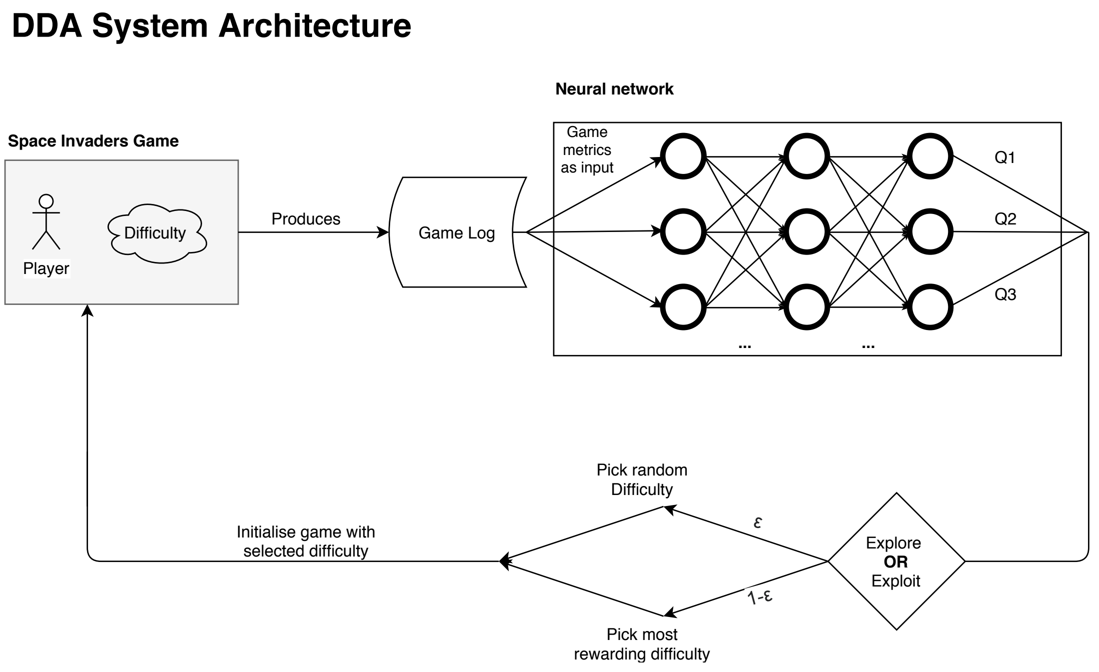


Figure 5.1: Dynamic Difficulty Adjustment system architecture, the neural network enables the system to predict what difficulty fits best to a player by using a game log that represents players their performance

This diagram does not contain the training or learning aspect of the neural network, which will be explained in section 5.1.3.

5.1.1 State and action space

The observed state consists of several game metrics that were captured after a game was played along with the different difficulty parameters that were used. They provide a summary of how a player performed on a certain game with a certain difficulty.

The action space consists of a discrete number of difficulties. This discrete number of difficulties results in all possible combinations of difficulty parameter values for the parameter that is made adaptable.

Selecting a difficulty for a certain player uses the neural network as a function approximator for the different Q-values associated with each possible state-action pair. Following an -greedy policy then determines whether or not the best assumed suitable difficulty for the player will be picked, exploiting current knowledge, or a random difficulty will be picked to perform exploration in the action space.

5.1.2 Reward

The reward the DDA system receives depends on the player his liking of the game, and thus the difficulty that was picked. When the player likes a game, the agent is rewarded and vice versa. The condition of liking the game its difficulty is bound to each player. This condition can be for example bound to the player achieving a certain score or killing a certain amount of enemies before dying.

Replay memory

The replay memory is a large memory containing all experienced sars' (state, action, reward, state') transitions by the DDA System. It is used by the neural network for training and thus learning how to accurately estimate Q-values. The original implementation of Deep-Q-learning[4] uses a randomly sampled batch of this replay memory to train the neural network.

The use of batch sampling the replay memory in Deep-Q-Learning is mainly to help avoid the instability of the neural network its predictions of the Q-values. Its value mainly comes from providing more independent learning samples from the network and thus avoiding using correlated data. This random sampling mechanism is more applicable to precise control problems instead of applying DDA though. Because of the nature of DDA, meaning that each action corresponds with a played game, values are already much less correlated since they are whole games being played. Secondly, running an entire game of Space Invaders is an expensive computational procedure because of the amount of time it takes to complete. Samples in the replay memory are thus expensive to come by and not using every one of them is illogical. The implementation for this DDA system thus uses all available samples in the replay memory for training.

5.1.3 Learning

The DDA system does not know beforehand what type of difficulties a player likes and needs to learn this by experience. The system learns to estimate the Q-values associated with each possible action and state, where each action is some difficulty. The neural network uses a large memory, called the replay memory to train itself. The neural network uses the stored transitions in the replay memory to train itself in a supervised way. Each transition is used as an input-output pair for the network to learn with. It does this by comparing the provided output with the network its output for that particular input. A loss function is then used to determine how different these outputs are from each other. Here Adam[23] was used, a first-order gradient-based optimisation algorithm. The network edges their weights and network nodes their biases are then adjusted in a way that minimises this loss.

5.1.4 Exploration

Exploration is, like for all RL problems, an important point. There must be enough exploration done to explore the action space, meaning all the possible difficulties for a player. If there is not enough exploration, the system could get stuck in a local minima solution, where other difficulties can be better suited but are not evaluated. Nevertheless, after a certain time, exploration also has to change into exploitation, where the best-suited difficulty at the time of training is picked by the DDA system consistently. The system must decay its exploration over time to avoid losing many possible rewards. A balance between exploration and exploitation must be evaluated and is dependent on how many possible difficulties there are in the action space of the DDA system. More difficulties result in a bigger action space and require more exploration to be done.

6 Results

The following table summarises the conducted experiments and the corresponding difficulty parameters that were varied for each.

Experiment	**Enemy shoot interval**	**Player lives**	**Enemy movement speed**	**Enemy movement interval**	**Action Space size**
1	x				3
2	x	x			9
3	x	x	x	x	81

Table 6.1: Experiment summary: Overview of different difficulty parameters that were made adaptable with corresponding action space size

6.1 Experiment One: Varying the enemy shoot interval

In a first experiment, the ability of the DDA system to match the right difficulty with the right AI persona was tested. One difficulty parameter was varied, the enemy shoot interval. The interval is the number of game ticks between a random enemy shooting one bullet. This could make the game challenging with a small interval, as the enemies would shoot often. On the other hand, a large interval means a lot of time between enemy shots, making easy games.

It is hypothesised that both the beginner and safe AI will like the easiest difficulties the most, as they increase the chances of fulfilling their like condition. For the advanced AI though, the shoot interval of the easiest difficulty may be too low for him to consistently lose lives while playing, a part of his like condition.

As a first result, each persona played 300 games on each possible difficulty, to see which difficulty is liked by which persona. This is displayed in figure 6.1. The corresponding difficulty parameter name and value is shown on the x-axis. These results serve as a validation tool to compare the difficulties the DDA system picks to the most liked difficulties for each persona.

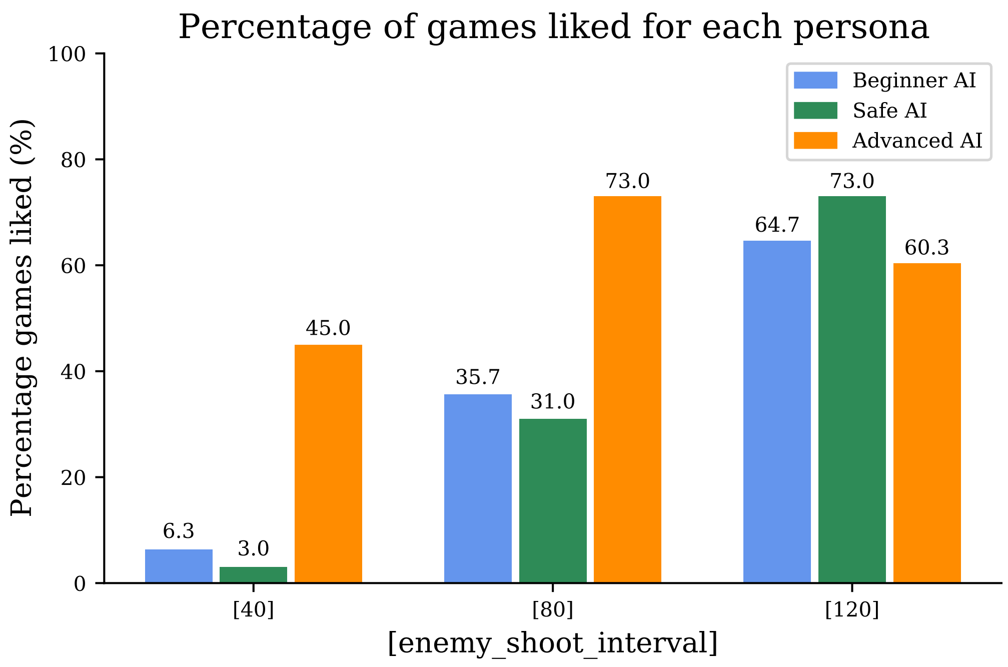


Figure 6.1: Results of each difficulty being played 300 times by each persona, ordered by increasing enemy shoot interval, meaning decreasing difficulty. Show the beginner and safe AI preferring the easiest difficulty, the advanced the middle one.

The preference of the beginner AI goes towards the easier difficulty, as expected as they give the biggest chance of the AI achieving its like condition. The same goes for the safe AI, it can win the game 73 % of the time for the easiest difficulty. The advanced AI has more interesting results. The hardest difficulty, having the enemy shoot interval equal to 40, is a bit too hard for the player. The easiest is not much of a challenge then. His main preference lies in the middle difficulty, showing that the advanced AI likes something in between a challenging and easy game, as designed.

After these results, the DDA system was tasked, for each persona, to match the right difficulty with the right persona. The DDA system was trained over 300 games and each run was repeated for three times. The resulting mean Q-values of each action, corresponding to one difficulty, throughout this training can be seen in figure 6.2.

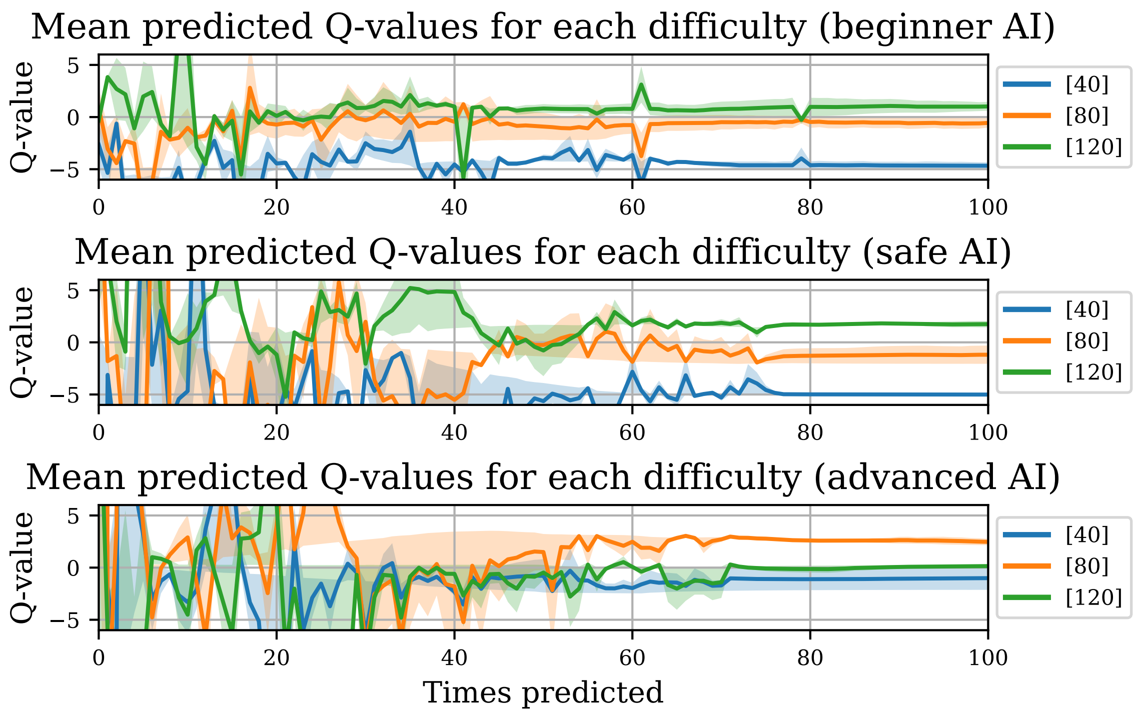


Figure 6.2: Mean estimated Q-values for each persona for experiment one. Correctly reflects the likeability for different difficulties for different personas i, figure 6.1.

It can be seen that the DQN can correctly estimate Q-values as they correspond with the most preferred difficulty for each persona. The beginner AI, for example, likes the easiest difficulty the most, this validates the Q-value of this difficulty as an action being the highest. The same goes for the safe and advanced AI. The network is also able to differentiate the two rewarding actions for the advanced AI, as the middle difficulty gets the highest Q-value, which again corresponds with the results of figure 6.2. Difficulties that are liked less than half of the time played also have a logical negative Q-value. The spread on these mean Q values for each state-action pair is also quite small, which shows that the network is behaving consistently across multiple runs, even with the small random aspects of performance (accuracy, dodging of enemy bullets) of each AI persona playing.

Looking at the like rates for the last hundred games played in table 6.2, the difference between the percentage of last hundred games liked and the highest like percentage of any difficulty for each persona is small. Meaning that the personas are indeed presented with games that match their skill.

	Served by DDA	**Validation**
Type AI	**% of last 100 games liked**	**Highest like % of any difficulty**
beginner	64.6	64.7
safe	71.4	73.0
advanced	70.7	73.0

Table 6.2: Percentage of last hundred served games by DDA system liked by each persona for experiment one. A comparison can be made to the highest like percentage of any difficulty for each persona respectively. Shows that the DDA system is performing adequately as the difference between the two metrics is small.

6.2 Experiment Two: Varying the enemy shoot interval and amount of lives

Having only three difficulties in the action space of the first experiment is quite limited. Scaling up things a bit, in a second experiment, two difficulty parameters were now adaptable at the same time, resulting in nine different difficulties. For this, again the enemy shoot interval had three possible values. Besides that, the amount of starting lives a player receives also had three possible values. To give the safe AI better chances at winning the game, more lives can help. Same goes for helping the beginner and advanced AI achieve their like condition.

Each persona played each possible difficulty 300 times to see which difficulty suits which persona the most. The resulting histogram of this can be found in figure 6.3.

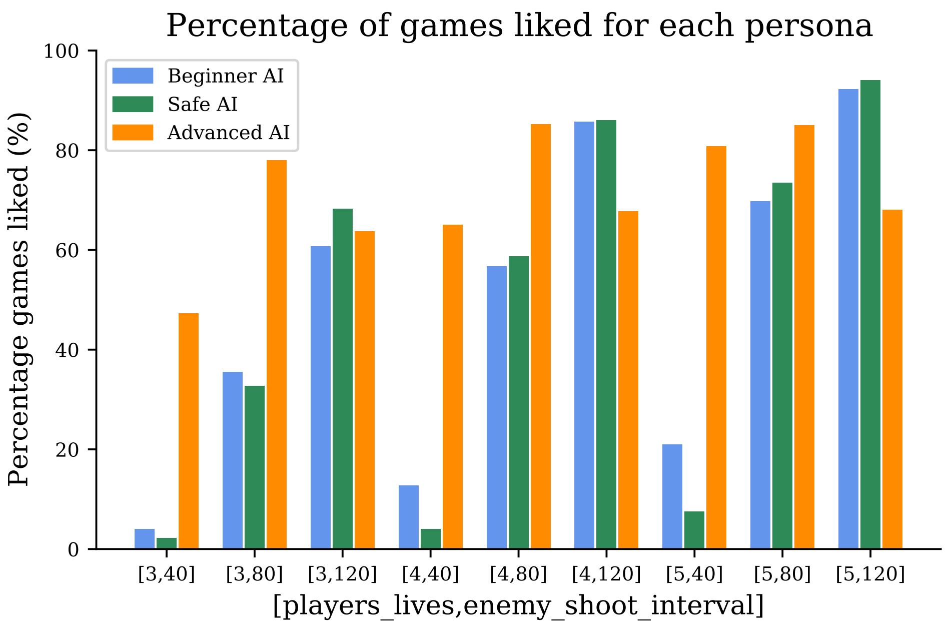


Figure 6.3: Results of each difficulty being played 300 times by each persona for experiment two. Shows the enemy shoot interval having the biggest impact on the likeability of each difficulty. Beginner and safe AI prefer easier difficulties, the advanced AI prefers something in between easy and hard.

The enemy shoot interval has the greatest impact on the desirability of certain difficulties, as it is the primary factor in allowing AI's to achieve their like conditions. The figure also shows the advanced AI likes the most games, followed by the beginner and then by the safe AI. Increasing the number of lives has an overall positive impact on the beginner and safe AI their general experience, as it increases the chances of achieving their goals. The amount of lives has a smaller impact on the advanced AI, as it only really increases its liking on the difficulties where the enemy shoot interval is lowest.

The DDA system was then tasked to find suitable difficulties for each of the personas in separate. In figure 6.4 it can be observed that the network is successfully able to determine the best difficulty suited for the beginner AI, with a relatively small spread on values. The spread on values gets larger as the mean Q-values are closer to zero. This corresponds with difficulties that have a close to 50 % chance of being liked, resulting in a bigger spread. The predicted Q-values correctly reflect the percentage an AI is to like a difficulty. As the easiest difficulty has the largest estimated Q-value and the hardest difficulty the worst.

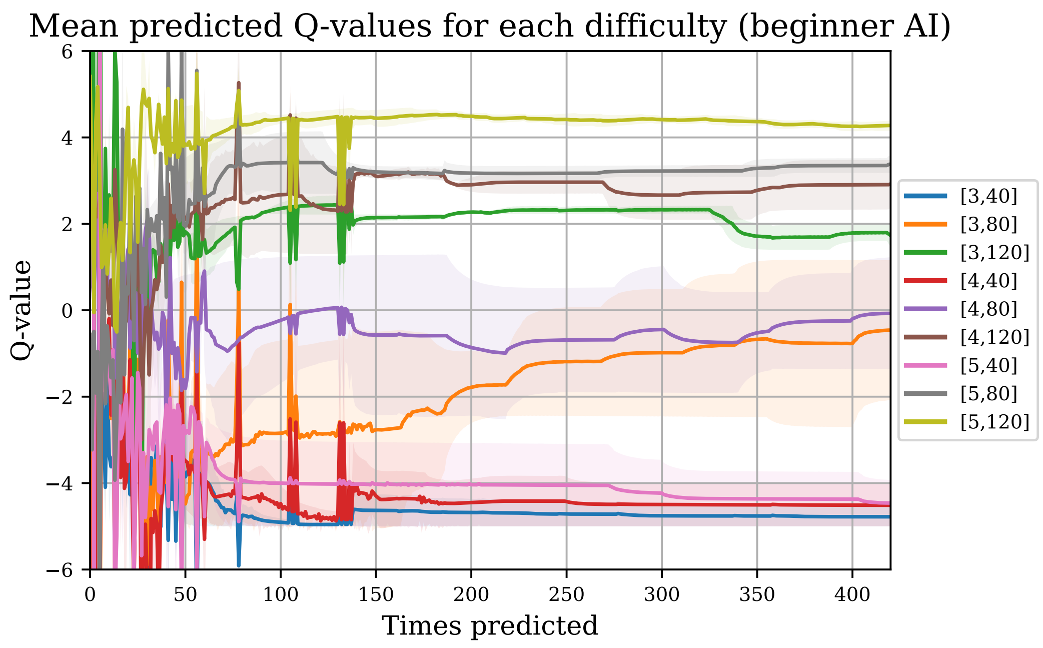


Figure 6.4: Mean predicted Q-values for each difficulty for the beginner AI for experiment two. These Q-values correspond correctly with the likeability of each difficulty. The easier the difficulty, the higher the estimated Q-value.

When the network was being trained with a safe AI playing, the three difficulties that gained the better Q-values were the difficulties with a slow shooting enemy interval, which is validated by 6.3. The difficulties that have a low chance of being liked get a corresponding low Q-value. These are the difficulties with a fast shooting interval, as these are most difficult. As the penalty for the DDA system for a disliked difficulty is -5, naturally these values converge at this number.

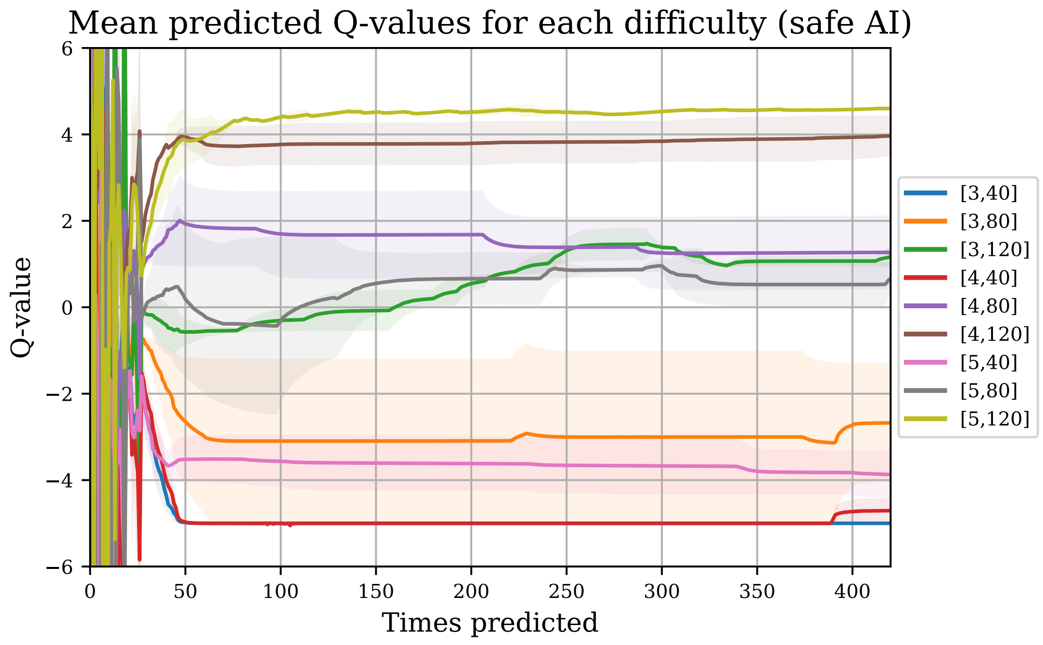


Figure 6.5: Mean predicted Q-values for each difficulty played for the safe AI for experiment two. These Q-values correspond correctly with the likeability of each difficulty. The easier the difficulty, the higher the estimated Q-value.

For the advanced AI, similar results are obtained. Because the advanced AI generally likes most difficulties, as seen in figure 6.3, all Q-values are above zero, which is a correct estimation. The Q-values for the difficulties that provide some challenge have the highest predicted Q-values, where a clear preference is seen for difficulties with an enemy shoot interval of eighty, as the three highest mean Q-values all have this parameter value.

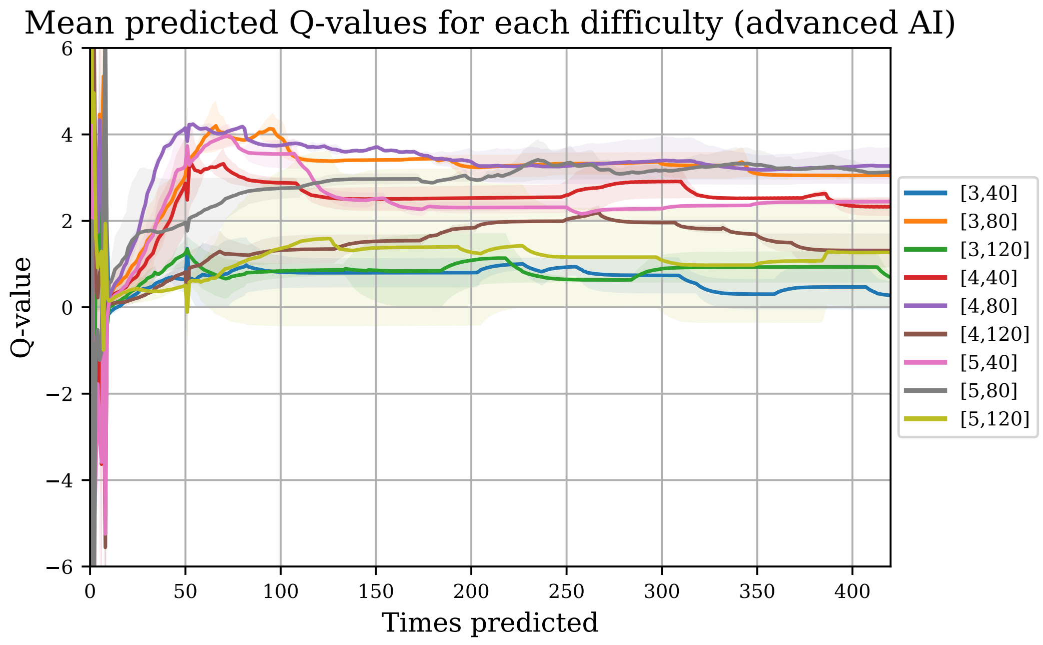


Figure 6.6: Mean predicted Q-values for each difficulty for the advanced AI for experiment two. These Q-values correspond correctly with the likeability of each difficulty. As the advanced AI likes all difficulties above 50 %, all estimated Q-values are positive.

All these Q-values generally correspond with the probability of it being liked in a validation play setting, this is the desired result and allows the network to pick suitable difficulties for each AI. In table 6.3 the like rates for each persona over the last hundred games served by the DDA system are presented. They show the DDA system performing well, as there is not a big difference between the percentage of the last hundred games liked and the percentage of the most liked difficulty overall.

	Served by DDA	**Validation**
Type AI	**% of last 100 games liked**	**Highest like % of any difficulty**
beginner	89.7	92.2
safe	87.7	94.0
advanced	83.7	85.2

Table 6.3: Percentage of last hundred served games by DDA system liked by each persona for experiment two. A comparison can be made to the highest like percentage of any difficulty for each persona respectively. Shows that the DDA system is performing adequately as the difference between the two metrics is small.

6.3 Experiment Three: Varying four different difficulty parameters

The action space was again extended. Four difficulty parameters were now made adaptable at once. Besides the number of lives and enemy shoot interval, now the movement speed and movement interval of the enemies was adaptable as well. This gives a lot of diversity in possible game difficulties, as these having a defining impact on the pace of the game. The enemy movement interval determines the time between enemy movements, meaning a lower value makes for quicker moving enemies and high value for slower enemies. The enemy movement speed, on the other hand, determines how much distance is travelled in the horizontal direction between each interval. A higher movement speed means that the aliens require less time to travel from screen edge to screen edge horizontally and thus will move more quickly towards the player as they move vertically downwards at the edge of the screen.

The experiment started with having each AI persona play each difficulty one hundred times. To see what difficulties are liked by each AI. Plotting all the different Q-values and difficulties like before would result in dense plots, so, the top and bottom three mean estimated Q-values are displayed for the different personas, along with their respective like percentage that serves as a validation tool. This gives us an easy and quick to validate if a highly liked corresponds with a high estimated Q-value, as desired. The difficulty column shows what difficulty parameters were used. For this experiment, they mean, in this order, the number of lives a player starts with, followed by the enemy shoot interval, followed by the enemy movement speed intervals over the game and finally the enemy movement step they make in the horizontal direction.

Top 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	beginner	[5, 120, [90, 60, 30, 15], 10]	99.0	4.86
2	beginner	[5, 120, [60, 40, 20, 10], 10]	98.0	4.62
3	beginner	[5, 120, [60, 40, 20, 10], 50]	99.0	4.58

Table 6.4: Top three mean Q-values estimated for each difficulty for beginner the persona for experiment three. Shows the beginner AI its preference for easy difficulties.

Table 6.4 shows that the network is able to find out highly liked difficulties and learn to estimate a high Q-value to them for the beginner persona. These difficulties are of the easiest combinations possible, which is logical since they give the beginner persona the biggest of opportunity of fulfilling its like condition. The beginner persona also likes difficulties that show a small enemy movement speed, because they are the easiest to hit. Having the extra lives, five as opposed to three or four, is also desirable for the beginner AI.

The same goes for the safe persona displayed in table 6.5. Again, because the safe persona its like condition is tied to winning the game, the easiest difficulties have the highest estimated Q-values. What is notable is that all the difficulties have a relatively small movement speed value, ten or thirty. This means that the safe AI has the least trouble hitting aliens that move in small increments. Having the extra lives, five, as opposed to three or four, is also desirable for the safe AI.

Top 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	safe	[5, 120, [90, 60, 30, 15],30]	91.0	3.67
2	safe	[5, 120, [40, 27, 13, 7], 10]	81.0	3.62
3	safe	[5, 120, [90, 60, 30, 15], 10]	81.0	3.57

Table 6.5: Top three mean Q-values estimated for each difficulty for the safe persona for experiment three. Shows the safe AI its preference for easy difficulties and slower moving enemies.

For the advanced persona, the DDA system is again able to attribute high Q-values to liked difficulties. These difficulties are not the most easiest, nor the hardest, they lie somewhere in the middle. Providing some challenge to the persona as desired. This can be observed through the enemy movement speed steps combined with the enemy shoot interval. Having the extra lives, five as opposed to three or four, is also desirable for the advanced AI.

Top 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	advanced	[5, 80, [40, 27, 13, 7], 30]	92.0	4.39
2	advanced	[5, 120, [90, 60, 30, 15], 50]	92.0	4.1
3	advanced	[5, 80, [90, 60, 30, 15], 50]	83.0	4.0

Table 6.6: Top three mean Q-values estimated for each difficulty for the advanced persona for experiment three. Shows the advanced AI its preference for difficulties that provide some challenge and are not the easiest.

The same tables for the difficulties that get the lowest Q-values assigned to them can be found below. Table 6.7, 6.8 and 6.9 all show that disliked difficulties all get a mean predicted Q-value close to minus five, as desired. For all the personas, they show the hardest difficulties being disliked the most. A combination of quick-moving enemies that shoot a lot. The dominating parameter is the enemy shoot interval. Difficulties with quick shooting enemies are thus experienced as the hardest.

Bottom 3

n	Type AI	Difficulty	% liked	Q-value (mean)
1	beginner	[3, 40, [40, 27, 13, 7], 30]	3.0	-5.0
2	beginner	[5, 40, [40, 27, 13, 7], 30]	9.0	-4.94
3	beginner	[4, 40, [40, 27, 13, 7], 30]	7.0	-4.9

Table 6.7: Bottom three mean Q-values estimated for each difficulty for the beginner persona for experiment three. Shows that hard difficulties correctly get a highly negative value attributed to them.

Bottom 3

n	Type AI	Difficulty	% liked	Q-value (mean)
1	safe	[3, 40, [60, 40, 20, 10], 30]	1.0	-5.04
2	safe	[4, 40, [90, 60, 30, 15], 30]	1.0	-5.03
3	safe	[3, 40, [40, 27, 13, 7], 30]	1.0	-5.01

Table 6.8: Bottom three mean Q-values estimated for each difficulty for the safe persona for experiment three. Shows that hard difficulties correctly get a highly negative value attributed to them.

Bottom 3

n	Type AI	Difficulty	% liked	Q-value (mean)
1	advanced	[3, 40, [60, 40, 20, 10], 50]	10.0	-5.0
2	advanced	[4, 40, [60, 40, 20, 10], 50]	25.0	-5.0
3	advanced	[3, 40, [40, 27, 13, 7], 50]	15.0	-4.12

Table 6.9: Bottom three mean Q-values estimated for each difficulty for the advanced persona for experiment three. Shows that hard difficulties correctly get a highly negative value attributed to them.

The DDA system is thus successfully able to find suitable difficulties for each persona, with a large number of difficulties to choose from. This is again validated in table 6.10. Here, a bigger difference between the two metrics is observed for the safe persona. Analysing the Q-values more for the safe AI showed that the top estimated Q-values were still fluctuating and not yet converged. This means that the safe AI needs more training than the other AI's and why the like rate is a bit lower.

	Served by DDA	**Validation**
Type AI	**% of last 100 games liked**	**Highest like % of any difficulty**
beginner	93.9	99.9
safe	76.7	92.0
advanced	88.3	97.0

Table 6.10: Percentage of last hundred served games by DDA system liked by each persona for experiment three. A comparison can be made to the highest like percentage of any difficulty for each persona respectively. Shows that the DDA system is performing adequately as the difference between the two metrics is relatively small for each persona.

6.3.1 Alternative ratings

Because of the relatively simple rating system of the current AI personas, the decision was made to test the system with a more complex condition of rating the game for the safe and advanced personas. The two new ratings take into account how close the last enemy got to the player, either when the player loses the game or when he wins. For the safe AI, a new rating conditioned that the AI was to like a game if he reached the second round and that the last enemy position he observed, either before dying or finishing the game, was at least 450 pixels away. This persona is called the safe-slow persona. This was done to try and make the safe player like games that had a slower pace.

On the other hand, the new rating system for the advanced AI wanted just the opposite of the safe AI. Meaning that the AI preferred games where the last enemy got close to the player. This translates in a condition that the last enemy had to be closer than 360 pixels away. To vary the rating from the safe-slow AI, the advanced-fast AI also needed to lose two lives and get a score above 2000 while playing. This was done to encourage the advanced AI to get tense and close games, meaning fast-moving enemies. This persona is called the advanced-fast persona.

The DDA system was then tasked to find suitable difficulties with the same action space as before in experiment three. Table 6.11 shows that liked difficulties get a positive Q-value attributed to them. The safe rating persona its preference for slow games is visible through the difficulty parameters. The combination of high enemy movement intervals and slow horizontal movement speed result in slow-moving enemies as desired.

Top 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	safe-slow	[5, 120, [90, 60, 30, 15], 10]	94.0	3.89
2	safe-slow	[5, 12, [60, 40, 20, 10], 10]	95.0	3.85
3	safe-slow	[4, 80, [90, 60, 30, 15], 10]	71.0	3.18

Table 6.11: Top three mean Q-values estimated for each difficulty for the safe-slow persona for experiment three. Shows the persona its preference for slow-moving enemies.

The advanced-fast persona does not like many difficulties, in fact its highest like percentage does not go over 62 %. This is reflected in table 6.12, where the system estimates Q-values close to zero, reflecting the proximity to difficulties that are close to fifty % liked. Training for the advanced-fast persona was done longer as the system took more time to learn what difficulties fit the advanced-fast persona. The preference for fast-moving enemies for the advanced rating persona is visible through the small enemy movement intervals combined with high enemy movement speed, they result in enemies that quickly catch up to the player.

Top 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	advanced-fast	[5, 120, [90, 60, 30, 15], 50]	52.0	0.48
2	advanced-fast	[5, 80, [40, 27, 13, 7], 30]	50.0	0.17
3	advanced-fast	[5, 80, [40, 27, 13, 7], 50]	62.0	-0.27

Table 6.12: Top three mean Q-values estimated for each difficulty for the advanced-fast persona for experiment three. Shows the persona its preference for fast-moving enemies.

The bottom three mean Q-values correctly pair with difficulties that are highly disliked as seen in table 6.13 and 6.14 for the safe-slow and advanced-fast personas respectively. The difficulties that are most disliked correspond to the conditions implemented for liking a difficulty for each AI. This means that the safe rating persona dislikes fast-moving enemies and the advanced rating persona dislikes slow-moving enemies. The third lowest estimated Q-value for the advanced-fast persona shows that having an enemy shoot interval of forty makes it hard for the AI to achieve a score of 2000 and thus does not like these games.

Bottom 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	safe-slow	[4, 120, [40, 27, 13, 7], 50]	0.0	-5.03
2	safe-slow	[4, 120, [60, 40, 20, 10], 50]	3.0	-5.0
3	safe-slow	[5, 40, [40, 27, 13, 7], 50]	0.0	-5.0

Table 6.13: Bottom three mean Q-values estimated for each difficulty for the safe-slow persona for experiment three. Shows the persona its dislike for fast-moving enemies.

Bottom 3				
n	**Type AI**	**Difficulty**	**% liked**	**Q-value (mean)**
1	advanced-fast	[4, 120, [40, 27, 13, 7], 10]	0.0	-5.02
2	advanced-fast	[3, 80, [60, 40, 20, 10], 10]	0.0	-5.02
3	advanced-fast	[3, 40, [90, 60, 30, 15], 50]	1.0	-5.02

Table 6.14: Bottom three mean Q-values estimated for each difficulty for the advanced-fast persona for experiment three. Shows the persona its dislike for slow-moving enemies.

Table 6.15 shows the like rate over the last hundred games served by the DDA compared to the highest like percentage of any played difficulty as validation. The DDA system performs adequately for the safe-slow persona as the difference between the two metrics is relatively small. The system struggles more for the advanced-fast persona, as it has more limited options available for liked difficulties.

	Served by DDA	**Validation**
Type AI	**% of last 100 games liked**	**Highest like % of any difficulty**
safe-slow	87.1	95.0
advanced-fast	49.6	62.0

Table 6.15: Percentage of last hundred served games by DDA system liked by each persona for experiment three. A comparison can be made to the highest like percentage of any difficulty for each persona respectively. Shows that the network is performing well for the safe-slow persona, but not so much for the advanced-fast persona.

7 Conclusion

In this work, dynamic difficulty adjustment was applied to the game of Space Invaders using a deep reinforcement learning technique called Deep-Q-Learning. A data collection was done on human players to gain insight into relationships between player game metrics and player enjoyment. No clear relationships were found. Data collection was also done on three artificial player personas that were designed. They showed clear differences in player metric characteristics between them. Deep-Q-learning was applied in able to let the DDA system learn to pick suitable difficulties for these artificial players using game metrics as an input.

The system was tested using several experiments with increasing variability of difficulty parameters using artificial player personas that each had their liking for certain difficulties. A first experiment was done with three difficulties available for the system, where it was able to pick suitable difficulties for each persona. A second experiment had nine possible difficulties, again performing adequately for each persona. A third experiment was done with eighty-one possible difficulties, where the DDA system performed well. In this third experiment, two new personas with more advanced difficulty preferences were designed as well, the experiment showed that more training of the system is necessary when the preference for certain difficulties is limited.

In general, the system showed adequate performance for picking suitable difficulties for each persona and showed scalability in increasing the number of variable difficulty parameters.

8 Future Work

This work lends itself to a couple of different options for future work. First and foremost, a more extensive analysis of the gathered datasets could identify some key player characteristics of players and relationships between enjoyment and player metrics. This could lead to possible better state representations of games played leading to more accurate performance of the DDA system and faster convergence. Player clustering could also be performed on the gathered datasets, to identify player types using some form of machine learning. The DDA system could then use this cluster knowledge to classify a player and pick suitable difficulties for each cluster.

Secondly, a continuous action space could also be used instead of the current discrete one. This would allow the DDA system to have fine granularity in adjusting the difficulty to a player his liking. Actor-critic methods of reinforcement learning can be used to approach a continuous action space.

Another research option is the use of a single network for all different AI personas, as currently a separate neural network was used to assign difficulties for each. Preliminary experiments with a single network were performed but showed lacking performance. Multiple factors could be the cause of this, such as the need for a more complex neural network or a better input state representation. More research is thus needed.

Finally, the DDA system was only tested using artificially generated data coming from rule-based AI's. A possible option would thus be to set up an experiment comparing the enjoyment players experience when playing with or without the DDA system turned on. Results could then indicate if using DDA had a positive effect on the players their experience.

Bibliography

[1] "Likert scale." [Online]. Available: https://www.sciencedirect.com/topics/psychology/likert-scale

[2] P. Desmet, M. Vastenburg, and N. Romero, "Mood measurement with pick-a-mood: Review of current methods and design of a pictorial self-report scale," *Journal of Design Research*, vol. 14, no. 3, pp. 241–279, 2016.

[3] "Cs188: An intro to ai." [Online]. Available: http://ai.berkeley.edu/home.html

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, 02 2015.

[5] "The original 'space invaders' is a meditation on 1970s america's deepest fears," Jun 2018. [Online]. Available: https://www.smithsonianmag.com/science-nature/original-space-invaders-icon-1970s-America-180969393/

[6] K. Anderton, "The business of video games: A multi billion dollar industry [infographic]," Apr 2018. [Online]. Available: https://www.forbes.com/sites/kevinanderton/2017/04/29/the-business-of-video-games-a-multi-billion-dollar-industry-infographic/

[7] R. Koster and W. Wright, *A Theory of Fun for Game Design.* Paraglyph Press, 2004.

[8] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 429–433. [Online]. Available: https://doi.org/10.1145/1178477.1178573

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

[10] A. Zook and M. O. Riedl, "A Temporal Data-Driven Player Model for Dynamic Difficulty Adjustment," in *AIIDE*, 2012.

[11] B. Cowley, D. Charles, M. Black, and R. Hickey, “Toward an understanding of flow in video games,” *Comput. Entertain.*, vol. 6, no. 2, Jul. 2008. [Online]. Available: https://doi.org/10.1145/1371216.1371223

[12] J. Chen, “Flow in games (and everything else),” *Commun. ACM*, vol. 50, no. 4, p. 31–34, Apr. 2007. [Online]. Available: https://doi.org/10.1145/1232743.1232769

[13] M. Booth, “Ai systems of left for dead.” [Online]. Available: https://steamcdn-a.akamaihd.net/apps/valve/2009/ai_systems_of_l4d_mike_booth.pdf

[14] M. Csikszentmihalyi, *Flow: The psychology of happiness.* Random House, 2013.

[15] I. Mauss and M. Robinson, “Measures of emotion: A review,” *Cognition emotion*, vol. 23, pp. 209–237, 02 2009.

[16] M. D. Robinson, “Nihms134765.Pdf,” vol. 23, no. 2, pp. 1–23, 2009.

[17] P. Ekman, R. W. Levenson, and W. V. Friesen, “Autonomic nervous system activity distinguishes among emotions.” *Science*, vol. 221 4616, pp. 1208–10, 1983.

[18] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[19] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. P. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, “Starcraft II: A new challenge for reinforcement learning,” *CoRR*, vol. abs/1708.04782, 2017. [Online]. Available: http://arxiv.org/abs/1708.04782

[20] G. Hinton, “lecture slides lecture 6,” pp. 26–31, 2012. [Online]. Available: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[21] L. Robinson, “Creating a space invaders clone with python,” Oct 2014. [Online]. Available: https://leerob.io/blog/space-invaders-with-python

[22] P. Desmet, M. Vastenburg, and N. Romero, “Pick-a-mood manual: Pictorial self-report scale for measuring mood states,” 07 2016.

[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

Appendices

Game Flow

The following screenshots show an example flow a player goes through for playing one game.

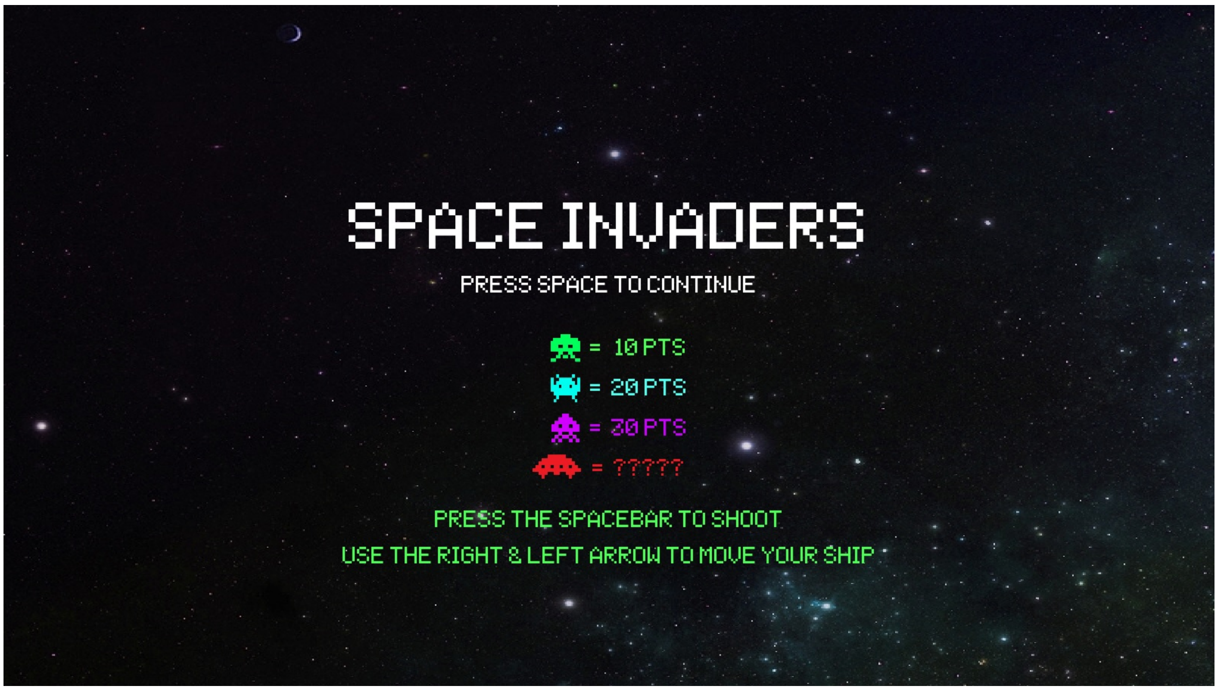
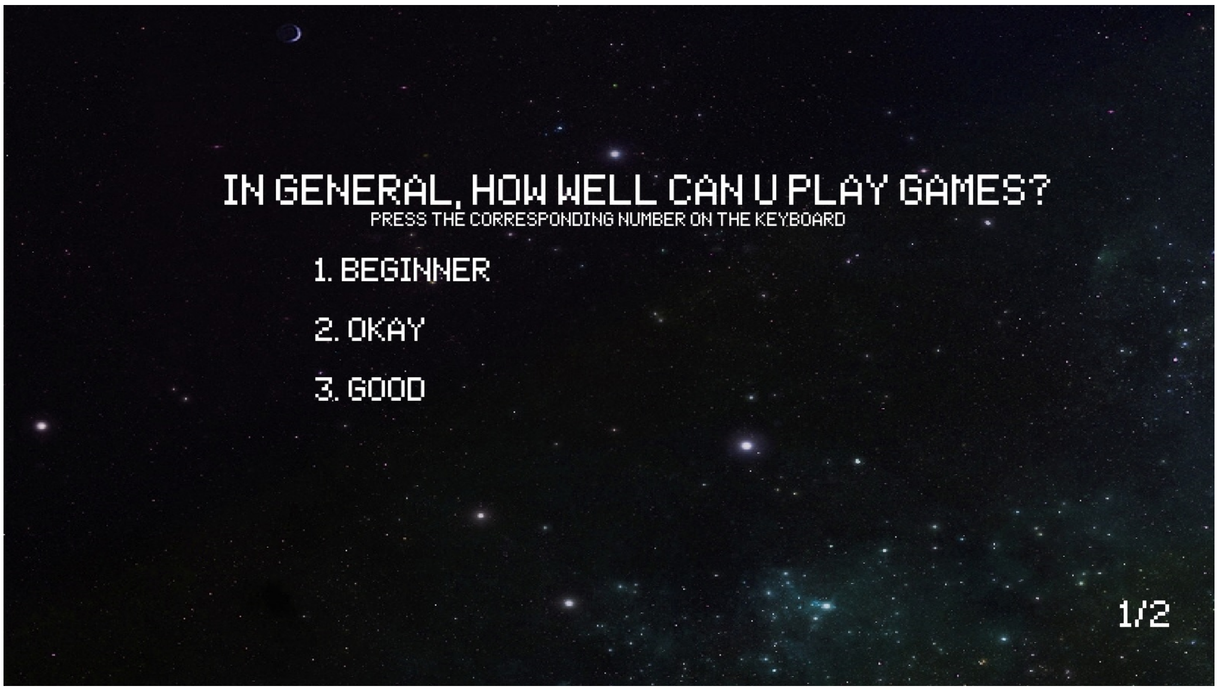

SPACE INVADERS
PRESS SPACE TO CONTINUE
= 10 PTS
= 20 PTS
= 30 PTS
= ?????
PRESS THE SPACEBAR TO SHOOT
USE THE RIGHT & LEFT ARROW TO MOVE YOUR SHIP

Figure 1: Start screen of game



IN GENERAL, HOW WELL CAN U PLAY GAMES?

PRESS THE CORRESPONDING NUMBER ON THE KEYBOARD

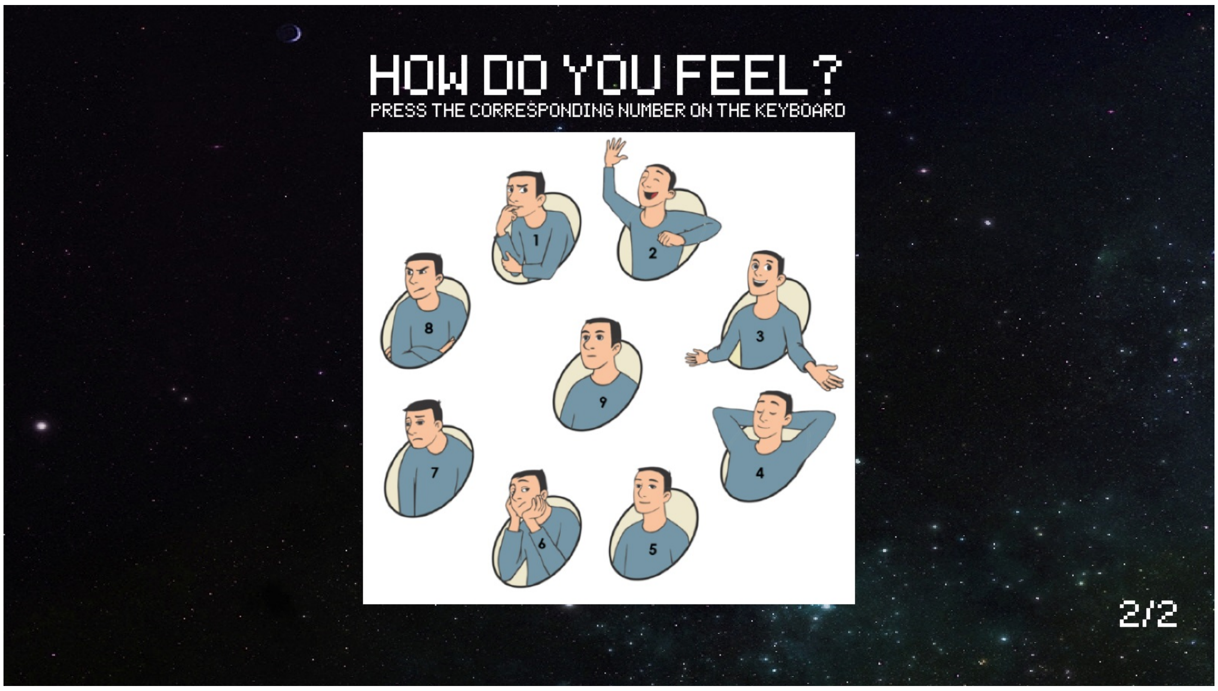
1. BEGINNER

2. OKAY

3. GOOD

1/2

Figure 2: Asking players their perceived skill level



HOW DO YOU FEEL?

PRESS THE CORRESPONDING NUMBER ON THE KEYBOARD

2/2

Figure 3: Asking players their perceived mood

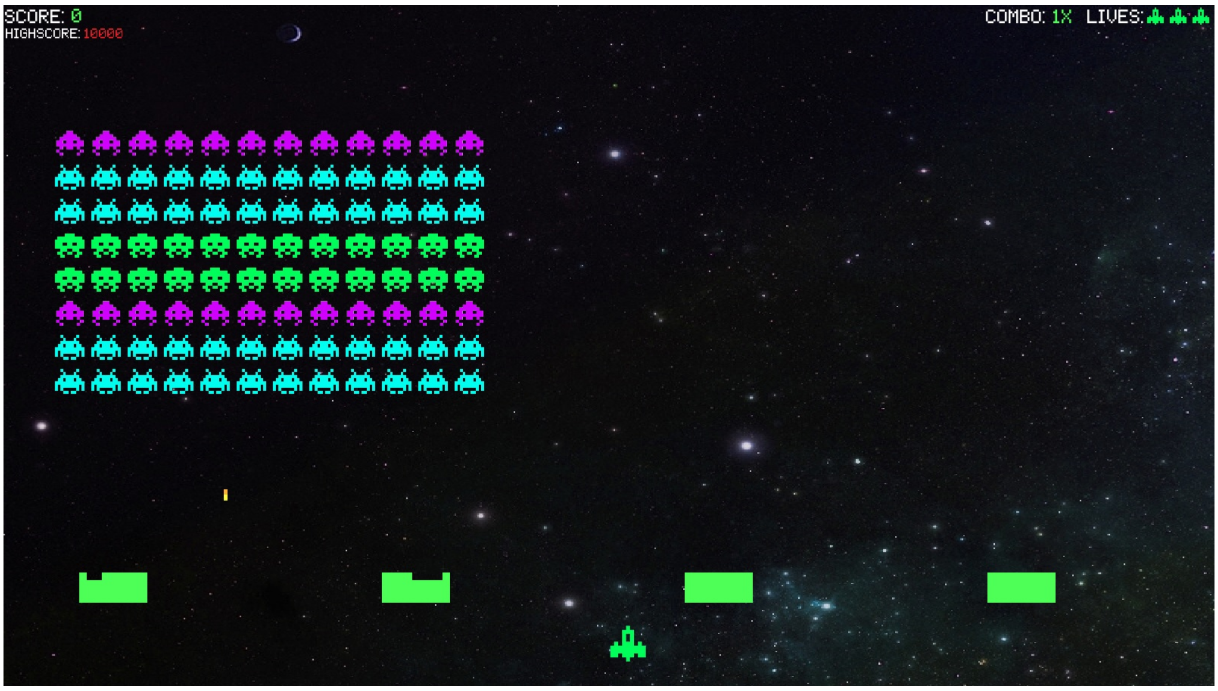


Figure 4: Example game play

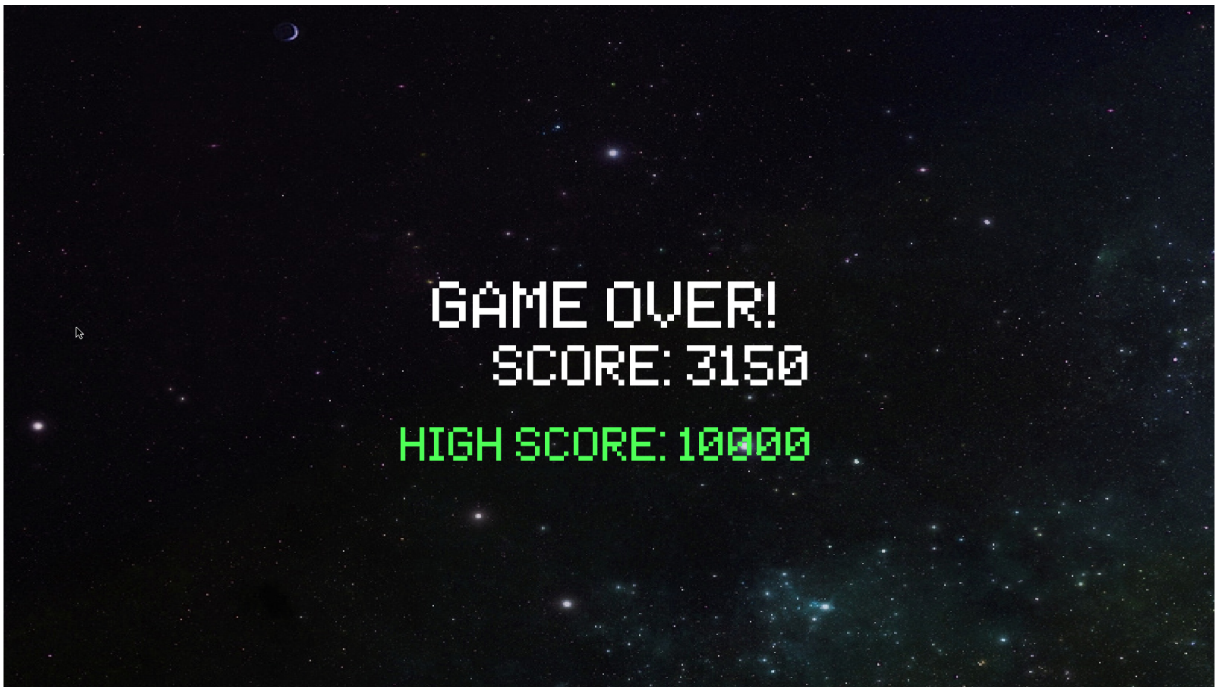


Figure 5: Game over screen

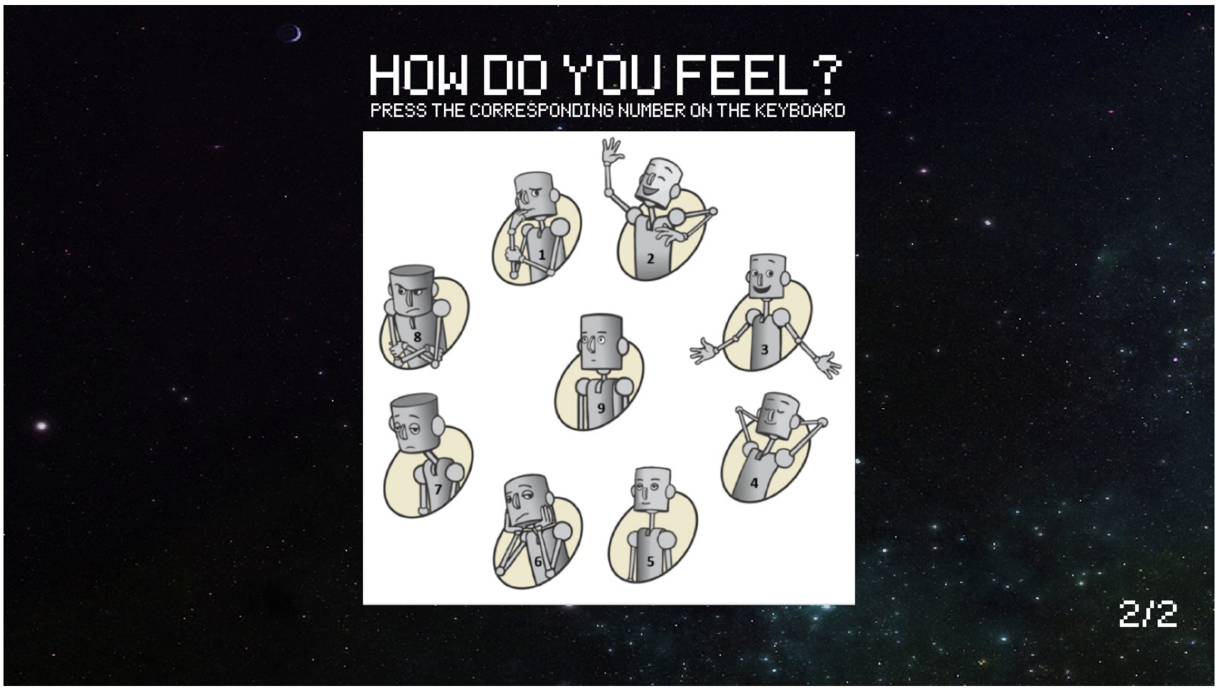


Figure 6: Asking players their perceived mood after playing

AI Gameplay Video's

Beginner AI Gameplay: `https://drive.google.com/open?id=1-VuAOgXbWRZtdRePITrrzY5MDJMe87je`
Safe AI Gameplay: `https://drive.google.com/open?id=1z9y7-5xVkAUssrsVniKI54SrNJhKo7eF`
Advanced AI Gameplay: `https://drive.google.com/open?id=1_2vk5F88Vf5vhmvRtFHdwpPVoi47lDMl`