Machine Learning Applied to Pac-Man
Final Report

David Bigas Ortega

Director: Javier Béjar Alonso
Specialisation: Computing

Barcelona School of Informatics - FIB UPC

June 2015

Acknowledgements

First and foremost, I would like to express my deep gratitude to Javier Béjar, the supervisor of my research project, for his support. He has guided me to define the aim of the project. Moreover, he has kindly clarified any questions that have come out throughout the development.

Second, I would like to thank the ten participants of the test conducted:

- Adrián Urgell
- Albert Carrión
- Angel Batlles
- Elena Blanco
- Jesús Diaz
- Joaquim Casals
- Jordi Moyés
- Lluís Alemany
- Oscar Argudo
- Roger Hernando

Third, I would like to thank Elisenda Sulleva for reviewing and checking the correct spelling, grammar and coherence in the written project, and Victor Sulleva for composing some tracks for the music and songs for the application.

Finally, I would like to thank my family for all their support and encouragement throughout my studies.

Abstract (English)

Machine learning applied to Pac-Man has been researched in several studies and in different ways, some of them focusing on the development of the Pac-Man's side and others focusing on the development of the ghosts' side. This project will evaluate how to develop a model for the ghosts based on machine learning which will adapt itself depending on the user level.

The difficulty of Pac-Man increases by the difference between Pac-Man's speed and the ghosts' speed. In addition to previous differences, it reduces the time of the ghosts' departure delay and their frightened state. In the project's version, the ghosts will take better or worse movement decisions according to the level of Pac-Man, no other attribute will be changed in the execution time.

To develop a machine learning model with these criteria, the development of another machine learning model for the role of Pac-Man will be required. This model has to be able to simulate different playing levels (beginner, medium and expert). The objective of this second development is to have enough data to reach the previous criteria. Therefore, the application of the project will include two modes, the game and the simulation.

The main target of this project is to evaluate the experience of the users in front of a game difficulty fitted to their playing level.

Abstract (Catalan)

L'aprenentatge automàtic (*machine learning*) en el Pac-Man ha estat investigat en diversos estudis i de diverses maneres, uns centrant el desenvolupament en la part del Pac-Man i altres en la part dels fantasmes. En aquest projecte s'avaluarà com desenvolupar un model de cara al rol dels fantasmes basats en aprenentatge automàtic de tal forma que el seu nivell es vegi afectat segons la forma de jugar de l'usuari actual.

La dificultat del Pac-Man s'incrementa mitjançant la diferència entre la velocitat del Pac-Man i la velocitat dels fantasmes, addicionalment va decrementant el temps de sortida dels fantasmes i el seu estat d'espantat. En la versió d'aquest projecte, els fantasmes prendran millors o pitjors moviments d'acord amb el nivell del Pac-Man, cap altre factor es veurà manipulat en temps de joc.

Per desenvolupar un model d'aprenentatge amb aquests criteris, s'haurà de desenvolupar un altre model d'aprenentatge pel rol del Pac-Man, el qual sigui capaç de simular diferents nivells de joc (principiant, intermedi i expert), amb l'objectiu de poder tenir una mostra suficientment gran per a determinar un model d'aprenentatge adient pels criteris. Per tant, l'aplicació del mateix projecte inclourà dos modes, el joc i la simulació.

L'objectiu principal d'aquest projecte és avaluar l'experiència que s'aporta als usuaris, davant d'una complexitat ajustada al seu nivell.

Abstract (Spanish)

El aprendizaje automático (*machine learning*) en Pac-Man ha estado estudiado en diversos estudios y de diferentes maneras, unas centradas en el rol de Pac-Man y otros en el rol de los fantasmas. En este proyecto se evaluará cómo desarrollar un modelo de cara al rol de los fantasmas basados en aprendizaje automático de tal manera que su nivel se vea afectado según la manera en la cual juega el usuario actual.

La dificultad de Pac-Man se incrementa mediante la diferencia entre la velocidad de Pac-Man y la velocidad de los fantasmas, adicionalmente va decrementando el tiempo de salida de los fantasmas y su estado de asustado. En la versión de este proyecto, los fantasmas tomarán mejores o peores movimientos de acuerdo con el nivel de Pac-Man, ningún otro factor se verá manipulado en tiempo de juego.

Para desarrollar un modelo de aprendizaje con estos criterios, se deberá desarrollar otro modelo de aprendizaje para el rol de Pac-Man, el cual sea capaz de simular diferentes niveles de juego (principiante, intermedio y experto), con el objetivo de poder tener una muestra suficientemente grande para determinar un modelo de aprendizaje adecuado para los criterios. Por lo tanto, la aplicación del mismo proyecto incluirá dos modos, el juego y la simulación.

El objetivo principal de este proyecto, es evaluar cual es la experiencia que se aporta a los usuarios, delante de una complejidad modelada a su nivel.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

An exhaustive analysis of the different areas related to the project will be seen through this chapter. In addition to that, the different objectives of the project will be explained. The scope and limitations will be introduced as well. Finally, a methodology and the system to validate the objectives will be shown.

1.1 Context

In this section, we are going to describe the different areas of interest related to this project. Besides, the actors that are going to be affected by this project will be presented.

1.1.1 Areas of interest

There are three main areas of interest in this project. The first one is the video game Pac-Man. The second one is Machine Learning (ML) applied to video games. The last one is the technical aspects related to the development of the project.

- **Pac-Man:** This project has the goal of developing a ML for the ghosts of the video game Pac-Man. Because of that, a platform based on this video game is required.

  The game consists in eating all dots found around a stage, while there are four ghosts with different patterns that want to catch the player. The key is to run away from the ghosts or else eat them. The ghosts can only be eaten if they are frightened and this will happen if Pac-Man is energised. Thus, Pac-Man has to take some energisers distributed all over the stage[1]. Not only should the player aim at eating all dots but the high score has to be beaten as well. All eaten elements will add points into the score. In the following list[2] there are the rules of the score:

  - Dot 10 points.
  - Energiser 50 points.
  - 1^{st} ghost 200 points.
  - 2^{nd} ghost 400 points.
  - 3^{rd} ghost 800 points.
  - 4^{th} ghost 1,600 points.

  - Cherry 100 points.
  - Strawberry 300 points.
  - Orange 500 points.
  - Apple 700 points.
  - Pineapple 1,000 points.
  - Galaxian spaceship 2,000 points.
  - Bell 3,000 points.
  - Key 5,000 points.

  The difficulty that this game entails is the reduction of some variables: the difference in speed between the ghosts and Pac-Man, the length of the energised state of Pac-Man and the time of departure of the ghosts[1] for each stage cleared. In this case, we will not change previous variables for each stage cleared but we will improve the decisions the ghosts make for each action performed by Pac-Man.

- **Machine Learning applied to video games:** It is a branch of artificial intelligence (AI) in which a computer generates rules based on data that has been fed into it[3]. Video games are applications that usually need the generation of different rules depending on some environments. Even then, there are some reasons for declining to use ML in video games.

  When designing a ML, there are some complexities to take into account. Looking upon a learning model too adjusted for the difficulties of the video game, the player will not have a great challenge. But considering another one too prepared for its dangers, the player will not have any opportunity to win[4]. In addition to that, the ML algorithms must be created with caution because when the problem is quite complex a high process time might be needed.

  Nevertheless, many companies have chosen to develop ML on video games and there have been very good results such as Black & White[5] which is considered one of the revolutionaries in those video games with ML[6].

- **Technical aspects:** We need to take into account some technical aspects due to some requirements of the project.

  First of all, a game engine[7] is needed for developing the platform of the video game with all possible elements and its interactions. Furthermore, a tool to navigate on the stage is required too. For that reason, we will use different cameras offered by the game engine. Game engines will be more detailed in the section *1.2 State of the art*.

  Besides, we need one way to obtain or design 3D models from the original 2D models of Pac-Man, since the platform has to be in 3D. A modelling[8] and rendering[9] software will be required for this project accordingly. This tool offers the possibility of creating different polygonal shapes as well as making deformations and joints with other polygonal forms. We can also get some deformations by moving some of its vertexes or edges. Once the deformations are completed we will get a final form: the 3D model searched.

1.1.2 Stakeholders

This project contains several stakeholders. They will be accounted for in the following list:

- **Project developer:** This project will only have one developer, that means I am going to work on all needed tasks for the project. Because of that, I will work on the project planning, development, documentation and validation of the objectives.
- **Project director:** The project director is Javier Béjar. He is an associate professor in the Software department of *Universitat Politècnica de Catalunya.* His role is to supervise the project and guide the project developer. Providing some issues or questions about the ML algorithms appear, he will support the developer by suggesting ideas to improve while progressing.
- **Participants:** They are the members of the *validation with user*'s team. The team is composed of 10 individuals. These subjects will be asked to play the video game and fill in a questionnaire.
- **Users:** The aim of the project is to develop a Pac-Man's video game in which the level of difficulty is adapted to the user level. We will carry out experiments with a simulation so as to improve upon the difficulty in different user levels. Initially, the project is thought for any user. But perhaps not all users will realise the fact that the complexity of the game depends on their level.
- **Benefited actors on the video game sector:** They are those who will be able to profit if we end up commercialising this game. They will probably belong to some video games companies such as *Google play*[10]. Furthermore, advertisement industries are likely to be another potential benefited actor in the video game sector so long as we finally opt for contracting some *sponsor*[11].
- **Benefited actors on the researching sector:** Other possible benefited actors are the individuals whose studies are about the methodologies to increase the satisfaction of the user experience. Furthermore, there are those who only need a ML's algorithm to study other different topics. Despite the fact that both groups of benefited actors will not profit directly, they will certainly safe a lot of time.

1.2 State of the art

In this section, we are going to point out the state of the art of the different elements related to this project.

1.2.1 Machine Learning in Pac-Man

Pac-Man has been a good tool for carrying out research on ML in video games. In fact, Pac-Man's video game could help to encourage some students to learn AI in an easy way[12]. Besides, there are many studies about evolved Pac-Man's AI and the evolved ghosts' AI. Each study chooses which features are more compatible with their work. Because of that, they apply some different features to their Pac-Man's platforms from the original. The following table shows two relevant studies notably related to this project:

References	Objectives	Results
Play Ms. Pac-Man using an advanced reinforcement learning agent.[13]	Developing an AI for a Ms. Pac-Man character using an abstract but informative state space description.	They tested the experiment in three stages (one of them the classic stage), and verified that the AI was compatible (similar results) with three stages, denoting that it could be for others.
Evolving Opponents for Interesting Interactive Computer Games.[14]	Developing an AI for the opponents of Pac-Man (the ghosts) which learns from a player while playing against it. Pac-Man's intelligence can be found in three different profiles based on some criteria.	They tested the experiment in a different stage from the classical game. Hence, the test did not include the energiser feature (the ghosts cannot be eaten). The test for the three different Pac-Man's profiles were succeeded in accordance with the ghosts' adaptability.

Table 1.1: Previous studies related to the project.

1.2.2 Machine Learning algorithms

There are several areas of ML. Still, we are going to talk about some of them: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

Supervised Learning (SL). SL is made up of a model that determines a result given a set of data. The model of SL is formed by a set of input data, also known as training data. The input data required to execute the algorithm is a vector of attribute-value where each attribute can be discrete or continuous. Its usage is closed to classification and regression problems[15].

- Classification problems are those which have N clusters and some entities that need to be associated with some clusters. An example of a classification problem is a generation in real-time strategies for a video game like StarCraft II[16][17].
- Regression problems are those with a continuous value as solution. An example of regression problem is a prediction power of a combined gas and steam turbine[18].

If some counterexamples appear in the model, it will become invalid. For that reason, the models' knowledge can be changed or updated with the new invalidated information. When we generate the model, the data usage is labelled, which means that the clusters of these data are known.

Unsupervised Learning (UL). UL uses a similar composition to SL. But in this case, the data usage to generate the model is not labelled. Its learning is based on the similarity and dissimilarity among the examples. Because of that, its problems are about clustering. An example of a problem of UL is a classification of human actions using spatial-temporal words[19].

Reinforcement learning (RL). RL is made up of agents, graphs and rewards. Its functionality is simple to understand. The agents receive the current state and make a decision according to their knowledge. Finally, when the agents have made the decision they update their knowledge with the corresponding reward [20]. Both robotics and video games tend to require to make decisions according to the surroundings and the possible actions to do. For that reason, these areas are the main aim of RL algorithms. In the context of robotics we can see an example of an application to solve the problem to balancing an inverted pendulum on an unmanned aerial vehicle[21]. In the context of video games we can see other examples of an application to solve the AI for a video game with a complex strategy[22].

1.2.3 Game engines

Game engines are programs with the purpose of creating and developing video games[7]. They are composed of the following components. Some of them can be missed in some game engines.

- **Main game program:** Its objective is to manage the behaviour of the different components joined together with the algorithms programmed.
- **Render engine:** It is the handler to send the image to print in the screen. It works with low-level libraries to be able to work directly with CPU and GPU.
- **Audio engine:** It is the handler to send the sound to the sound card. It usually works with algorithms related to sound.
- **Physics engine:** It is the responsible for detecting collisions and forces applied to the bodies of the video game.

There are several game engines as well as different methods to classify them. One of them is a classification focused on *Indie Games* and *AAA Games*. In the following lines we are going to give some details about four game engines, Unity, Unreal Engine, CryEngine and GameMaker:Studio. The first two entities were mentioned in the Game Developers Conference (GDC) 2015[23] and they are getting more strength in the development of the video games' world. Cry Engine is rather popular among big video games companies and GameMaker: Studio has less reputation than the other three.

- **Unity**[24]. Unity 4 [25] had been the biggest release for a while. This version had two kinds of licenses which are the PRO and the standard. The standard one was free and PRO had more features by default like occlusion culling[26]. Recently in GDC 2015[27], the Unity Technologies team talked about the new version of Unity which contains many features from Unity 4 PRO. Moreover, it makes improvements in other features from Unity 4 like the illumination effect. Likewise, this version increases and improves the support for the different 21 platforms to build[28]. Besides, the professional edition is distributed into different ways. It can be via license (only one payment of $1,500) or via subscription (payment of $75.00 per month). The professional edition offers extra services and improvements in some features like physics games[29]. Unity is regarded as a good tool for people inexperienced in video games development and it has good resources for 2D and 3D video games development. Finally, this software is an open source, this fact allows us to make modifications and improve the game engine according to our point of view.

- **Unreal Engine**[30]. It has been developed by the company Epic Games (known by Gears of War series[31]). The current version of Unreal Engine is the fourth version. Unreal Engine is known by having a good graphical quality. It is quite similar to Unity with better graphical aspects. Because of that, its system requirements are higher than Unity. Unreal Engine had been for free until recently. In GDC 2015, Epic Games published that Unreal Engine would be for free with all its contents. But the company is required to pay 5% royalty if they earn $3,000 per quarter per game[32]. Like Unity, it is a good tool to develop for 2D and 3D video games, and its source is also open source.
- **CryEngine**[33]. It has been developed by the company Crytek (known by Crysis series[34]). Now, its current version is the third version which is an extremely powerful engine with a great realistic physics, graphical and special effects. Its system requirements are rather similar to Unreal Engine which is quite competent versus CryEngine. This engine is not available for free ($9.90 per month). Besides, they have to authorise you to have the full source code according to some requirements like the long term project[35].
- **GameMaker: Studio**[36]. This engine has been developed by the company YoYo Games. GameMaker: Studio is a good engine to make 2D video games. It is also possible to work with this engine for 3D video games development but its documentation warns about the possibility of having several problems with depths, views and other attributes[37]. There is a free trial license for this engine but if we want to commercialise a video game developed with this, we should take a Professional ($49.99) or a Master Collection ($799.99) license.

The following table summarises the previous game engine features:

Game engines	**Payment**	**3D compatibility**	**Graphical features**	**Open source**
Unity 4	Free license available	Good	Enough for the project	Yes
Unity 5	Free license available	Good	Enough for the project	Yes
Unreal Engine	Free but a royalty is required if we earn $3,000	Good	One of the best proposed	Yes
Cry engine	$9.90 per month	Good	One of the best proposed	No
GameMaker: Studio	Free but a payment license is required if we want publish it	Compatible with some issues	Good in 2D	Yes

Table 1.2: Summary of the different game engines introduced.

1.3 Use of the previous results

In this section, we are going to explain whether we will use some tools or information presented in *1.2 State of the art* for each of the different areas explored.

1.3.1 Machine Learning in Pac-Man

This project will develop a similar experiment to the previous studies introduced. We are going to explain which features will be used for distinguishing our project from the others. Comparing with the first study introduced[13], our project proposes to generate different playing levels for Pac-Man according to some previous experience given. Thus, we could simulate a beginner, medium and expert level profile. Besides, both Pac-Man and the ghosts will be learning in simulation. In this project, different models will be implemented and analysed so as to know which one fits better for our problem. The second study introduced[14] opts for removing the energisers because they increase the complexity of the development for the ghosts' learning model. Nevertheless, this project will include the energisers and cause effect to the learning of the ghosts will be kept in mind. Finally, the project will include test carried out by ten individuals to value the ghosts' adaptability for real users. In general, the studies do not cross the development of a learning model for a Pac-Man and for the ghosts. We think it can be a good practice to do it because we will be able to take more samples for our study.

1.3.2 Machine Learning algorithms

We have presented three different techniques of ML. But in this project we will only use one of them. We ruled out Supervised and Unsupervised Learning because the main target of this project is to develop a learning model which makes movements according to the user's action. To use the previous techniques and make this project available for a great number of users, a huge training data would be required to generate the model. Taking this data will be neither fast nor easy and besides it will be more restrictive than the Reinforcement Learning version. For this reason, we would rather use only Reinforcement Learning implementation with different learning models to contrast.

1.3.3 Game engines

Four game engines have been presented but to develop the platform of the video game only one is required. In the first place, we directly dismissed GameMaker: Studio due to the fact that the project platform needs to be in 3D. Although CryEngine is a powerful engine, it is too complex for the project platform's requirements. Furthermore, it does not have a free license available. So, the final choice lays between Unity and Unreal Engine. But Unreal Engine was not free when we did the hardware choice. Because of that, we chose the hardware for the development without keeping in mind the Unreal Engine's system requirement. Thus, we took an incompatible hardware with that engine. In the end, we preferred to use Unity 4 (free version) rather than Unity 5 because the project does not need the new features included in the fifth version. Besides, we think the fourth version is more stable.

1.4 Objectives

The main objective of this project is to develop a machine learning model for the ghosts of the video game Pac-Man. They will improve their movements in accordance with Pac-Man's actions. If Pac-Man takes all dots and eats the frightened ghosts rapidly, the ghosts will begin to move better. Nevertheless, if Pac-Man has greater difficulties taking all dots, the ghosts will be more kind and they will move worse. The complexity for each level will be increasing as the ghosts keep their experience for each level won.

The second objective is to develop a machine learning model for the Pac-Man character (user player) playing beginner, medium and expert. When Pac-Man is playing beginner, it starts without experience and it will not know a lot about the system. Pac-Man should learn which things are positive (dots, energisers, fruits and the frightened ghosts) and which ones are negative (the non frightened ghosts). Then, when Pac-Man is playing medium, it begins with some experience but not totally complete yet. Because of that, it will recognise some positive and negative factors but it might not recognise them all. Finally, when Pac-Man is playing expert, it begins with the maximum experience. Thus, it will know what to do in any situation from the first moment of the game.

The last objective is to develop a simulation (Pac-Man CPU versus the ghosts CPU) with a third-party interaction where the user will be able to navigate among the views of the characters and the whole scene.

1.5 Scope and obstacles

In this section, we are going to explain what we need to do in order to solve the previous exposed problem, and what can be dangerous to satisfy the objectives on time.

1.5.1 Scope and requirements

The scope and requirements of the project will be introduced in this section.

1.5.1.1 Scope

In order to solve all the objectives, developing a video game platform based on the classic Pac-Man is needed. We have to consider keeping some features from the classic game:

- Number of characters (the four ghosts and Pac-Man).
- Number of stages (one stage).
- Different items (dots, energisers and fruits).
- Restrictions in the movement of the characters (the ghosts cannot change to the opposite direction).
- Difference in speed between Pac-Man and the ghosts.
- Reduction of the ghosts' speed and increase Pac-Man's speed when Pac-Man eats an energiser.
- Delay time to the ghosts' departure.

- Different targets for each ghost.

On the other hand, the forecast features to discard are:

- Reducing speed when Pac-Man eats a dot.
- Reducing speed when the ghosts are in the tunnel.
- Shorting way when Pac-Man turns left or right.
- Original system collision between Pac-Man and the ghosts.

The platform needs to allow two modes: a game mode where Pac-Man is controlled by the user, and the ghosts by CPU, and a simulation mode where Pac-Man and the ghosts are controlled by CPU and the user controls the cameras of the scene. Game models will be modelled in 3D owing to the fact that one of the previous objectives was to navigate among the characters' views.

The algorithms to keep in mind for developing the machine learning are these based on reinforcement learning[20][38]. We are going to evaluate different reinforcement learning models, but we will choose one of them for the final application. Other algorithms implementations will be needed to evaluate some characters' attributes as well. Some characters' attributes are:

- How far a character is from other characters or elements.
- What the shortest path is between a character and another character or element.
- There is one kind of character or element near or far from a character.

The following list shows a brief summary of the scope:

- Development of a video game Pac-Man.
- A level of ghost adaptable to the level of users.
- Simulation between Pac-Man and the four ghosts.
- Navigation among the cameras in the simulation mode.

The project will not cover the following:

- All features of the video game Pac-Man.
- Multiplayer mode with two or more Pac-Men.
- Different stages.
- Simulation with different number of Pac-Man or/and ghosts.

1.5.1.2 Requirements

The requirements that we can perceive are the start and final date of the project. Before the planning of the tasks, the development will start. It will be after 24th February. The end of the project will be due June 15th at the latest, two weeks before the initial period of the oral defences accordingly.

1.5.2 Obstacles

We need to contemplate some aspects that can make it difficult to accomplish the dated milestones:

- Error code: The development of this project is formed by one developer. Thus, it can be difficult to notice when a bug appears. Furthermore, it is hard to realise what the cause is and how it can be fixed. There is no way to safe time with this problem. Consequently, we have to be cautious with this problem and try to make good analysis when we come across an error.

- Resources consume: The machine learning methodologies can consume a lot of resources if their configuration is too accurate. Determining the barrier between accuracy and consumption can be considered hard work. We should estimate which of the options would be better depending on how near the deadline is.

- Bad implementation: It is essential to meet up with the director of the project as often as possible so as to avoid carrying out wrong implementations. This way the current implementation can be supervised. It is always a good idea to ask the director whatever questions may arise.

- Validation of the project: We can have problems with the validation due to the fact that this project does not have a positive or negative result. Its objective is about adapting the complexity for the user, but the difficulty for a user is ambiguous and we might not cover all kind of users.

1.6 Methodology and rigour

We will explain the methodology that we are going to apply to formulate and develop the project. We will define the tools that we are going to use and what function they will do. Finally, the project has to be validated to check whether the work is well done. Thus, we will expose how we are going to validate them.

1.6.1 Work method

Before planning anything about the project, we have checked the feasibility of this. We searched more information about reinforcement learning and their applications. We noticed that the project was viable when we looked at some implementations for table board games and some explanations about their given usages.

The project will be done with a work methodology based on SCRUM, because it allows to plan the objectives of the project with some flexibility. In addition to that, several meetings are required for this methodology. In particular, the members of the team have to do sprints (meetings) every 7-30 days during the planning, sprints every day during the development and two final meetings for the review and retrospective at the end of the development[39].

Unlike SCRUM, we are going to speak with the project director each week to determine the scope and the tasks of the project. These tasks will be included between the development and validation phases. Once we have finished defining the features of the project, we will meet to fix some dates for each task. Unlike SCRUM again, the director and developer of the project will meet weekly to check the correctness of the ended tasks and readjust the next tasks date until the last week.

1.6.2 Development tools

The whole project will be developed via Unity[24] and Mono Develop[40] using C# scripts. Unity will offer all camera and model handlers, and Mono Develop will allow to apply debug on sources.

We will use Autodesk 3DS MAX[41] to modelling 3D models because it is a powerful tool for people without a lot of experience in modelling. Git[42] service will allow to have a safe version control. Because of that, we will use the Git server Bitbucket[43]. Using Bitbucket will help us notice which tasks are done and when we did those tasks.

1.6.3 Validation methods

We have to keep in mind that it is a subjective task to validate how good the AI of characters are. Because of that, it is difficult to globally determine whether they are right or wrong. Thus, the validation method planned is divided into two phases. For the first phase, we will study the results obtained from different configurations on the parameters (algorithm for learning and different implementations of them). We will focus on the following data:

- Number of deaths for each ghost.
- Number of deaths of Pac-Man.
- Number of dots taken.
- Duration of the game.
- Final score.

The second phase will consist in doing a test with ten real users, owing to the fact that the application requires to be orientated to offer a game difficulty fitted to the users' level. The test is made up of one match in Puck-Man (original game), another match in our platform and a questionnaire exposed in *B SMART PACMAN Test Document.*

CHAPTER 2 PROJECT MANAGEMENT

The different phases and tasks of the project will be presented in this chapter. Besides, we will describe these tasks and expose the temporal planning for each of them. Finally we will explain how the planning will be executed.

2.1 Description tasks

In this section, all tasks planned will be explained. The tasks of the project can be gathered in the following phases: project planning, main development and final tasks.

2.1.1 Project planning

This phase consists in analysing the following subjects:

- Scope of the project.
- Temporal planning.
- Economic management and sustainability.
- State of the art.

Each subject will be analysed in the same order as in the previous list. This order is well done due to their requirements. Before planning the tasks of the project, this one has to be analysed. To find out the cost of the project we have to know how much time we are going to spend.

2.1.2 Main development

Once the temporal planning is ended, some tasks can be initialised. Before processing the main development, we have to prepare all work systems. Thus, we have to install all software needed and generate a repository for the version system. After all the required software installations, we have to carry out the subsequent tasks:

- **Developing Pac-Man video game's platform:** We have to develop a classic Pac-Man. Consequently, we should analyse some details of the classic game before developing the platform. Then, we should develop the following attributes:
  - The ghosts' movement and Pac-Man's movement.
  - Load level with dots, energisers, bonus points (like fruit), the ghosts and Pac-Man.
  - Handler score.
  - Pause, win and lose the game.
  - Consume dots, bonus points and energisers with their respective effect.

  Furthermore, we have to link some input keys with the movement of Pac-Man. Finally, we have to develop a game flow which allows to access both modes (simulation and game) as well as exiting the video game. To finish up with development platform we have to test the whole video game to detect some possible bugs and fix them.

- **Developing reinforcement learning (RL) methods:** First of all, we will need to search for some information about the different learning models of RL and how these are applied. Then, we will develop the classes needed for the application of some of them. After that, we will be ready to apply the learning model for Pac-Man and the ghosts.

- **Applying RL to Pac-Man:** Pac-Man has to be able to play different levels (beginner, medium and expert). Pac-Man's objective will be that of getting the maximum score. After applying the Pac-Man's model, we have to analyse some results given some executions of the simulation.

- **Applying RL to the ghosts:** The ghosts need to decide the next movement according to the level of Pac-Man. Different objectives are assigned to each single ghost as it is seen in the classical game. Nevertheless, the goal of the frightened ghosts will be the same (running away from Pac-Man) for each of them. After applying that, we have to check if the behaviour of the ghost is appropriate.

- **Developing navigation on simulation:** We have to consider assigning each character a camera. Moreover, we have to link some input keys to each camera. Finally, when we are in the main camera, we have to connect some input keys with the following actions:
  - Move (front, behind, up, down, left and right).
  - Rotate (up, down, left and right).
  - Reset position and rotation.

  Finally, we should test the correct function of the navigation.

- **Modelling 3D models and applying textures:** We have to obtain all elements found in the surroundings of Pac-Man's video game, such as walls, dots, energisers, fruits, the ghosts and the own Pac-Man. We can use some simple shapes for some elements, e.g. use a cube as a dot, to avoid spending time on shaping some models. Once models have been defined, they have to be applied to the game with a texture.

We will save an average of 2 hours for all previous tasks to document the details about the process of the development task. Developing the platform, developing the AI, developing the navigation on simulation and modelling 3D models are independent tasks. Because of that, if there were more developers in the project, these tasks could be developed at the same time.

2.1.3 Final tasks

To conclude with the project's development, we have to validate the results of the project for the exposed problem. The project validation is divided into two dependent tasks:

- **Validating with simulation:** We will recollect the different data according to the previous chapter in the section *1.6.3 Validation methods* and we will analyse them. We will collect data for different executions with the different learning models, and depending on the results, we will determine which model will be taken for the final application.
- **Validating with users:** After choosing the most accurate method, we will arrange a meeting with some users. These players will play two matches one on the original Pac-Man and the other one on the project and they will answer the test explained in the previous chapter in the section *1.6.3 Validation methods.* Finally, we will analyse the answers to validate the main objective of the project.

Once the project has been validated, we can focus on ending the documentation. We will revise all content written in the project development. Finally, when we have submitted all the documentation we will be able to get ready for the oral defence.

2.2 Resources

In this section, all resources used to solve all the previous tasks will be explained. The resources are divided into software, hardware and a set of users.

2.2.1 Hardware

- *Personal computer (computer with the following specifications: AMD Athlon(tm) II X2 250 Processor Dual Core 3GHz, 4GB of RAM, nVidia GeForce 8400 GS):* used for modelling the 3D models and applying textures task.
- *MacBook (laptop with the following specifications: Inter Core 2 Duo 2.4GHz, 3GB of RAM, Intel GMA X3100):* used for developing the rest of the tasks.

2.2.2 Software

- *Windows XP:* used for modelling the 3D models and applying textures task.
- *Mac OS:* used for developing all tasks except for modelling the 3D models and applying textures task.
- *Unity*[24] *and MonoDevelop*[40]*:* used for all main development tasks except for modelling 3D models and applying textures task.
- *Autodesk 3DS MAX*[41]*:* used for modelling the 3D models and applying textures task.

- *Gimp*[44]*:* used for developing task modelling the 3D models and applying textures.
- *Google Sheets:* used for analysing all extracted data.
- LaTeX[45]: used for documenting all tasks.

2.2.3 Users

The project will require the participation of at least 10 individuals to perform the validation with users. They will play two matches one in the original Pac-Man and other one in the game mode of the project and they will answer a test with their opinions.

2.3 Temporal Planning

The initial date of the project is February 16th 2015. After this date we have planned provisional dates to finish the tasks. But this temporal planning can be changed owing to the feedback given by the director to the project's developer.

2.3.1 Timetable

The following table represents the expected duration in hours and the dependencies for each previous task.

ID	**Task name**	**Hours**	**Dependencies**
t1	*Project planning*	*115*	
t1.1	Scope of the project	20	
t1.2	Temporal planning	20	t1.1
t1.3	Economic management and sustainability	25	t1.2
t1.4	First presentation	20	t1.3
t1.5	State of the art	30	
t2	*Main tasks*	*305*	t1.2
t2.1	Initial preparation	5	
t2.2	Analysing Pac-Man video game's platform	30	t2.1
t2.3	Developing Pac-Man video game's platform	30	t2.2
t2.4	Testing Pac-Man video game's platform	20	t2.3
t2.5	Developing RL methods	60	t2.1
t2.6	Applying RL to the ghosts	50	t2.5
t2.7	Testing RL to the ghosts	10	t2.7
t2.8	Applying RL to Pac-Man	50	t2.5
t2.9	Analysing RL to Pac-Man	10	t2.9
t2.10	Developing navigation on simulation	15	t2.1
t2.11	Testing navigation on simulation	5	t2.10
t2.12	Modelling 3D models and applying textures	18	t2.1
t2.13	Applying models to video game	2	t2.12
t3	*Final tasks*	*75*	t2
t3.1	Validating with simulation	15	
t3.2	Testing with users	5	t3.1
t3.3	Validating with users	20	t3.2
t3.4	Finishing the documentation	35	t3.3

Total		**495**	

Table 2.1: Estimation of the development planning time.

2.3.2 Gantt chart

The following picture shows the expected temporary planning represented with a Gantt chart performed with GanttProject[46]:

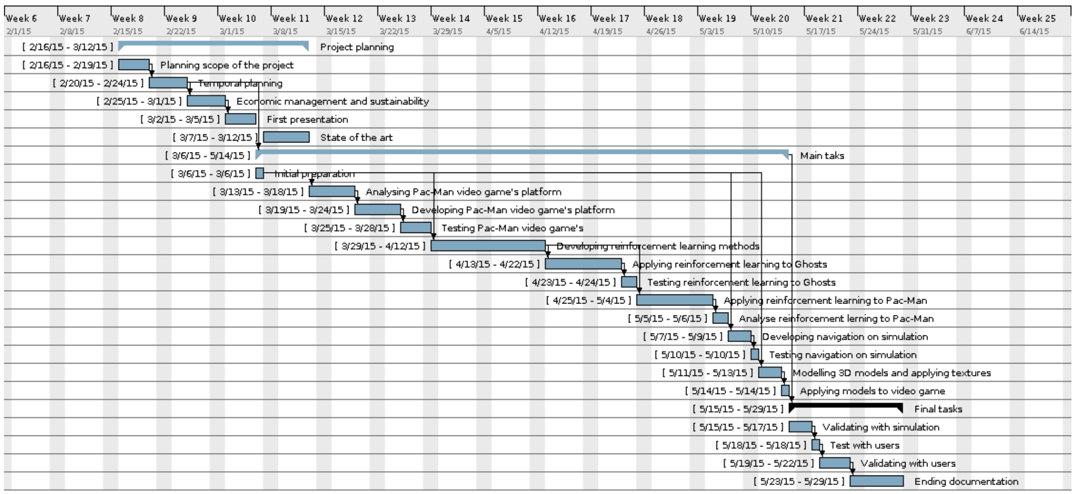

Figure 2.1: Gantt chart.

We can observe that the highest priority at development phase is to develop a video game platform with a RL method. 3D models and navigation on simulation can wait until the end of the development tasks.

2.3.3 Action plan

In this section, we are going to explain how we are going to execute the tasks according to the initial planning.

First of all, we will try to accomplish the dates determined in the Gantt chart. But we have to take into account all possible issues of the project to have a realistic view of the planning. Because of that, as we said previously, this is a provisional temporal planning. Accordingly, this project will have a flexible closing date from the beginning. Nevertheless, we will have a maximum end date. As the initial period of the oral defences is assigned on the 30^{th} of June, the latest deadline will be June 15^{th} (two weeks before). Finally, we are going to have 3 weeks leeway after the planned deadline date (29^{th} May) for any perceived hindrance. In principle, we do not need to sacrifice some time for any task. Even if it is necessary, we have three deviations to satisfy the latest deadline:

- **Avoiding modelling 3D models:** We will use the simple shapes with different colours to know which is one, and this deviation will let us save 20 hours in the development. But It will be applied if we need less than a week's time.

- **Omitting the free navigation on simulation:** It is the least important objective, and this task will give us more than half a week. However, this and the previous deviation will be applied together if we need more than a week.
- **Reducing the complexity of the learning model of Pac-Man:** It will remove a lot of work time but we will no longer have a good validation with simulation. Nevertheless, we will keep the validation with users (which is the most important validation) intact, due to the main objective. This and both previous deviations will be applied if we need more than two weeks.

The total time expected for the project is 495 hours for 15 planned weeks, this is an average of 5 hours per day (weekends included). But we organised to have a maximum availability of 40 hours per week. Thus, we can see that our project is likely to be finished in the given planning period.

2.3.4 Plan Monitoring

In this section, we are going to explain what changes from the initial planning have appeared.

After concluding with the GEP's documentation, the development of the Pac-Man video game's platform was planned to be done. Before this development, a huge analysis was required. This analysis was done with some documentation checked to elaborate the section *1.2 State of the art*. Some Pac-Man's features have not been added to our platform. One example of a discarded feature is the reduction of the speed when Pac-Man eats a certain element. During the development, an issue regarding the collisions between two objects appeared. Diminishing this issue, the development of the platform ended with successful results. During the development, the features were tested once finished. The last test was executed at the end of the development, checking the correct flow of the game and the simulation. Analysis, development and test platform were finished on time, that is, on the 29th of March.

Once the platform was ready to be used, a system of classes needed to use the Reinforcement Learning methods were expected to be prepared. Due to this fact, we have designed and developed those classes. The development of these classes has been done rather quickly. However, its definition has changed after trying to apply it for the ghosts' role. The first definition worked with a graph defined by a set of states. There was defined an action for each state, and a predefined state fate for each state-action relation. But an action executed in a particular state does not have a predefined destination state, because this operation is *nondeterministic* for our stage. For that reason, the graph will only have the relation between action and state defined. In addition to that, we have been studying some different methods to learn[47]. Thus, we determined the destination state in execution time. We have selected some of them to be candidates to study, but only four of them have been developed. At the beginning, we wanted to make sure that there is enough time to have at least a huge test for two versions. But if we ended the development on time we would consider to include other candidates.

The application of the learning model for the Pac-Man's role has been done after ending the learning model for the ghosts. In particular, the application of both learning models has consisted in defining the states and the actions, and linking each state to each possible action with a reward. Moreover, In the case of Pac-Man, a definition of the training methods has been required due to the different playing levels needed. Finally, the adjusting of learning model parameters for Pac-Man's role has been moved forward before the ghosts' role because we have to define the first one with a static role to be able to value which level is better than the other.

Finally, the simulation and the 3D models development have been developed faster than the defined planning. Because of that, we have had four days to develop two learning models. The final documentation has had to be postponed one week because we have been indisposed during all week.

2.3.5 Plan Modifications

In this section, we are going to show the final Gantt chart of the whole project (documentation and development).

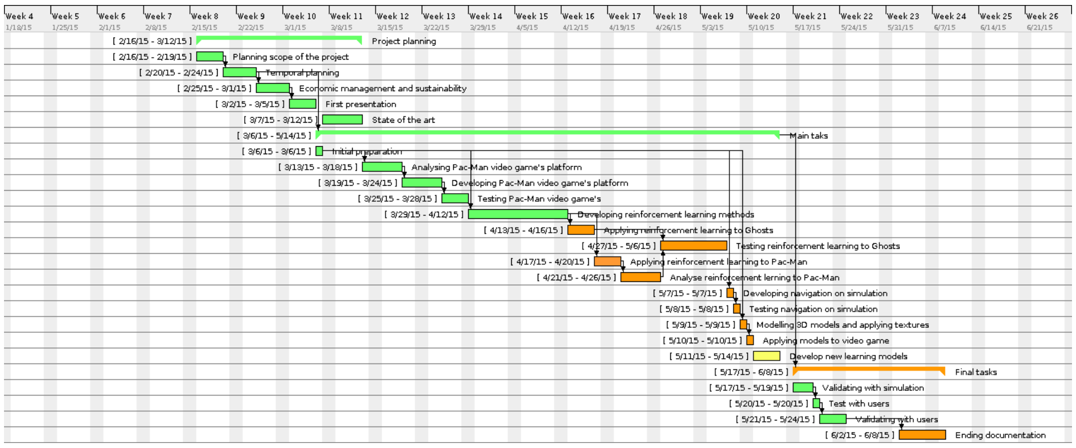

Figure 2.2: Gantt chart modified.

The colour legend is represented in the following list:

- Green: The task has been kept as the initial planning.
- Orange: The task has been changed (the duration or/and the initial date).
- Yellow: It is a new task.

The duration of the project has been prolonged one week, but the working time has been kept.

2.4 Budget

In this chapter the budget and the sustainability of the project will be estimated. Because of that, this document contains detailed information about the costs of the different elements and resources needed by the project, and a detailed evaluation about the impact on economic, social and environmental areas. In addition to that, we will expose some control mechanisms in order to counteract the possible deviations in the project.

2.4.1 Identification of costs

In this section, the resources of the project will be identified and analysed to establish an estimated price. The identification of costs can be classified into two categories: direct and indirect costs.

2.4.1.1 Direct costs

Direct costs mean the cost of the resources directly applied to fulfil the project's tasks. These costs are usually classified into three categories: hardware, software and human resources. But owing to the fact that the length of the project is less than 3 years, hardware and software will not be used for all their lifespan. For that reason, human resource will be the only direct cost.

2.4.1.1.1 Human resources

In this section, we will account for the cost of the different roles assigned to the specific tasks of the project. The following table shows the cost per hour[48] and the whole project for each role:

Role	Price per hour	Hours	Total
Project manager (PM)	70.00 €	150	10,500.00 €
Game designer (GD)	40.00 €	30	1,200.00 €
Designer (D)	30.00 €	18	540.00 €
Data analyst (DA)	35.00 €	45	1,575.00 €
Game programmer (GP)	25.00 €	207	5,175.00 €
Technical assistant (TA)	25.00 €	5	125.00 €
Game tester (GT)	25.00 €	40	1000.00 €
Participants (P)	0.00 €	5	0.00 €
Total		500	**20,115.00 €**

Table 2.2: Estimated cost for each role.

The roles chosen are the appropriate ones to develop a video game with a complex AI. The Project manager is not directly connected with the topic of the project but he or she has one of the most important roles because if he does a bad job, the project will probably fail. This role will require a great amount of hours to plan and verify the project. The Game designer is needed to analyse the classic Pac-Man game as well as to introduce the idea for the Game programmer. Furthermore, the analysis of the different learning models to apply for the project will be in the Game designer's charge. The Designer is not as essential as the Game designer but they will definitely make the game much more attractive for the users. The Data analyst role is important for this project because we need a good verification for the main objective of the project. Their cost is not really expensive due to our requirements. The Game programmer will devote more hours than anyone else. Although it is not very expensive, it will be a large portion of our investment due to the number of hours. The Technical assistant will be required so as to set up our machines for the project. Because of that, they will not spend a lot of hours on this project. The Game tester is usually a cheap role. Furthermore, they are required to verify the playability of the video game developed. Finally, the participants are included in this table, even though they are not members of the team and they do not get any economic contribution for their collaboration. The following table relates the different tasks of the project to the different roles and costs:

Task	**Role**	**Hours**	**Price**
Scope of the project	PM	20	1,400 €
Temporal planning	PM	20	1,400 €
Economic management and sustainability	PM	25	1,750 €
First presentation	PM	20	1,400 €
State of the art	PM	30	2,100 €
Initial preparation	TA	5	125 €
Analysing Pac-Man video game's	GD	30	1,200 €
Developing Pac-Man video game's	GP	30	750 €
Testing Pac-Man video game's	GT	20	500 €
Developing reinforcement learning methods	GP	60	1,500 €
Applying reinforcement learning to the ghosts	GP	50	1,250 €
Testing reinforcement learning applied into the ghosts	GT	10	250 €
Applying reinforcement learning to Pac-Man	GP	50	1,250 €
Analysing reinforcement learning applied into Pac-Man	DA	10	350 €
Developing navigation on simulation	GP	15	375 €
Testing navigation on simulation	GT	5	125 €
Modelling 3D models and applying textures	D	18	540 €
Applying models to video game	GP	2	50 €
Validating with simulation	DA	15	525 €
Testing with users	GT, P	10[1]	125 €
Validating with users	DA	20	700 €
Finishing the documentation	PM	35	2,450 €
Total		500	**20,115.00 €**

Table 2.3: Tasks estimation cost for each role.

2.4.1.2 Indirect costs

In this section, there will be introduced the different resources that will not be used for all their lifespan, and also those which cannot be assigned a particular task. The first ones are hardware and software resources and the last ones are the consumption resources.

2.4.1.2.1 Hardware resources

We will expose the cost of the different hardware resources needed to accomplish the tasks of the project. Those costs will be along with their depreciation which will be calculated considering the resource usage time. In particular, the depreciation will follow the next formula ($\frac{price}{lifespan} * usage$).

[1] 5 hours for GT and 5 hours for P. We have only had in mind the developer tasks on the estimation of the temporal planning

Product	**Price**	**Lifespan**	**Usage**	**Depreciation**
PC (with all the needed devices)	750.00 €	4 years	3 days	1.55 €
MacBook	900.00 €	4 years	99 days	61.03 €
Total	**1650.00 €**			**62.58 €**

Table 2.4: Hardware estimation cost.

We recall PC will be used to modelling the 3D models while MacBook will be applied to carry out the rest of the tasks of the project.

2.4.1.2.2 Software resources

The costs of the software resources needed to develop the tasks of the project will be announced in this section. Like hardware resources, these costs will be explained together with their depreciation following the same formula.

Product	**Price**	**Lifespan**	**Usage**	**Depreciation**
Windows XP	50.00 €	3 years	3 days	0.14 €
Mac OS Snow Leopard	20.00 €	3 years	99 days	3.36 €
Unity	0.00 €	3 years	60 days	0.00 €
Autodesk 3DS MAX (student edition)	0.00 €	3 years	3 days	0.00 €
Gimp	0.00 €	3 years	1 day	0.00 €
Git	0.00 €	3 years	64 days	0.00 €
ShareLatex	0.00 €	3 years	94 days	0.00 €
Total	**70.00 €**			**3.50 €**

Table 2.5: Software estimation cost.

The operative system (OS) Windows XP will be installed in PC, and Mac OS Snow Leopard in MacBook. Because of that, the distribution of the hours for these operative systems are equivalent to the related machines. We have tried to use the rest development tools with free license, since the aim of this project is not economic. Because this project is carried out by a college institution, we are free to use an Autodesk 3DS MAX student edition, otherwise its cost would be 7,260.00 €.

2.4.1.2.3 Consumption resources

In these sections we will explain the different costs related to the sources needed to allow the project's resources work. These sources are:

- Electricity: The average electricity used for a conventional PC is 115 Whr. It costs annually near 60.00 € when PC works 5 hours. Because of that, the estimation cost for the project will be:

$$\frac{price\ of\ electricity\ per\ year}{days\ for\ a\ year} * worked\ days = \frac{60}{365} * 102 = 16.77€$$

- Internet: The Internet receipt is about 60.00 € each month. As a consequence, the estimation cost for the project will be:

$$\frac{price\ per\ month * number\ of\ months}{days\ of\ the\ price} * worked\ days = \frac{60 * 12}{365} * 102 = 201.21€$$

2.4.1.3 Contingency

The contingency is a prefixed percentage applied to the direct and indirect costs. The percentage of the contingency for this project will be 10%. The project will have a cost increase of:

- Direct cost: $20,115.00€ * 0.10 = 2,011.50€$
- Indirect cost: $284.06€ * 0.10 = 28.41€$

2.4.2 Control management

It is important to bear in mind that some modifications in the budget are likely to be made since every task can be difficult itself.

The PC or MacBook can break down like any machine. If the PC broke down, we would omit the modelling 3D and we would use simple shapes as we planned before. For that reason, this event would not affect the budget. Otherwise, if the MacBook fell apart we could try using PC as substitute. In the event that both machines broke down, we should buy another machine to complete the tasks. Thus, we will assume a 20% of probability that PC or MacBook cease to function. Because of that, the probability that both computers go wrong is 4%. Another computer can cost 900 € with a depreciation of 62.88 € (counting all hours of the project). Finally, the cost for these unforeseen events will be 2.52 €.

As planned in the temporal planning chapter in the section *2.3.3 Validation methods*, we can omit some Designer, Analyse and Programming tasks, but we should increase other tasks with the aim of accomplishing the deadline with deviations. However, if we need to apply this mechanism, we will do it with some improvisation depending on the current status of the project. For that reason, we will not pre-calculate its cost.

2.4.3 Budget Modifications

In this section, we are going to expose the different changes of the budget generated by the modifications of the plan.

The following table shows the changes in the costs of the tasks in the budget:

Task	Role	Hours	Old price	Current price
Applying reinforcement learning to the ghosts	GP	16.5	1,250.0 €	412.5 €
Testing reinforcement learning applied into the ghosts	GT	43.5	250.0 €	1,087.5 €
Applying reinforcement learning to Pac-Man	GP	16.5	1,250.0 €	412.5 €

Analysing reinforcement learning applied into Pac-Man	DA	43.5	350.0 €	1,522.5 €
Developing navigation on simulation	DA	5	375.0 €	125.0 €
Modelling 3D models and applying textures	D	8	540.0 €	240.0 €
Develop new learning models	GP	20	0.0 €	400.0 €
Total			**4015.0 €**	**4200.0 €**

Table 2.6: Cost of tasks modified.

The difference between the new and the old budget is **185.0 €** ($4,200.0 - 4,015.0 = 185.0$). We will need to increase the price of the final budget adding this modification.

2.4.4 Summary

The estimated budget for this project is indicated in the following table:

Resource	**Price**
Direct Costs	20,115.00 €
Human resources	*20,115.00 €*
Indirect Costs	284.06 €
Hardware resources	*62.58 €*
Software resources	*3.50 €*
Consumption resources	*217.98 €*
Contingency	2,039.91 €
Unforeseen	2.52 €
Budget Modifications	185.00 €
Total	**22,626.49 €**

Table 2.7: Estimation of the project's cost.

2.5 Sustainability

The analysis of the project's sustainability will be divided into three sections: economic, social and environmental. Each section will explain how the project affects both negatively and positively. Finally, we will present a sustainability matrix which will summarise and evaluate each category.

2.5.1 Economic sustainability

In this section there is a study about the cost of different elements including human resources. Nevertheless, there is no planning about the tasks' cost before the development. Providing the project was commercialised, we do not think it would be feasible. In fact, the costs of the roles are a big investment and we do not focus the aim of the project on a marketing vision to sell a video game. We could decide to try to make a collaboration with the university to use the video game platform for some subjects to practice applying AI, or formalising a library with ML algorithms developed for some researching entities interested in the methodologies to increase the satisfaction of the user experience. Finally, it is easy to see that each task is equivalently assigned according to their importance, tasks related to AI have more time assigned than those linked with 3D models. This project has a planning to reuse some simple forms as 3D models but it does not have any other component such as AI libraries. At the end of the project, we have only had an increase of 8.2% in our budget. Thus, we consider good economical results. There are not any unexpected costs during the project besides the variances in human resources which we consider to monitor during the development according to the necessities of each task.

2.5.2 Social sustainability

The country where the project is going to be developed has a stabled social and political situation. Currently, the sector of video games is at a high social position. Smartphones have allowed many small video games companies to take advantage of their projects and earn a lot of money. This project will not change the world of video games but we have not found out any entity which could be harmed by this project either. In fact, some people will play with it and will enjoy a Pac-Man with a difficulty fitted by their level.

2.5.3 Environmental sustainability

A computer is required by each task planned. Because of that, the resource consumption is high. The tasks are divided into two computers:

- The personal computer is used for 3 days during 5 hours per day and it consumes 130 W.
- The MacBook is used for 99 days during 5 hours per day and it consumes 100 W.

The whole estimated energy needed is 51.45 kWh, which is equivalent to 19.8kg of CO_2. In general the project does not affect drastically to the world's environment, owing to the fact that we do not need to generate any physical material nor take any primary material except for some electric powers.

2.5.4 Sustainability matrix

In the following table we summarise the explanation of the previous section as values between a range.

Sustainable?	**Economic**	**Social**	**Environmental**
Planning	Economic viability	Improve quality life	Resources analysis
Value (0:10)	4	4	1
Results	Final cost versus prevision	Impact into social environment	Resources consumption
Value (-10:10)	8	10	10
Risks	Adaptation for changes of situation	Social damage	Environment damage
Value (-20:0)	0	0	0
Total value (-90:60)	**37**		

Table 2.8: Sustainability matrix of the planning of the project.

CHAPTER 3 PLATFORM DEVELOPMENT

In this chapter we are going to explain the platform, its structure, its features and the different modes available in the application.

3.1 Platform

The platform has been baptised as **SMART PACMAN** and it is based on the original video game Pac-Man, its models are in 3D. However, the camera used to project the stage is orthogonal, due to the fact that the perspective camera causes a weird effect. The platform begins with the following screen:

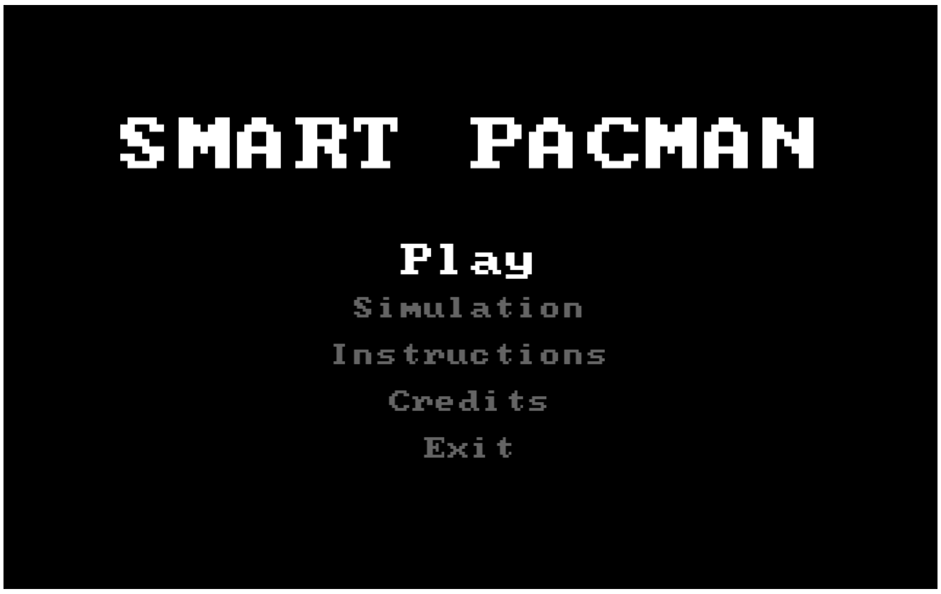

Figure 3.1: The start menu.

The following list describes the different options of the menu:

- **Play.** It starts the game mode.
- **Simulation.** It shows another menu to select the user level followed by starting the simulation mode.
- **Instructions.** It shows the relation between the keys and the actions in game and simulation mode.
- **Credits.** It shows the people related to the development of the platform.
- **Exit.** It ends the application.

You can make the following actions to interact with the menu:

- **Moving the option selected to up:** ↑ or W.
- **Moving the option selected to down:** ↓ or S.
- **Going into selection:** Enter.

The design and the stage of the project's platform are relatively equal to the original Pac-Man.

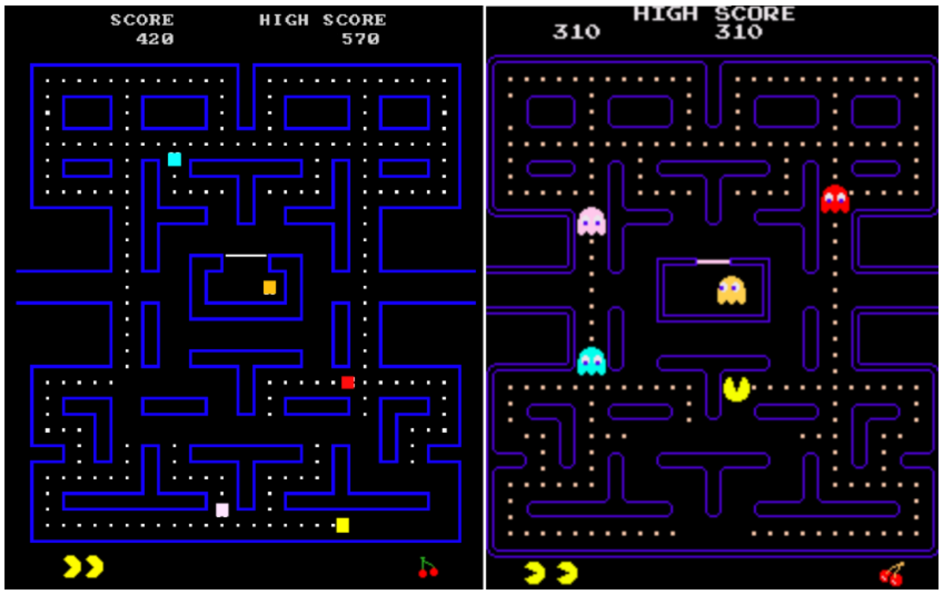

Figure 3.2: Left Pac-Man platform of the project and right original Pac-Man.

3.2 Structure

The structure of this platform consists of a set of classes divided into some categories:

- **AI.** It contains the set of classes referred to the artificial intelligence for Pac-Man and the ghosts. In these categories there are not the Reinforcement Learning classes but the ones that will query the movements to them. This category has the following classes:

  - **AI.** It contains intelligent algorithms such as A* which determines the distance between a character and other character or element. Also, It contains a method to get which direction will be the next movement.
  - **AIGhost.** It is a child of **AI**, and it is adapted to the ghosts' necessities. Although there are two roles defined, Random and Agent, the last one is the role used to query the agent.
  - **AIPacman.** As **AIGhost**, it is is a child of **AI**, and it is adapted to the necessities of Pac-Man.
  - **AutoGhost.** It determines the movements when the ghosts do not need to take decisions, such as when a ghost dies that its only target is to arrive to the ghost's house taking the shortest path.

- **Animation.** It contains the set of classes that make the animation. Their aim is to embellish the game.
  - **AnimationEnergizer.** In the original game the energisers are animated to emphasise that these elements are important to do something. Thus, this class contains the methods to animate energisers.
  - **AnimationPacman.** It makes the animation of Pac-Man recreating eating.
- **Auxiliaries.** It contains the set of classes to allow to execute some intelligent algorithms such as A*. The classes (**Heuristic**, **PriorityQueue**, **TrackPosition**) that belong to this category do not require an explanation.
- **Handlers.** It contains the handlers needed to execute the platform. The following classes belong to this category:
  - **FooterHandler.** It handles the footer indicator where the number of lives and the list of fruits of the stages passed are shown.
  - **FruitHandler.** It handles the appearance of the fruit when they need to appear.
  - **InputHandler.** It handles the input keys of the platform.
  - **LevelHandler.** It handles the load of the stage. This class is ready to load other stages owing to the fact that it loads the stage by means of a file. But in the project we only use the stage of the official Pac-Man.
  - **LifeHandler.** It handles the extra and subtraction lives.
  - **LogicHandler.** It handles the collisions and the order of the factors such the movement of the ghosts and Pac-Man. Besides, it calls the rest of handlers for the execution of the platform.
  - **ScoreHandler.** It handles the increase of the score and keeps the score when it is the highscore.
  - **SoundHandler.** It handles the different sounds and music of the platform.
- **Positioning.** It contains the set of classes needed to move and locate the different characters and elements of the platform. The classes belonging to this category are:
  - **CameraMovement.** It contains all methods to move a camera.
  - **CharactersMovement.** It contains all methods to move a character. Similarly, it has the transformation from real to stage position.

  - **PositionElement.** It contains the methods to transform real to stage position. Basically, it is used to get the stage position of a particular element.
  - **Teleport.** It contains the methods to use the teleports of the stage.

- **Score.** It contains the set of auxiliaries classes to score:
  - **FruitScore.** It contains the methods to determine a particular score for a fruit.
  - **TextScore.** It contains the methods to print the score.

- **Status.** It contains the set of classes needed to handle the status of the platform and its characters:
  - **EatItem.** It contains the methods to make a particular element eaten by Pac-Man disappear.
  - **Status.** It contains the methods to determine the status for general characters.
  - **StatusDying.** It contains the methods to determine if the player is dying.
  - **StatusGhost.** It is a child of **Status**, and it is adapted to the role of the ghosts. Also, it loads the different textures for each status.
  - **TextureGhost.** It contains the texture for the different status.

- **UI.** It contains the set of classes which allows the users to interact with the platform's user interface.

3.3 Features

The platform will not include exactly the same features from the original Pac-Man. But in the first place, we can see that the stage is relatively equivalent to the original Pac-Man *Figure 3.2*. The following list will enumerate the differences between the SMART PACMAN and Puck-Man (original Pac-Man):

- **The ghosts' states.** In Puck-Man the ghosts can be in three different states (chase, scatter and frightened). SMART PACMAN has only two different states (frightened and non frightened). When the ghosts are not frightened they can choose to attack or flee depending on their learning.

- **Difference speed reduction.** In Puck-Man the difference in speed between Pac-Man and the ghosts is reduced for each stage passed. SMART PACMAN does not change this variable, it is always the same unless the ghosts are frightened.

- **Speed in tunnel.** In Puck-Man the ghosts' speed reduces in tunnel. SMART PACMAN keeps the same ghosts' speed outside and inside the tunnels.

- **Speed when eating dots.** In Puck-Man Pac-Man reduces speed while it is eating a dot. SMART PACMAN keeps the same Pac-Man's speed except when the ghosts are frightened.

- **Turning.** Puck-Man allows Pac-Man to turn some pixels before the ghosts. SMART PACMAN lets Pac-Man and the ghosts turn only when they are in the middle (as the ghosts in Puck-Man).

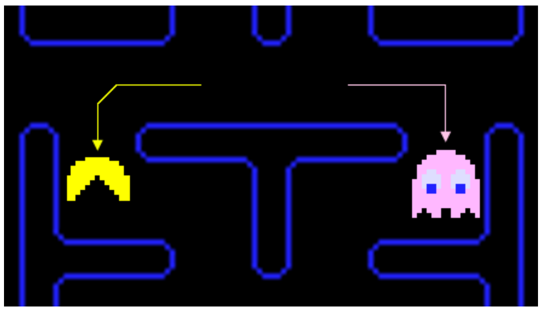
Figure 3.3: Puck-Man turning system.

- **Special zones:** Puck-Man does not allow the ghosts to turn up in the red zones of the *Figure 3.4* when they are in chase or scatter state. Because of that, they can only crossing the zone from right-to-left or left-to-right. SMART PACMAN does not use those zones in different ways.

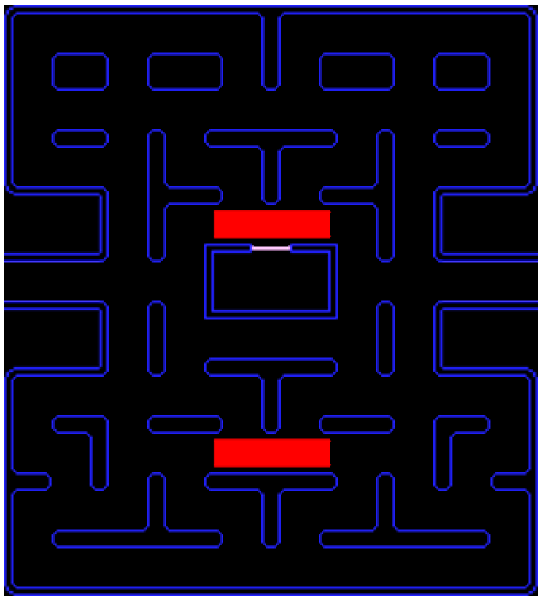
Figure 3.4: Red zones of Puck-Man.

The rest of features are practically equal according to the documentation[1]:

3.4 Game Mode

This mode allows to play the game of the application where we control Pac-Man. In this mode, we start with three lives which one of them is lost when some non frightened ghosts catch us. However, we can reach a life when we earn 10.000 points. Thus, we have to eat the frightened ghosts, energisers and dots to earn points. The aim of the game mode is to earn as many points as we can to try to beat the highscore.

The controls are simple, Pac-Man can go up, down, left and right when the stage allows it. We can choose to play with WASD keys or arrow keys. The following list enumerates the relation between action and keys:

- **Up:** ↑ or W.
- **Down:** ↓ or S.
- **Left:** ← or A.
- **Right:** → or D.
- **Pausing:** P.
- **Exiting:** Esc.
- **Moving the option selected to left in exit:** ← or A.
- **Moving the option selected to right in exit:** → or D.
- **Selecting selection in exit:** Enter.

3.5 Simulation Mode

This mode allows to analyse a match between the ghosts and Pac-Man controlled by CPU. The user can interact with the different cameras in the scene. There are six cameras, one for each character (Pac-Man and the four ghosts) and the last one for the scene. Even though the first five can only use to look through them, the camera of the scene can be translated and rotated. The following list enumerates the different actions with the related key:

- **Changing to Pac-Man camera:** 1.
- **Changing to Blinky (red ghost) camera:** 2.
- **Changing to Pinky (pink ghost) camera:** 3.
- **Changing to Inky (blue ghost) camera:** 4.
- **Changing to Clyde (orange ghost) camera:** 5.
- **Changing to scene camera:** 0.
- **Moving scene camera to front:** ↑ or W.
- **Moving scene camera to behind:** ↓ or S.
- **Moving scene camera to left:** ← or A.
- **Moving scene camera to right:** → or D.
- **Moving scene camera to up:** Ctrl + (↑ or W).
- **Moving scene camera to down:** Ctrl + (↓ or S).
- **Rotating scene camera to up:** Shift + (↑ or W).
- **Rotating scene camera to down:** Shift + (↓ or S).
- **Rotating scene camera to left:** Shift + (← or A).
- **Rotating scene camera to right:** Shift + (→ or D).

- **Putting scene camera to initial position:** R.
- **Pausing:** P.
- **Exiting:** Esc.
- **Moving the option selected to left in exit:** ← or A.
- **Moving the option selected to right in exit:** → or D.
- **Selecting selection in exit:** Enter.

All cameras are perspective. But the scene camera changes between perspective and orthogonal depending if it is in the initial positions or it is rotated or translated.

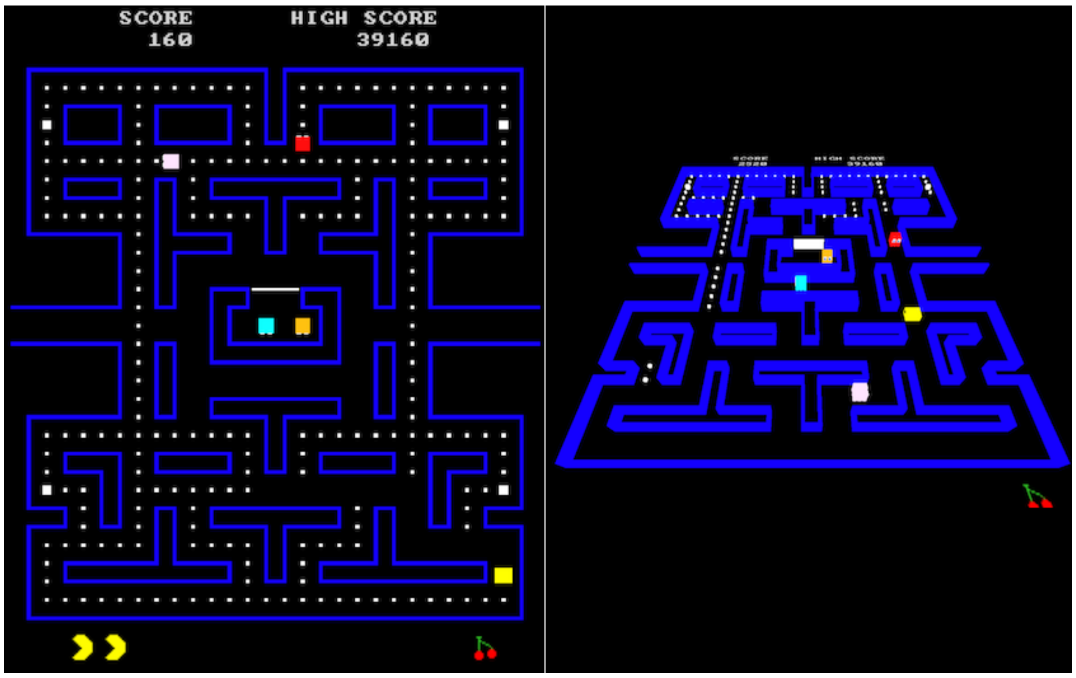

Figure 3.5: Left orthogonal camera and right perspective camera.

The cameras of the different characters are in first person and there is a sign at the top of the screen notifying who the owner of this camera is.

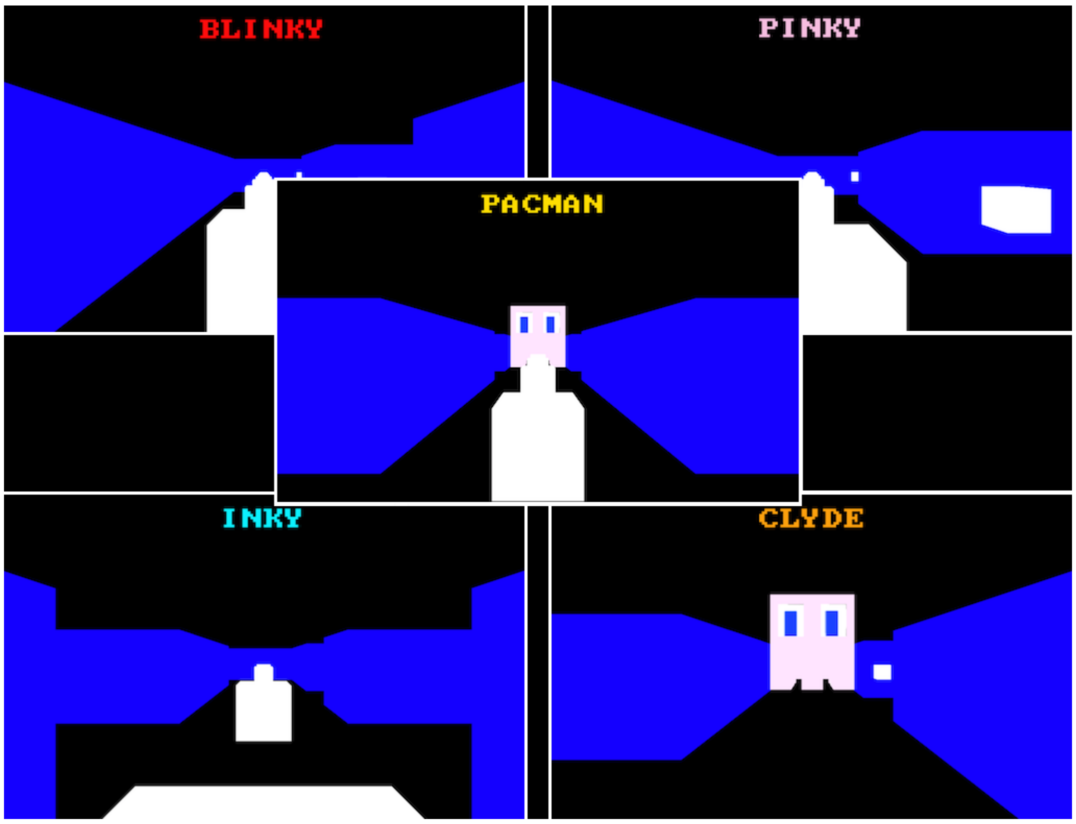

Figure 3.6: The different first person cameras.

CHAPTER 4 REINFORCEMENT LEARNING

In this chapter we are going to explain the development of the reinforcement learning system, the differences between the Pac-Man's role and the ghosts' role, and the different learning models.

4.1 Implementation

The implementation of these methods is very simple. It consists in determining what the current state is. Given this state, it chooses the action according to the policy of the model which works with the Q-values of each action available in the state. Q-value is a real value assigned to a relation of a state and an action. Once the action is chosen the AI determines which is the most appropriate direction for the action given. The next time to choose the following action, it will update the Q-value for the previous state and action given the current state obtained and the formula expressed in *Figure 4.1.* Given the current state, the process will start again choosing the next action.

$$QV_{(t+1)}(s_t, a_t) = (1-\alpha)\cdot QV_t(s_t, a_t) + \alpha \cdot (R(s_t, a_t) + V_t(s_{t+1}) - QV_t(s_t, a_t))$$

$QV_{(t+1)}(s_t, a_t)$	is the new Q-value given the old state s_t and old action a_t.
α	is the learning rate.
$R(s_t, a_t)$	is the reward value given the old state s_t and old action a_t.
$QV_t(s_t, a_t)$	is the old Q-value given the old state s_t and old action a_t.
$V_t(s_{t+1})$	is the learning value given the new state s_{t+1}.

Figure 4.1: Formula to get the new Q-value.

4.2 Structure

The system of classes of the Reinforcement Learning is gathered in the category **RL**. In this category we will find the following classes:

- **Action.** It contains the representation of an action. The actions are the decisions that the character is able to do.
- **LearningModel.** It contains the matrix with Q-values and the reward for each action of every state. It also has the methods to apply learning to Q-value.
- **DoubleQLearning.** It is a child of LearningModel adapted with a method to learn with the Q-value evaluated by the learning model Double Q-Learning.
- **ExpectedSARSA.** It is a child of LearningModel adapted with a method to learn with the Q-value evaluated by the learning model Expected SARSA.
- **QLearning.** It is a child of LearningModel adapted with a method to learn with the Q-value evaluated by the learning model Q-Learning.
- **SARSA.** It is a child of LearningModel adapted with a method to learn with the Q-value evaluated by the learning model SARSA.
- **State.** It contains the representation of a state. The states are the status that the character can be.
- **Value.** It contains the representation of a set of values. It is used to represent Q-values and rewards.

Also, there is a category required, **Agent**. This category contains the agents which will be interacted with the ghosts and Pac-Man AI. We can find the following classes:

- **Agent.** It contains all the methods to query the current action given the current state and the methods to learn given the next state.
- **AgentGhost.** It is a child of Agent, it has the definition of the graph for each ghost and the methods to query what the current state is.
- **AgentPacman.** It is a child of Agent, it has the definition of the graph for Pac-Man and the methods to query what the current state is. Also, it contains the methods to train the model.

4.3 Pac-Man

The agent of Pac-Man is queried for each step. That is because Pac-Man can turn to the opposite direction whenever it wants, and it always has two directions to choose. The agent of Pac-Man returns to an abstract action. The following list explains how Pac-Man chooses the next direction given each abstract action:

- **Attacking.** Pac-Man remembers the nearest frightened ghost's position and chooses the direction which takes him to the shortest path.

- **Fleeing.** Pac-Man recollects the distance from all the non frightened ghosts and takes the average distance for each available direction. Finally, Pac-Man chooses the direction with the largest average distance, or in case of ending in a tie, the direction that is closer to the nearest dot.
- **Going to a dot.** Pac-Man remembers the nearest dot's position and decides on direction like **Attacking** choice.
- **Going to an energiser.** Pac-Man remembers the nearest energiser's position and chooses the direction like **Going to dot** choice.
- **Going to the fruit.** Pac-Man recollects where the fruit is and picks the direction like **Going to energiser** choice.

There is another abstract action, **End**, but it is reserved for ending states. These states are able to value the events appeared suddenly such as when Pac-Man dies.

The definition of the graph is *nondeterministic* since the states are dependent to the position of the ghosts. Because of that, the graph does not define a destination for a particular state and action. Its destination is checked in the execution time. The following list enumerates the different states and their actions with their reward values. The order is the same as their priority to be assigned in the current turn:

- **Close to a ghost:** This state is shown when Pac-Man is a maximum of four cells from the closest non frightened ghost. The actions available are:
  - Attacking with 5 points.
  - Fleeing with -2 points.
- **Close to a frightened ghost:** This state is shown when Pac-Man is a maximum of four cells from the closest frightened ghost and at least five cells from the closest non frightened ghost. The possible actions are:
  - Attacking with -2 points.
  - Fleeing with 5 points.
- **Far from a frightened ghost:** This state is shown when Pac-Man is at least five cells from the closest frightened ghost. The possible actions are:
  - Attacking with -1 point.
  - Fleeing with 5 points.
- **Far from a ghost and close to a dot:** This state is shown when Pac-Man is a maximum of four cells from the closest dot and at least five cells from the closest non frightened ghost. The possible actions are:
  - Attacking with 5 points.
  - Fleeing with 0 points.
  - Going to a dot with -1 point.
- **Far from a ghost and close to an energiser:** This state is shown when Pac-Man is a maximum of four cells from the closest energiser and at least five cells from the closest non frightened ghost. The possible actions are:

  - Attacking with 5 points.
  - Fleeing with 0 points.
  - Going to an energiser with -1 point.

- **Far from a ghost and close to the fruit:** This state is shown when Pac-Man is a maximum of four cells from the fruit and at least five cells from the closest non frightened ghost. The possible actions are:
  - Attacking with 5 points.
  - Fleeing with 0 points.
  - Going to the fruit with -1 point.
- **Far from a ghost, far from a dot and there are many dots:** This state is shown when there are at least 25 dots and Pac-Man is a maximum of four cells from the closest dot and at least five cells from the closest non frightened ghost. The possible actions are:
  - Attacking with -1 point.
  - Fleeing with 0 points.
  - Going to a dot with 4 points.
- **Far from a ghost, far from a dot and there are few dots:** This state is shown when there are a maximum of 24 dots and Pac-Man is a maximum of four cells from the closest dot and at least five cells from the closest non frightened ghost. The possible actions are:
  - Attacking with -1 point.
  - Fleeing with 0 points.
  - Going to a dot with 4.5 points.
- **Kill:** This state is shown when Pac-Man eats a frightened ghost. It is a terminal state with 5 points.
- **Dead:** This state is shown when Pac-Man is caught by a non frightened ghost. It is a terminal state with -5 points.
- **Dot taken:** This state is shown when Pac-Man eats a dot. It is a terminal state with 5 points.
- **Energiser taken:** This state is shown when Pac-Man eats an energiser. It is a terminal state with 5 points.
- **Fruit taken:** This state is shown when Pac-Man eats the fruit. It is a terminal state with 5 points.

Owing to the requirements of Pac-Man's role, the agent of Pac-Man needs some methods to train, allowing Pac-Man to begin with different playing levels (beginner, medium or expert). Those training methods are focused on learning to die, kill and execute each abstract action. Because of that, there are seven different training methods. When we choose beginner level, the agent does not apply any training method. In medium level case, the agent uses three training methods. But the choice of these three methods are randomised, this means that maybe Pac-Man focus its training on learning to kill or other training. Finally, when we choose expert level, the agent calls three times each training method.

4.4 Ghosts

Each ghost queries its agent when it satisfies two conditions:

- The ghost is located in position where it can go in three different directions without colliding with a wall.
- The random factor is less than a random number generated. The random factor is decreasing due to some events appeared such as the ghost having been eaten by Pac-Man.

Like in the case of Pac-Man, the agent of each ghost returns an abstract action. The following list explains how the ghosts make the choice to get the direction for the current turn given each abstract action:

- **Attacking.** The ghosts remember where their target is located and choose the direction which takes them to the shortest path to Pac-Man.
- **Fleeing.** The ghosts remember where Pac-Man is located and check which direction takes them farther from Pac-Man.

Just as Pac-Man, there is another abstract action, **End**, reserved for terminal states. The target changes for each ghost:

- **Blinky (red ghost):** Its target is Pac-Man.
- **Pinky (pink ghost):** Its target is four cells in front of Pac-Man's position. If the position is not accessible, it takes the closest position accessible from its target.

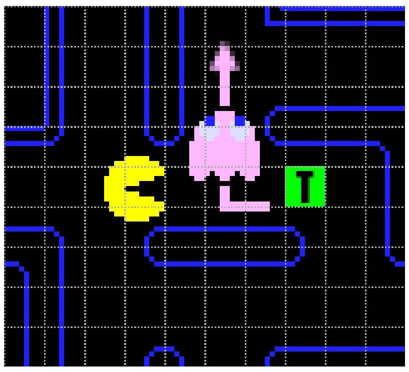

Figure 4.2: Pinky's target.

- **Inky (blue ghost):** Its target is related to Blinky's position and Pac-Man's position. The target is the opposite direction of Blinky given two cells in front of Pac-Man's position.

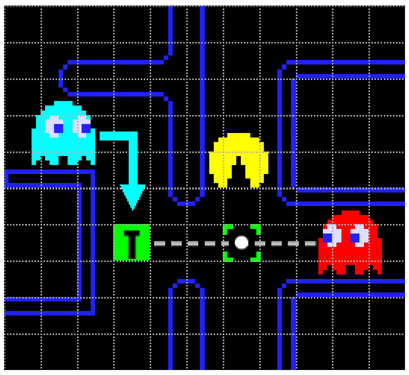

Figure 4.3: Inky's target.

- **Clyde (orange ghost):** As Blinky, its target is Pac-Man.

Like in the case of Pac-Man, the definition of the graph is *nondeterministic* for the same reason. In this case all ghosts have the same graph, but Clyde like the Puck-Man game has a particular way of thinking. For that reason, it has different reward values in its graph. The following list shows the different states with its actions and values. The order of the list is the same as the priority of each state:

- **Far from Pac-Man and frightened:** This state is shown when the ghost is frightened and it is at least 10 cells from Pac-Man. The actions available are:
  - Attacking with -5 points (5 points in Clyde version).
  - Fleeing with 5 points (-5 points in Clyde version).
- **Close to Pac-Man and frightened:** This state is shown when the ghost is frightened and it is a maximum of 9 cells from Pac-Man. The actions available are:
  - Attacking with -10 points (as in Clyde version).
  - Fleeing with -5 points (as in Clyde version).
- **Close to Pac-Man, non frightened, close to an energiser and few dots:** This state is shown when the ghost is non frightened, it is a maximum of 9 cells from Pac-Man, Pac-Man is closer than 10 cells from an energiser and there are less than 75 dots. The possible actions are:
  - Attacking with 5 points (1 point in Clyde version).
  - Fleeing with 1 point (5 points in Clyde version).
- **Close to Pac-Man, non frightened, close to an energiser, many dots and no ghosts:** This state is shown when the ghost is non frightened, it is a maximum of 9 cells from Pac-Man, Pac-Man is closer than 10 cells from an energiser, there are 75 dots or more, and there are not any other non frightened ghosts closer than 10 cells from Pac-Man. The possible actions are:

  - Attacking with 7.5 points (0 points in Clyde version).
  - Fleeing with 0 points (7.5 points in Clyde version).

- **Close to Pac-Man, non frightened, close to an energiser, many dots and some ghosts:** This state is shown when the ghost is non frightened, it is a maximum of 9 cells from Pac-Man, Pac-Man is closer than 10 cells from an energiser, there are 75 dots or more, and there are at least one or more non frightened ghosts closer than 10 cells from Pac-Man. The possible actions are:
  - Attacking with 3.75 points (1.25 points in Clyde version).
  - Fleeing with 1.25 points (3.75 points in Clyde version).
- **Close to Pac-Man, non frightened, close to an energiser, many dots and all ghosts:** This state is shown when the ghost is non frightened, it is a maximum of 9 cells from Pac-Man, Pac-Man is closer than 10 cells from an energiser, there are 75 dots or more, and all the rest of non frightened ghosts are closer than 10 cells from Pac-Man. The possible actions are:
  - Attacking with 5 points (as in Clyde version).
  - Fleeing with 5 points (as in Clyde version).
- **Close to Pac-Man, non frightened and far from an energiser:** This state is shown when the ghost is non frightened, it is a maximum of 9 cells from Pac-Man and Pac-Man is 10 cells or farther from an energiser. The possible actions are:
  - Attacking with 10 points (0 points in Clyde version).
  - Fleeing with 0 points (10 points in Clyde version).
- **Far, non frightened and few dots:** This state is shown when the ghost is non frightened, it is at least 10 cells from Pac-Man and there are less than 75 dots. The possible actions are:
  - Attacking with 5 points (10 points in Clyde version).
  - Fleeing with 0 points (0 points in Clyde version).
- **Far from Pac-Man, non frightened and many dots:** This state is shown when the ghost is non frightened, it is at least 10 cells from Pac-Man and there are 75 dots or more. The possible actions are:
  - Attacking with 3.75 points (5 points in Clyde version).
  - Fleeing with 1.25 points (as in Clyde version).
- **Kill:** This state is shown when the ghost catches Pac-Man. It is a terminal state with 10 points (as in Clyde version).
- **Dead:** This state is shown when the ghost is eaten by Pac-Man. It is a terminal state with -10 points (as in Clyde version).

4.5 Learning models

There are several learning models in Reinforcement Learning methodologies. Each learning model works with different learning values given the next state entered. In this particular project, we work with four models:

- **Q-Learning.** It works with the maximum Q-value of the state.

$$V_t(s_{t+1}) = \gamma \cdot \max_a QV_t(s_{t+1}, a)$$

$V_t(s_{t+1})$	is the learning value given the state s_{t+1}
γ	is the discount factor
$QV_t(s_{t+1}, a)$	is the Q-value given the state s_{t+1} and action a

Figure 4.4: Formula to get the learning value of the Q-Learning model.

- **SARSA.** It works with the Q-value of an action predicted according to the state and the policy of the learning model.

$$V_t(s_{t+1}) = \gamma \cdot QV_t(s_{t+1}, a_{t+1})$$

$V_t(s_{t+1})$	is the learning value given the state s_{t+1}
γ	is the discount factor
$QV_t(s_{t+1}, a_{t+1})$	is the Q-value given the state s_{t+1} and action predicted a_{t+1}

Figure 4.5: Formula to get the learning value of the SARSA model.

- **Double Q-Learning.** It works similarly to Q-Learning. But this learning model uses two Q-values. For each query there is a random factor (with the same percentage for each Q-values) that determines which Q-value will be updated in the current turn. For this particular learning model, the choice is made keeping in mind both Q-values.

$$V_t(s_{t+1}) = \gamma \cdot QV_t(s_{t+1}, a*)$$

$V_t(s_{t+1})$	is the learning value given the state s_{t+1}
γ	is the discount factor
$QV_t(s_{t+1}, a*)$	is the Q-value given the state s_{t+1} and action $a*$ from the Q-values chose to be updated in the current round.
$QV'_t(s_{t+1}, a*)$	is the maximum Q-value given the state s_{t+1} from the Q-values didn't choose to be updated in the current round.

Figure 4.6: Formula to get the learning value of the Double Q-Learning model.

- **Expected SARSA.** It works in a similar way to SARSA. But this learning model calculates the learning value adding the different Q-values of the state with a weight determined by the policy of the learning model, instead of predicting the next action according to the policy.

$$V_t(s_{t+1}) = \gamma \cdot \sum_a (\pi_t(s_{t+1}, a) \cdot QV_t(s_{t+1}, a))$$

$V_t(s_{t+1})$	is the learning value given the state s_{t+1}
γ	is the discount factor
$QV_t(s_{t+1}, a)$	is the Q-value given the state s_{t+1} and action a
$\pi_t(s_{t+1}, a)$	is the value of the policy given the state s_{t+1} and action a

Figure 4.7: Formula to get the learning value of the Expected SARSA model.

4.6 Parameters adjusting

We have to adjust the parameters according to reach the best configuration of the learning models to take better decisions. The following parameters are the parameters to adjust for both roles:

- **Alpha (α).** This value can be between 0 and 1. When this value is 0 there is no learning and it always uses the initial Q-value, when this value is 1 the learning does not keep any percentage of the Q-value. This can be translated as if it is near to 0 the Q-value will not change much and if it is near to 1 the Q-value will change very much.
- **Gamma (γ).** This value can be between 0 and 1. When this value is 0 the learning will ignore the value given by the learning model and it will learn only with the reward. When this value is 1 the learning will use the full value given by the learning model.
- **Number steps to decrease α.** The α value is reduced for each n decisions to avoid over passing the knowledge.

We have a particular parameter for the ghosts' role, **Speed reduction for random factor.** The factor random begins with 50 percent. So, there is a 50 percent to take a random direction and other 50 percent to ask the agent. This factor is reduced for each element eaten by Pac-Man, depending on the element the reduction will be higher or lower. There is a weight to manage the speed reduction.

The process to adjust the parameters begins with the Pac-Man's role plays. First of all, we analyse the results given the different levels versus a ghost with a simple intelligence (without using machine learning). This simple intelligence is composed of a singular target, attacking Pac-Man when the ghosts are non frightened and fleeing from Pac-Man when they are frightened. The best values for the different parameters are the following:

- **Alpha (α).** The value is 0.25 at the beginning. Before training on medium and expert level, the value reduces to 0.23 and 0.20 respectively.
- **Gamma (γ).** The value is 0.7.
- **Number steps to decrease α.** It reduces 0.01 to alpha for each 10 decisions taken.

Once the parameters for the Pac-Man's role are adjusted, we can process to adjust the parameters for the ghosts' role. The ghosts will be executed against the different playing levels of the official version of Pac-Man with the best values introduced in its parameters. The best values identified for the ghosts' role are the following:

- **Alpha (α).** The value is 0.25 at the beginning.
- **Gamma (γ).** The value is 0.75.
- **Number steps to decrease α.** It reduces 0.01 to alpha for each 10 taken decisions.
- **Speed reduction for random factor.** The value assigned to each event was quite good, because of that the weight was valuable as 1.

CHAPTER 5 NAVIGATION AND 3D MODELS

In this chapter, we are going to explain both navigation and 3D models developments.

5.1 Navigation

In this section, we are going to explain the different features of the navigation. This navigation is only available in simulation mode. As we said previously, the navigation consists of six cameras, five for the characters and one for the scene.

5.1.1 Scene camera

The scene camera is changing between orthogonal and perspective as long as it is in the initial position or it is translated or rotated. When the camera is in the initial position, it is the same point of view as the game mode. Thus, if we do not press any key to translate or rotate we will see the whole simulation as game mode. The translation and rotation interaction are done based on accelerations. For that reason, when we press some button to interact, it begins to accelerate. If we keep pressing the button it will accelerate more and more until arriving to the limit speed. When we let the button be pressed it will stop slowly. Thus, all navigation, translation and rotation, are smoothed.

5.1.2 Characters camera

These cameras are perspective and are located in front of each character as a first person. Moreover, there is a title on the top of the camera for the purpose of being able to identify the character of the current first person camera. They cannot be translated or rotated, but they move themselves with the character assigned. The characters' rotation was changed (instantly rotation by smoothly rotation) because the cameras in first person could not be followed by the human eye.

5.2 3D Models

In this section, we are going to explain the models that have been designed and the other models that have been provided from simple subsection.

5.2.1 Simple models

These models are created directly from Unity, they consist of the union of cubes and their texture is a uniform colour. The following list enumerates the different elements which were designed with this system:

- **Walls:** There are different kinds of walls: planes and corners. The planes are simple cubes with the scale 10x1x5. The corners are the union of two half planes with scale 5x1x5, which are joined in the extreme of both planes.

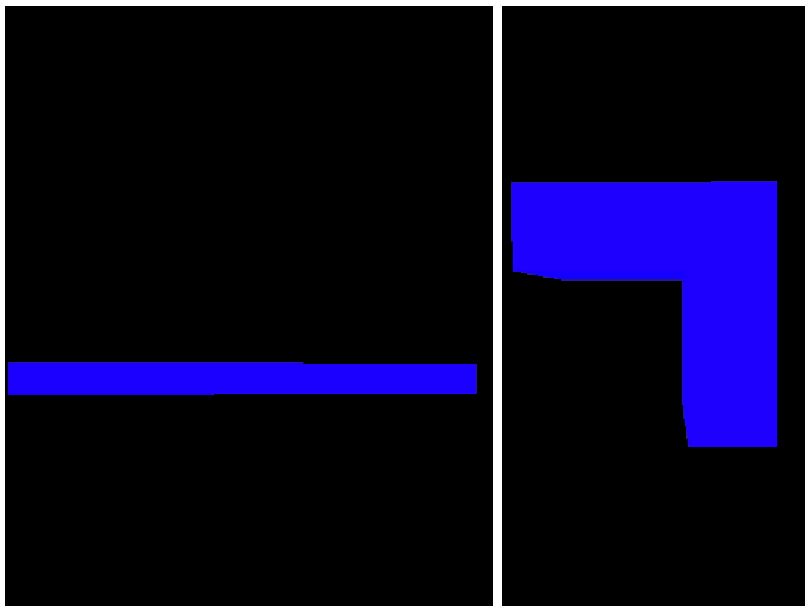

Figure 5.1: Left plane and right corner, they are in different scale.

- **The ghosts' door:** The ghosts' door is similar to the plane. But in this case, the scale is 8x0.5x5 and the texture is white.
- **Dot:** The dots are simple cubes with scale 1x1x1, their texture is white.
- **Energiser:** The energisers are the same shape as the dot but with scale 3x3x3, their texture is also white.

5.2.2 Models designed

The majority of models designed have a simple design. We take advantage of the fact that we focus the design of the game on pixel design to compound the model from cubes and modify some of their vertexes. The following list enumerates the different elements which were designed by ourselves:

- **Pac-Man:** It consists of a cube which represents the shape of the head and other two small cubes which depict the shape of the eyes:

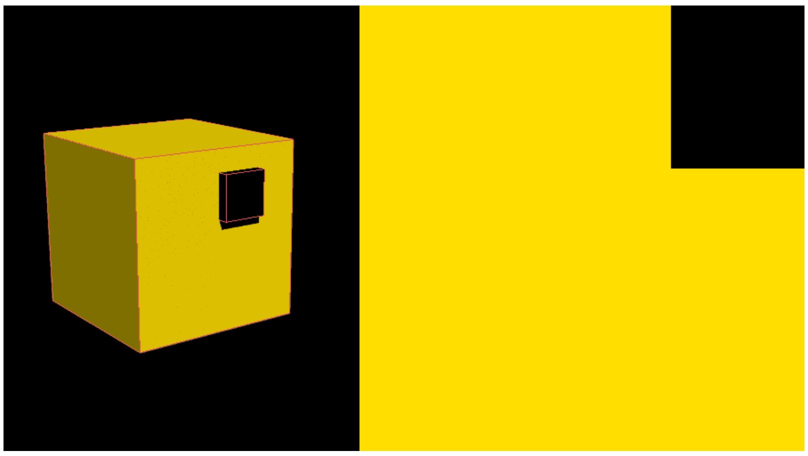
Figure 5.2: Left model of Pac-Man and right texture of Pac-Man.

- **Ghost:** It consists of a cube representing the body, two small cubes which simulate the eyes and finally nine medium cubes which make the bottom of the ghost. There is a texture for each ghost, each texture has its particular body colour:

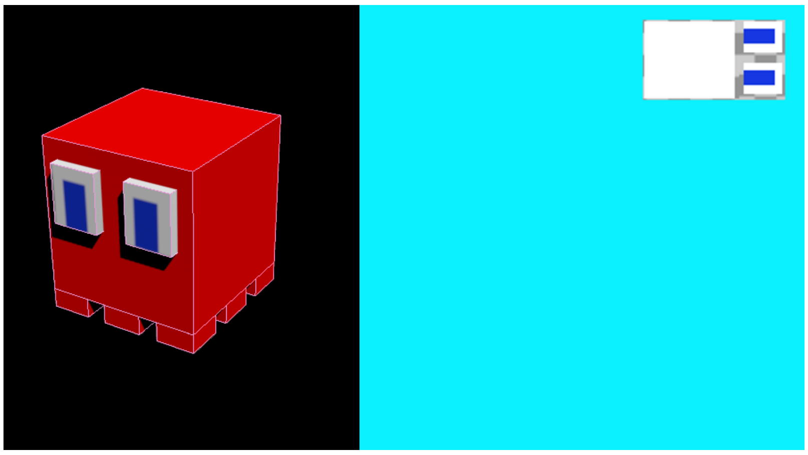
Figure 5.3: Left model of Blinky (red ghost) and right texture of Inky (blue ghost).

- **Cherry:** It consists of two cubes which depict the shape of a cherry and other long cubes which create the tail of the fruit:

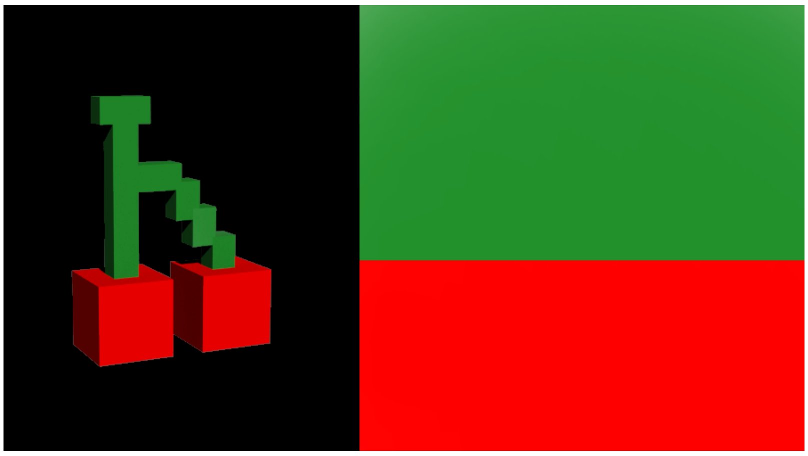
Figure 5.4: Left model of cherry and right texture of cherry.

- **Strawberry:** It consists of five cubes which make the shape of the strawberry and one last which makes its tail:

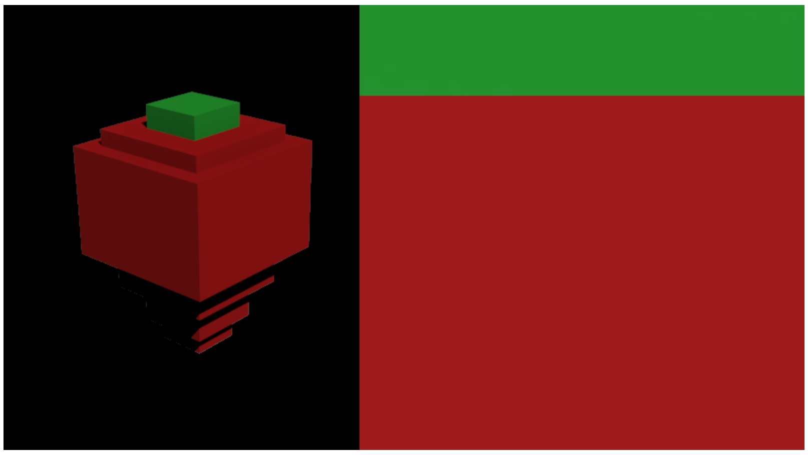
Figure 5.5: Left model of strawberry and right texture of strawberry.

- **Orange:** It consists of three cubes which depict the shape of the orange, one cube which makes the shape of the tail and one last which makes the shape of the leaf:

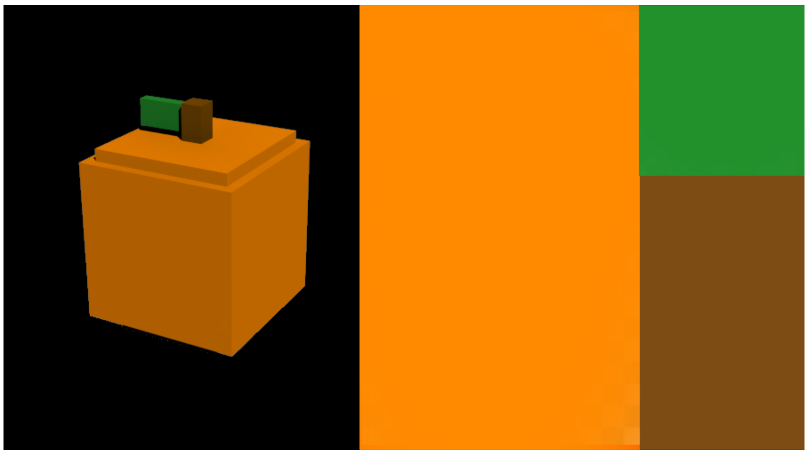

Figure 5.6: Left model of orange and right texture of orange.

- **Apple:** It consists of nine cubes which make the shape of the apple and two cubes which depict the shape of the tail:

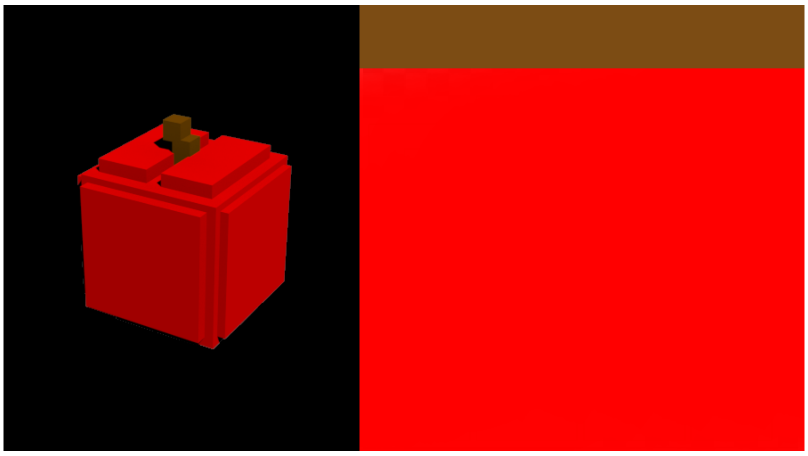

Figure 5.7: Left model of apple and right texture of apple.

- **Pineapple:** It consists of three cubes which make the shape of the pineapple and other three which make the shape of the tail:

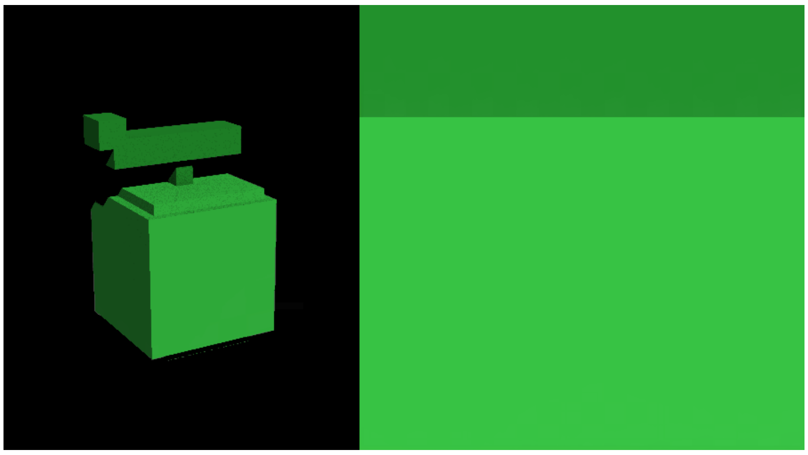

Figure 5.8: Left model of pineapple and right texture of pineapple.

- **Galaxian spaceship:** It consists of several squares divided into the base, the wings and the back of the space ship:

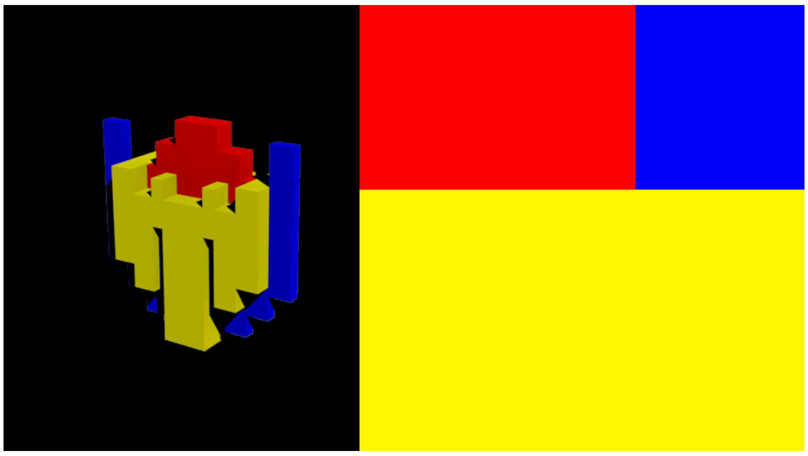

Figure 5.9: Left model of galaxian spaceship and right texture of galaxian spaceship.

- **Bell:** It consists of four cubes which make the shape of the bell:

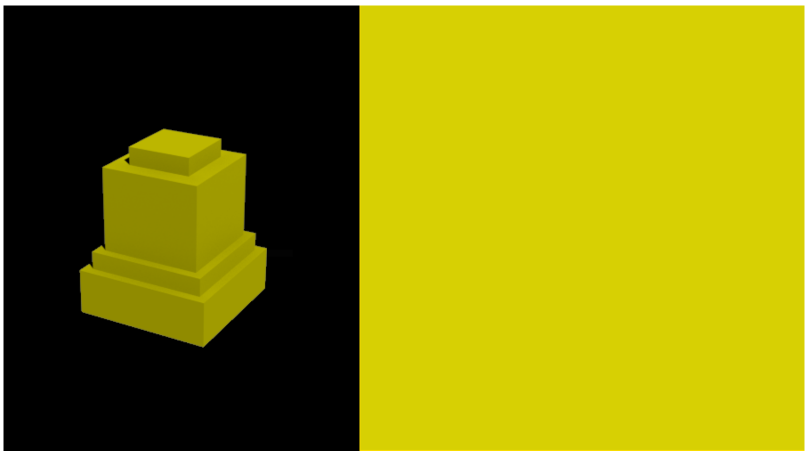
Figure 5.10: Left model of bell and right texture of bell.

- **Key:** It consists of one cube which makes the shape of the handle and another one which makes the shape of the key:

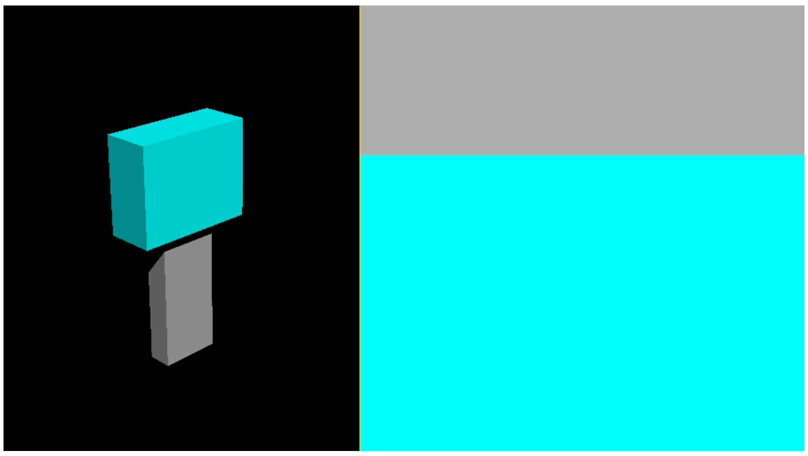
Figure 5.11: Left model of key and right texture of key.

CHAPTER 6 VALIDATION

In this chapter, we are going to expose the results obtained in both tests executed and we will take some conclusions with regard to the results.

6.1 Simulation

The target of this test is to choose which learning model is better for all three kinds of users (beginner, medium and expert). This test consists in executing 30 times the simulation for each learning model along with each playing level of Pac-Man. So, there will be 360 executions. Then, we will join the executions in sets by playing level and learning model taken the average result of the following factors:

- Final score.
- Number of dots eaten.
- Number of ghosts eaten.
- Number of deaths.
- Total time.

We obtain quite variability in the results of the different learning models. We can see graphically the results in the appendix *A Simulation Test Results.* We are going to explain the result for each learning model below:

- **Double Q-Learning:** In this model, the beginner level seems to have had more difficulties than medium and expert levels. We have a difference in the score of 2200 between beginner and medium and 500 between medium and expert, this can be assessed graphically with *Figure A.1.* Focusing on the score, we can notice that it seems not balanced. The same thing happens with the rest of the factors. It is right for the medium level to be closer from beginner to expert depending on its training. As a curiosity, medium and expert levels die the same number of times. Also, we can see that beginner level does not eat a lot of dots in the second stage, due to the fact that the stage has 244 dots and it eats (in average) 252 dots.

Factor	Beginner	Medium	Expert
Final score	6581	8842	9377
Number of dots eaten	252	315	349
Number of ghosts eaten	9	12	13
Number of deaths	3.14	3.34	3.34
Total time	89.71	113.50	114.11

Table 6.1: Results of the simulation test in Double Q-Learning execution.

- **Expected SARSA:** In this model, the expert level seems to have had rather easier executions than beginner and medium level. In fact, both levels (beginner and medium) have quite similar results. They die the same number of times.

Factor	Beginner	Medium	Expert
Final score	6377	6777	8572
Number of dots eaten	270	285	337
Number of ghosts eaten	8	9	11
Number of deaths	3.17	3.17	3.31
Total time	100.96	104.22	119.45

Table 6.2: Results of the simulation test in Expected SARSA execution.

- **Q-Learning:** In this model we can see something weird, medium level has the worst executions. It is true that this project tries to balance the difficulty, but it does not make sense the difference between beginner and expert (extremely small) and then the difference between medium and the other two levels (quite big).

Factor	Beginner	Medium	Expert
Final score	7648	6254	7698
Number of dots eaten	305	274	324
Number of ghosts eaten	10	8	10
Number of deaths	3.17	3.10	3.14
Total time	115.93	106.09	112.24

Table 6.3: Results of the simulation test in Q-Learnig execution.

- **SARSA:** In this model we can relatively increase the success among the different levels. We can see how the difference in the final score is more similar than the other versions. The results of this model seem rather realistic since it is obvious that the expert will reach better punctuation because of its initial knowledge, despite trying to balance the difficulty. Because of that, it begins with advantage. But as we said, the results are balanced which is good news.

Factor	Beginner	Medium	Expert
Final score	6618	7296	8002
Number of dots eaten	290	314	340
Number of ghosts eaten	8	10	11
Number of deaths	3.14	3.10	3.21
Total time	106.06	117.69	117.99

Table 6.4: Results of the simulation test in SARSA execution.

Given the results, we think that the learning model SARSA is more appropriate for being assigned as official for the final application. For that reason, the test with real users will be executed with SARSA.

6.2 Real users

The process of this test is executed following the next steps:

1. Reading the document located in the top of the document of the test which can be found in the appendix *B SMART PACMAN Test Document*.

2. Playing one match in Puck-Man (the original Pac-Man).

3. Playing one match in SMART PACMAN (the version of the project).

4. Answering the questionnaire located below the text of the first step.

We can find the answers of the questions in the appendix *C Results of Real Users Test*. Given the answers, we can observe that the people selected to make the test is considerably variable according to their user experience regarding games. There are people who play every day and others who do not play at all. Also, we noticed that the majority of the selected people are not really into Pac-Man games. But they enjoyed playing this version. Besides, all users noticed that the difficulty was increasing and most of them thought our version has a more appropriate difficulty fitted for their level.

CHAPTER 7 CONCLUSIONS

We can gladly say that this project has achieved all the objectives planned inside the time allotted. We found a few issues during the development yet none of them made the end of the development fall behind.

The first task of the development, the platform of Pac-Man, was successfully completed with the majority of features from the original Pac-Man.

The second task of the development, the machine learning of Pac-Man with different levels, was fortunately completed with the planned expectations, in spite of the fact that our requirements were only three kinds of playing levels (beginner, medium and expert). We achieved a beginner as a user who has to learn everything, forcing it to learn what is bad and what is good, one example is that it sometimes goes to a non frightened ghost given its ignorance. We achieved medium as a quite variable user. Depending on its training, we can reach a level nearly to beginner or expert. Finally, we achieved expert as a user who knows what is bad and what is good.

The third task of the development, the machine learning model of the ghosts, was victoriously finished. Having in mind the results of the questionnaire, we can notice that we satisfy our expectations about developing a machine learning which adapts its level according to the user level. Even though the sample of users is small (ten users), all ten people noticed the increase in the difficulty and most of them thought that the level suggested was appropriate.

The fourth task of the development, the navigation, was extremely fast to develop, it did not suppose a great effort implementing it at all. Although its contribution in the project was not really used in the end, it clearly has great possibilities in front of new features for future projects.

We cannot get a clear conclusion about whether users enjoy more with a complexity adjusted to their level or not. However, our little sample of users seems to show that

SMART PACMAN has had more preference than Puck-Man.

Finally, we have to consider new factors to keep in mind for future projects. Because of that, the last question of our questionnaire can help us consider some of these new factors. Some users agree that there should be a good practice trying to use other variables to make decisions, such as keeping in mind the direction of Pac-Man or the direction of other ghosts. We could try to make a study with different graphs trying to analyse which factors make the ghosts involve in a better way. Also, we could try to add other extra features, such as different stages, more or fewer ghosts, multiple Pac-Man and the rest, to analyse which features cause more effect to the user experiences. The last point to bear in mind is the application of our reinforcement learning system to other kinds of games according to check our study in other kinds of platforms.

CHAPTER

8 REFERENCES

[1] Jamey Pittman. The pac-man dossier. `http://home.comcast.net/~jpittman2/pacman/pacmandossier.html`, 2011. Last visited on 08/03/2015.

[2] Namco. Pac-man game scoring. `http://www.pacmanmuseum.com/history/pacman-scoring.php`, 2009. Last visited on 08/03/2015.

[3] Collins English Dictionary. The free on-line dictionary of computing. `http://dictionary.reference.com/browse/machinelearning`, 2015. Last visited on 09/03/2015.

[4] James Paul Gee. Learning by design: Good video games as learning machines. *University of Wisconsin-Madison*, (6. Pleasantly Frustrating):13–15, 2005.

[5] Wikipedia Community. Black & white (video game) - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Black_%26_White_%28video_game%29`, 2014. Last visited on 10/03/2015.

[6] Alex J. Champandard. Top 10 most influential ai games aigamedev.com. `http://aigamedev.com/open/highlights/top-ai-games/`, 2007. Last visited on 10/03/2015.

[7] Wikipedia Community. Game engine - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Game_engine`, 2015. Last visited on 09/03/2015.

[8] Wikipedia Community. 3d modeling - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/3D_modeling`, 2015. Last visited on 09/03/2015.

[9] Wikipedia Community. Rendering (computer graphics) - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Rendering_(computer_graphics)`, 2015. Last visited on 09/03/2015.

[10] Google. Google play. `https://play.google.com`, 2015. Last visited on 09/03/2015.

[11] Wikipedia Community. Sponsor (commercial) - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Sponsor_(commercial)`, 2015. Last visited on 09/03/2015.

[12] John DeNero and Dan Klein. Teaching introductory artificial intelligence with pac-man. *Computer Science Division, University of California, Berkey*, 2001.

[13] Nikolaos Tziortziotis, Konstantinos Tziortziotis, and Konstantinos Blekas. Play ms. pac-man using an advanced reinforcement learning agent. *University of Ioannina, Greece*, 2014.

[14] Gergios N. Yannakasis and John Hallam. Evolving opponents for interesting interactive computer games. *The University of Edinburgh and The University of Southern Denmark*, 2004.

[15] Javier Béjar. Machine learning. `http://www.cs.upc.edu/~bejar/apren/docum/trans/01-apind-eng.pdf`, 2013. Last visited on 12/03/2015.

[16] Blizzard Entertainment. Starcraft ii official game site. `http://us.battle.net/sc2/en/`, 2015. Last visited on 12/03/2015.

[17] Quentin Gemine. Imitative learning for designing intelligent agents for video games. *Faculty of Applied Sciences, Institute Montefiore, Belgium*, 2011-2012.

[18] Heysem Kaya, Pinar Tüfekci, and Fikret S. Gürgen. Local and global learning methods for predicting power of a combined gas & stream turbine. *International Conference on Emerging Trends in Computer and Electronics Engineering, Dubai*, 2012.

[19] Juan Carlos Niebles, Hongcheng Wang, and Li Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *Princeton University, USA and United Technologies Research Center, USA*, 2007-2008.

[20] Tim Eden, Anthony Knittel, and Raphael van Uffelen. Reinforcement learning. `http://www.cse.unsw.edu.au/~cs9417ml/RL1/introduction.html`, 2011. Last visited on 11/03/2015.

[21] Rafael Figueroa, Aleksandra Faus, Patricio Cruz, Lydia Tapia, and Rafael Fierro. Reinforcement learning for balancing a flying inverted pendulum. *Department of Electrical and Computer Engineering and Department of Computer Science, University of New Mexico, Albuquerque*, 2014.

[22] Christopher Amato and Guy Shani. High-level reinforcement learning in strategy games. *University of Massachussetts, USA, and Ben-Gurion University, Israel*, 2010.

[23] UBM Tech. Gamasutra - gamasutras gdc 2015 live event coverage. `http://www.gamasutra.com/gdc2015`, 2015. Last visited on 11/03/2015.

[24] Unity Technologies. Unity - game engine. `http://unity3d.com`, 2011. Last visited on 11/03/2015.

[25] Unity Technologies. Unity - unity 4.0. `http://unity3d.com/unity/whats-new/unity-4.0`, 2014. Last visited on 12/03/2015.

[26] Unity Technologies. Unity - manual: Occlusion culling. http://docs.unity3d.com/Manual/OcclusionCulling.html, 2014. Last visited on 12/03/2015.

[27] John Riccitiello. Unity 5 launch – unity blog. http://blogs.unity3d.com/2015/03/03/unity-5-launch/, 2015. Last visited on 12/03/2015.

[28] Unity Technologies. Personal edition. http://unity3d.com/unity/personal-edition, 2015. Last visited on 12/03/2015.

[29] Unity Technologies. Professional edition. http://unity3d.com/unity/professional-edition, 2015. Last visited on 12/03/2015.

[30] Epic Games. Unreal engine technology — home. https://www.unrealengine.com/, 2015. Last visited on 12/03/2015.

[31] Epic Games. Gears of war - home. http://gearsofwar.xbox.com/en-US/, 2015. Last visited on 12/03/2015.

[32] Epic Games. What is unreal engine 4. https://www.unrealengine.com/what-is-unreal-engine-4, 2015. Last visited on 12/03/2015.

[33] Crytek. Cryengine — the complete solution for next generation game development by crytek. http://cryengine.com/, 2015. Last visited on 12/03/2015.

[34] Crytek. Crysis - official site. http://www.crysis.com/, 2013. Last visited on 12/03/2015.

[35] Crytek. Cryengine — crytek. http://cryengine.com/get-cryengine, 2015. Last visited on 12/03/2015.

[36] YoYo Games. Gamemaker: Studio — yoyo games. https://www.yoyogames.com/studio, 2015. Last visited on 12/03/2015.

[37] YoYo Games. Working with 3d. http://docs.yoyogames.com/source/dadiospice/002_reference/drawing/drawing%203d/index.html, 2014. Last visited on 12/03/2015.

[38] Javier Béjar. Aprendizaje. http://www.cs.upc.edu/~bejar/apren/teoria.html, 2011. Last visited on 15/03/2015.

[39] Ikujiro Nonaka and Hirotaka Takeuchi. Scrum (software development). http://en.wikipedia.org/wiki/Scrum_(software_development), 1986. Last visited on 15/03/2015.

[40] Xamarin and the Mono community. Mono develop. http://www.monodevelop.com/, 2015. Last visited on 15/03/2015.

[41] Autodesk. 3ds max - pàgina del software de modelat 3d emprat. http://www.autodesk.com/products/3ds-max/overview, 2014. Last visited on 15/03/2015.

[42] Linus Torvalds. Git - sistema de versions emprat al projecte. http://git-scm.com/, 2005. Last visited on 15/03/2015.

[43] Atlassian. Bitbucket - servidor per al repositori de git. https://bitbucket.org/features, 2008. Last visited on 15/03/2015.

[44] GIMP Development Team. Gimp - the gnu image manipulation program. http://www.gimp.org/, 2014. Last visited on 15/03/2015.

[45] Leslie Lamport. Latex - a document preparation system. http://www.latex-project.org/, 2014. Last visited on 15/03/2015.

[46] GanttProject Team. Ganttproject: free desktop project management app. http://www.ganttproject.biz/, 2015. Last visited on 15/03/2015.

[47] Hado Philip van Hasselt. Insights in reinforcement learning: Formal analysis and empirical evaluation of temporal-difference learning algorithms. *Netherlands Organisation for Scientific Research*, 2011.

[48] Inc. PayScale. Payscale - salary survey, salaries, wages, compensation information and analysis. http://www.payscale.com/research/ES/Country=Spain/Salary, 2015. Last visited on 10/03/2015.

APPENDIX A SIMULATION TEST RESULTS

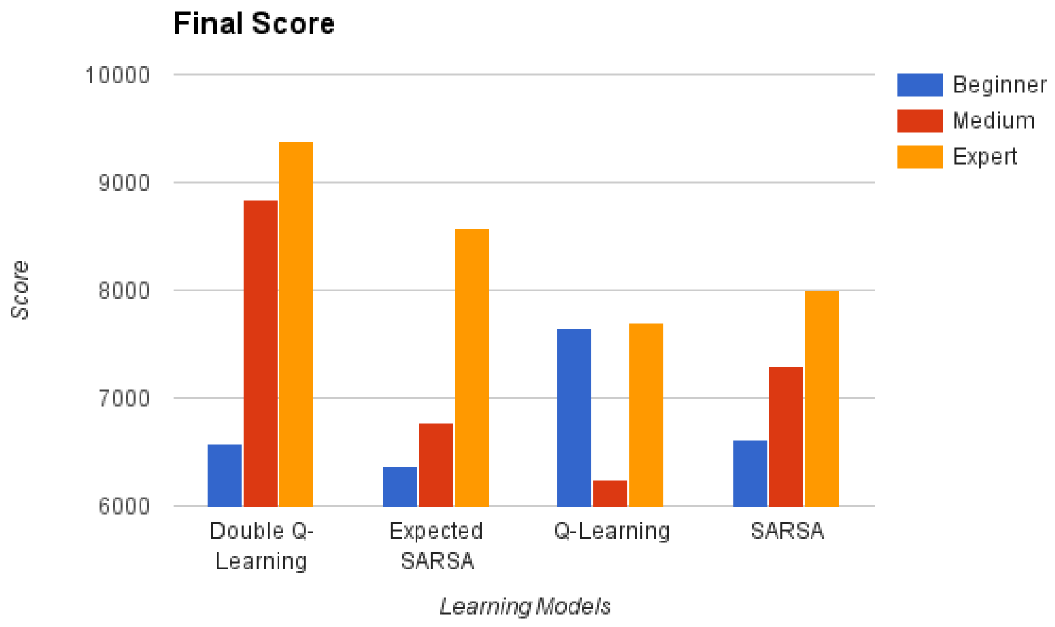

Figure A.1: Final score of the simulation test.

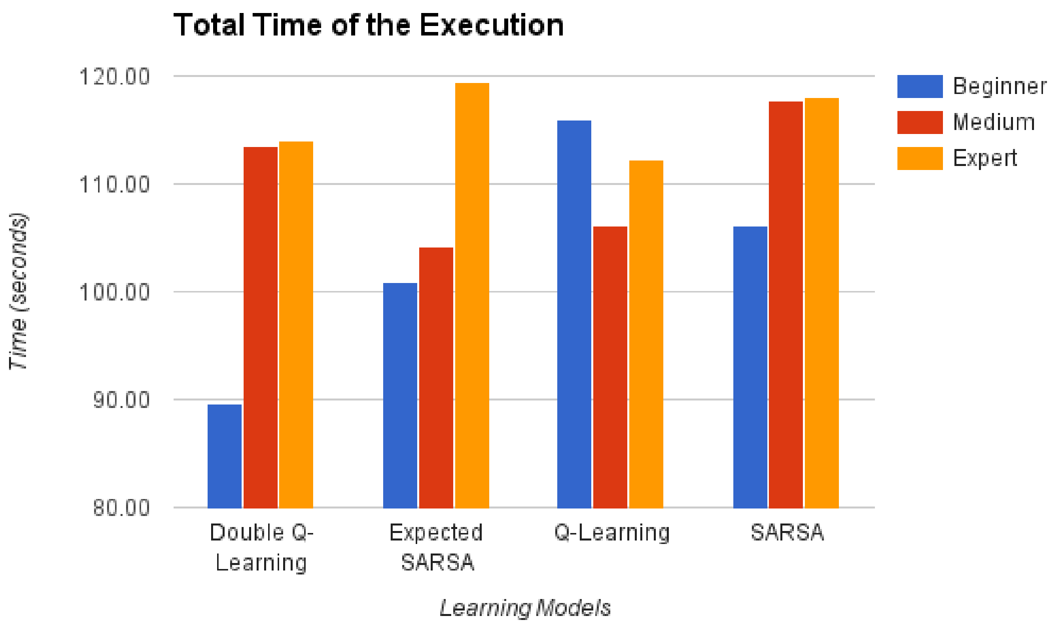

Figure A.2: Total time of the simulation test.

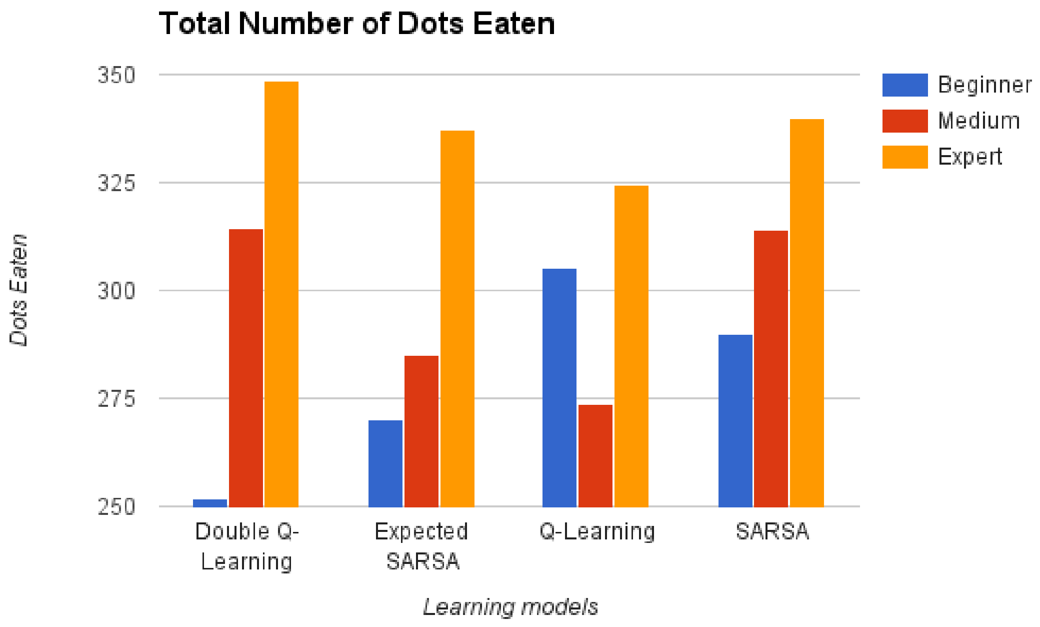

Figure A.3: Total number of dots eaten in the simulation test.

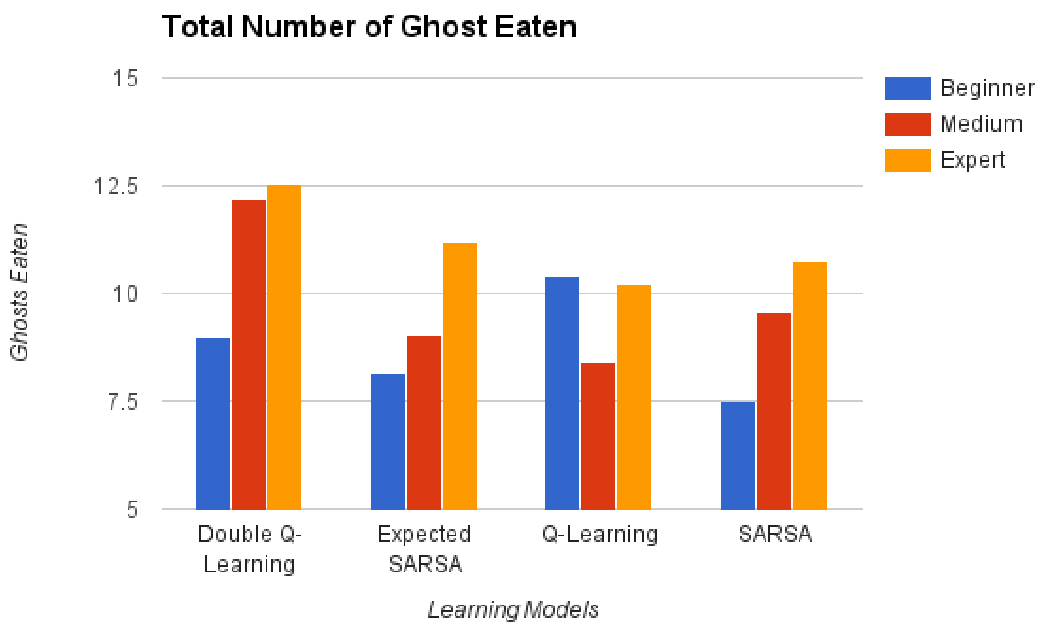

Figure A.4: Total number of ghosts eaten in the simulation test.

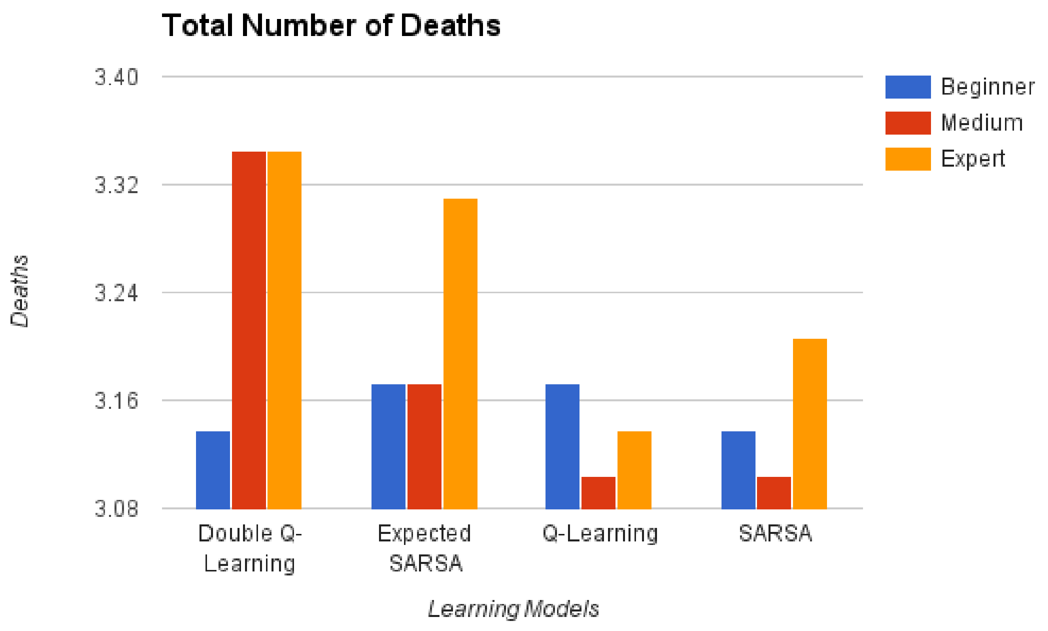

Figure A.5: Total number of deaths in the simulation test.

APPENDIX B SMART PACMAN TEST DOCUMENT

#Test:
Name:

The test consists in making a match in Puck-Man (the original Pacman) and other match in SMART PACMAN (the version of the project). The test finishes with a final questionnaire which is below this explanation. This questionnaire will allow to validate the interaction of the users with the application of the project. SMART PACMAN shares nearly the same features as Puck-Man. The greatest difference between both versions are the variable to change the difficulty. Puck-Man reduces the difference speed between Pac-Man and the ghosts, the time of departure waiting and the time of energy duration for each level passed. SMART PACMAN does not manipulate any variable of speed nor time, but while some events appear the ghosts are learning. Thus, they are making better decisions, getting to the point of corralling us up to a dead end corner.

- Have you ever played Puck-Man?

  1. Only this time.
  2. Sometimes.
  3. Usually.
  4. Constantly.

- Did you like Puck-Man?

  1. No, I didn't.
  2. Not bad.
  3. Yes, I did.
  4. It's the best game ever.

- Do you usually play to video games in your free time?

  1. Never.
  2. Some day per month.
  3. Some day per week.
  4. Every day.

- Which version do you think is more difficult?

  1. SMART PACMAN
  2. Puck-Man.

- Which difficulty do you think is more appropriate for your level?

  1. SMART PACMAN
  2. Puck-Man.

- Did you notice the difficulty was increasing?

  1. Yes, I did.
  2. No, I didn't.

- Did you enjoy playing SMART PACMAN?

  1. No, I didn't.
  2. Not bad.
  3. Yes, I did.
  4. It's the best Pac-Man ever.

- Do you think that the user experience would improve if the project used other additional variables to make the game more difficult?

  1. Yes, I do.
  2. No, I don't.

  In afirmative case, could you suggest someone?

APPENDIX

C

RESULTS OF REAL USERS TEST

Have you ever played Puck-Man?

Only this time	**Sometimes**	**Usually**	**Constantly**
1	9	0	0

Table C.1: Results of the first question of the SMART PACMAN test.

Did you like Puck-Man?

No, I didn't	**Not bad**	**Yes, I did**	**It's the best game ever**
0	7	3	0

Table C.2: Results of the second question of the SMART PACMAN test.

Do you usually play video games in your free time?

Never	**Some day per month**	**Some day per week**	**Every day**
1	3	4	2

Table C.3: Results of the third question of the SMART PACMAN test.

Which version do you think is more difficult?

SMART PACMAN	**Puck-Man**
4	6

Table C.4: Results of the fourth question of the SMART PACMAN test.

Which difficulty do you think is more appropriate for your level?

SMART PACMAN	Puck-Man
8	2

Table C.5: Results of the fifth question of the SMART PACMAN test.

Did you notice the difficulty was increasing?

Yes, I did	No, I didn't
10	0

Table C.6: Results of the sixth question of the SMART PACMAN test.

Did you enjoy playing SMART PACMAN?

No, I didn't	Not bad	Yes, I did	It's the best Pac-Man ever
0	1	8	1

Table C.7: Results of the seventh question of the SMART PACMAN test.