FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Wheelchair Serious Game: A Powerchair Football Video-Game with Machine Learning

Francisco José Sousa Silva

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Paulo Reis

Second Supervisor: Brígida Mónica Faria

October 30, 2020

Wheelchair Serious Game: A Powerchair Football Video-Game with Machine Learning

Francisco José Sousa Silva

Mestrado Integrado em Engenharia Informática e Computação

October 30, 2020

Abstract

Accidents happen frequently and some of them lead to permanent physical disabilities, furthermore these events may also be caused by conditions such as cerebral palsy, multiple sclerosis and others. With problems like this emerging and the population growing older, society becomes more cautious regarding independence and autonomy of these citizens. As a solution to fight this problems, many projects regarding motorized wheelchairs and intelligent wheelchairs have been developed. One of them, which is a big contribution for this dissertation, is the project Intellwheels 2.0.

Intellwheels 2.0 focus in the develop of a framework/kit capable of being implemented in a commercial wheelchair, allow the use of an interface to control not only the smart wheelchair but also multiple other equipment and systems, create a complete semiautomatic system based on computational learning techniques, create a realistic 3D simulator and develop PC serious games.

In this context, this thesis focus in the development of a Powerchair football serious game. The main objective is to develop a powerchair football video-game in 3D with some simulation properties. In addition, the game must allow players to play against intelligent agents. This agents will, therefore, learn to play the game by using machine learning algorithms.

It is expected to achieve an immersive, enjoyable, efficient and fluent video-game.

That said, this document explains how the game was developed taking into account the main objectives, focuses on the development of the game's logic and the realism of the controller that operates the motorized wheelchair.

In addition, it uses machine learning techniques, more concretely, reinforcement learning and imitation learning algorithms, to train intelligent agents capable of playing against the user.

Some techniques involving Delaunay triangulations and Voronoi diagrams were also explored to create high-level behavior for these agents.

The game was subsequently evaluated by users, based on a survey composed of a pre-developed and scientifically assessed questionnaire, the Game Experience Questionnaire. The results obtained were quite satisfactory.

Resumo

Acidentes acontecem com frequência e alguns deles levam a deficiências físicas permanentes. Para além disso, estes eventos também podem ser causados por condições como paralisia cerebral, esclerose múltipla ou outras. Com problemas como estes emergindo e a população envelhecendo, a sociedade torna-se mais cautelosa em relação à independência e autonomia destes cidadãos. Abordado como solução para combater este tipo de problemas, muitos projetos relacionados a cadeiras de rodas motorizadas e cadeiras de rodas inteligentes foram desenvolvidos. Um deles, que é uma grande contribuição para esta dissertação, é o projeto Intellwheels 2.0.

O projeto Intellwheels 2.0 tem como objetivos desenvolver um framework/kit capaz de ser implementado numa cadeira de rodas convencional, permitir o uso de uma interface para controlar não apenas a cadeira de rodas inteligente, mas também vários outros equipamentos e sistemas, cria um sistema semiautomático completo baseado em tecnicas de machine learning, criar um simulador 3D realista e desenvolver jogos sérios.

Neste contexto, esta tese foca no desenvolvimento de um jogo sério de Powerchair Football. O objetivo principal é desenvolver um video-jogo em primeira pessoa de powerchair football em 3 dimensões com algumas propriedades de simulação. Além disso, o jogo deve permitir que os jogadores joguem contra agentes inteligentes. Esses agentes irão aprender a jogar o jogo usando algoritmos de Machine Learning.

Espera-se alcançar um video-jogo imersivo, divertido, eficiente e fluido.

Dito isto, este documento explica como o jogo foi desenvolvido tendo em conta os principais objetivos, foca no desenvolvimento da logica do jogo e no realismo do controlador que opera a cadeira de rodas motorizada.

Para além disso utiliza tecnicas de machine learning, mais concretamente, reinforcement learning and imitation learning, para treinar agentes inteligentes capazes de jogar contra o utilizador.

Algumas técnicas que envolvem triangulações de Delaunay e diagramas de Voronoi foram também explorados para criar um comportamento de alto-nivel para estes agentes.

O jogo foi posteriormente avaliado por utilizadores, tendo como base um inquérito composto por um questionário pré-desenvolvido e avaliado cientificamente, o Game Experience Questionnaire. Os resultados obtidos foram bastante satisfátorios.

Acknowledgements

For all the support, reviewing and orientation I would like to thank to professor Luís Paulo Reis and professor Brígida Mónica.

To my father, my mother and my brother that raised me and orientated me to the right path, provide me with a good life quality and support, to my girlfriend Joana Alves that has constantly cheered-me up in the good and bad moments, and to all my friends that provide me good moments of fun and happiness, a deeply, deeply thank you.

To Sebastião Reis and Simão Reis, many thanks for the guiding and reviewing of this dissertation.

To Pedro Vendeira, Gabriel Silva, Yuan Zhi Heng and Behnam Aftab, a deep thank you for contributing with the model of the foot-guards, the goal posts and the basketball stadium model.

Last but not least, to all the staff from the Faculty of Engineering of University of Porto, professors and directors, thank you for this fabulous 5 years and for the opportunity of learning at one of the best university's of our country.

Francisco Silva

"Strength does not come from physical capacity.
It comes from an indomitable will."

Mahatma Gandhi

Contents

List of Figures

List of Tables

Abbreviations

PWC	Powerchair
PCF	Powerchair Football
FIPFA	Fédération Internationale de Powerchair football
RL	Reinforcement learning
IRL	Inverse Reinforcement learning
MDP	Markov Decision Process
EPW	Eletric Powered Wheelchair

Chapter 1

Introduction

1.1 Context

More than one billion people in the world live with some form of disability and nearly 200 million experience considerable difficulties in functioning[28].

Health, education, employment and transport are some of the basic services that many of us have long taken for granted, unfortunately, people with disabilities find this hard to achieve, therefore, many of them present higher rates of poverty than people without.

To achieve life quality for those, there is a constant need of removing the barriers which prevent them from participating in their communities, getting a quality education, finding decent work, and having their voices heard. In this context, wheelchairs have been developing a very important role in the life of these people. Research through time has been able to provide even better quality of wheelchairs, namely, powered and intelligent wheelchairs.

Even though these are remarkable accomplishments, powered wheelchairs and intelligent ones are often expensive and hard to maintain, furthermore many of them are no more than a prototype. As a way to deal with this problems and to provide better quality of life to these people, a project named Intellwheels 2.0, a derivation of the project Intellwheels[45], is being developed.

Intellwheels 2.0 focus in the develop of a framework/kit capable of being implemented in a commercial wheelchair, allow the use of an interface to control not only the smart wheelchair but also multiple other equipment and systems, create a complete semiautomatic system based on computational learning techniques, create a realistic 3D simulator and develop PC serious games. [8] [6] [17] [40] [51] [7] [18] [5] [46]

In this context, this thesis focus in the development of a Powerchair football serious game. The main objective is to develop a powerchair football video-game in 3D with some simulation properties. In addition, the game must allow players to play against intelligent agents. This agents will, therefore, learn to play the game by using machine learning algorithms.

It is expected to achieve an immersive, enjoyable, efficient and fluent video-game.

1.2 Motivation

The motivation for writing this dissertation comes mainly as an effort to help people with motor disabilities achieving better quality of life.

In order to people with disabilities have better life quality, there is a constant need of removing the barriers which prevent them from participating in their communities, getting a quality education, finding decent work, and having their voices heard. Therefore the development of this project may benefit people in this way.

People who learn to manoeuvre the powered wheelchair are taking a step closer to achieve autonomy since they will be able to move by them selves.

Furthermore, the video-game to be developed is referent to Powerchair Football, which is a sports modality adopted by real powered-wheelchair users.

These players often have little to no mobility and therefore need to be accompanied to the physical location of their training sessions. Furthermore, many of them may live far away from one another, which makes difficult for them to practice together consistently.

Therefore, the development of this dissertation may not only help these people to learn driving the wheelchair, but also help athletes of the sport to train without having the need to dislocate to the location of their training sessions or train alone due to the living distance of the team players.

1.3 Objectives

The main objective of this dissertation is to develop a serious game that has some simulation properties, it's immersive, enjoyable and efficient. In addiction, it would be great if, somehow, helped disabled people learn to drive better a powered wheelchair or teach them more about the sports modality.

That said, it is expected that a player will be able to perform a full match of Powerchair Football, taking in account the correspondent rules of the FIPFA Association. Furthermore, the video-game should be played in first person.

In addiction, and in order to make the game more entertain, the player must be able to play against intelligent agents. This agents will learn to play powerchair football by using machine learning algorithms. They must be capable of learn either by positive/negative reinforcement or by watching the player to play, thus, learn by imitation.

1.4 Document Structure

This initial chapter introduces the context of the dissertation, explaining how the world has evolved in way that wheelchairs are a very important contribution for life-quality of those who suffer from motor-disabilities, furthermore, it briefly explains how this thesis is related to the project intellwheels, describing the problem and providing the dissertation main objectives.

Chapter 2 is a derivation of the state of the art. Section 2.1 focus mainly on describing what are serious games, what related work has/is been/being done in wheelchair simulators. They are important for this dissertation since they develop an important role in the development of the game mainly because of its nature to simulate the conduction of the powered wheelchair in the video-game. It also explains the role of the enjoyment factor in serious games, which can be seen has a key factor to generate interest to the player.

Section 2.2 is a panoply of PCF (Powerchair football). This chapter introduces the PCF game as well as its definition and regulation. It covers topics such as equipment used in the modality, dimensions of the field, chair control and others that are vital for the development of the video-game logic.

Section 2.3 focus on Machine Learning techniques that are related to this work. It defines, succinctly, reinforcement learning, imitation learning and some relevant algorithms related to these. It also focus at environments that might be interesting for the development of this thesis.

Chapter 3 and 4 are mainly relative to the game development. Chapter 3 introduces and summarizes important notions of the Unity Engine, explains in detail how the game elements, such as the wheelchair, the ball and the field, were created and scripted for the game. Provides pseudo-code for the mechanisms of foul detection and ball out of bounds. Describes the controller succinctly and more. In a brief, explains in detail what was necessary to develop the game logic in its majority.

Chapter 4 explains how all the machine learning behaviours were developed. Provides information of the algorithms used to, the components for collecting environment observations, the actions used and the results obtained.

Chapter 5 discusses the evaluation of the dissertations. Discusses the approached solution based in the collected information from the inquiries.

Finally, Chapter 6 summarizes the whole dissertation and concludes it. It also references limitations and provides solutions for future work development.

Chapter 2

State of the Art

2.1 Serious Games

This chapter will provide information about serious games, namely its definition and related work in a way that explains how to approach the PCF as a serious game.

2.1.1 Definition

The "Game" word has been described in numerous forms for years, even today there is speculation in providing the most accurate definition of what Game is, thus, there is not a concrete definition of the word. However, there exists some key components that may seem generic to the majority of people on what a game must provide to a player, therefore, a game must provide some goals, rules, challenges, and interaction.

Games may be presented to potential players as a fun way to pass the time or interact with friends, thus, a form of entertainment. But what if the purpose of a game is not entirely related with this?

Serious game are precisely a type of game that might not be entirely fun neither a form of entertainment, therefore, serious game can be defined has games that do not have as primary purpose being fun, enjoyable or a form of entertainment[38]. In other hand, they must be games that should have an educational component or training.

It is important to understand that saying that a game must not have as primary purpose being fun it is not the same then saying that the game must not be fun. On the contrary, if serious games are made entertaining and fun, this could facilitate their adoption by children and adults alike[35][38], furthermore, a user may be more successful at learning some task if the task is presented in a fun way.

That said, in the context of this dissertation, the development of the PCF video-game can be seen has a key factor for teaching the user on how to manoeuvre the motorized wheelchair in a way that is fun. Furthermore, providing a multi-agent system and allowing multiplayer are both important factors to generate medium/long term interest on the game.

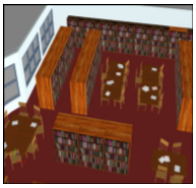
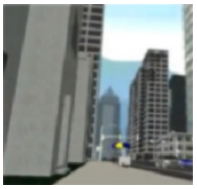
2.1.2 Wheelchair simulators

To learn and train the conduction of a powered wheelchair raises some questions.

What is the best way to approach the conduction in the game? Should it be a simulation? In order to understand this, this subsection will scope on related work of wheelchair simulators.

Simulation in wheelchairs has been studied since the early 80's. Pronk and his colleagues were the first to develop a simulation tool for electric wheelchair driving[48][3]. Researchers rapidly concluded that simulators could constitute a pertinent help for evaluation and/or adaptation of electric wheelchairs[3]. Since then, many different approaches have been presented in the development of simulators for powered wheelchairs. The objectives of these are mainly related with the improvement of driving powered wheelchairs[20].

Some pre-developed projects are presented in the following table.

	**Virtual Real. Wheelchair** In 2005, the Clarkson University presented a simulator that was also used by insure companies to give facilities for users to acquire wheelchairs.		**Uni. of Pittsburgh** In 2008 a study was published with tests performed by users with traumatic brain injury. A 2D virtual simulator was used to test the driving ability and performance of alternative controls.
	**ISIDORE** In 2008, Toulon Univ. presented a simulation project to have a better evaluation of the user and more efficient information to therapists and doctors that prescribe wheelchairs.		**University of Florida** In 2009, a wheelchair simulator was proposed in the University of Florida.
	**McGill simulator** In 2011, McGill simulator was proposed using the Unreal Development kit and a comparison in real and virtual environments were conducted using an electric wheelchair.		**WheelSim** The WheelSim © Life Tool is a commercial simulator. The wheelchair can be commanded with a joystick, the keyboard or with the rollerball.

Although many of these projects focus mostly in real electric wheelchairs, there are some others that concern in developing a simulator to perform tests[49]. One project that is very similar

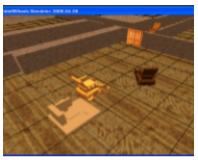
	**IntellWheels Simulator** In 2009 it was proposed the first simulator of an intelligent wheelchair with a multimodal interface.		**IntellSim** In 2012 Intellwheels project upgraded into a more realistic simulator based on USARSim and tested it with cerebral palsy patients.
	**ViEW** In 2015, ViEW was designed with several aims: to provide a setting for safe driving training, to test control skills, to aid in parameterizing wheelchair settings and to test new features..[41]		**SimCadRoM** In 2017, a virtual reality simulator for EPW safe driving learning purposes, testing of driving skills and performance, and testing of input interfaces was purposed [29].

Table 2.1: Wheelchair simulators (Some were adapted from [20])

to this in a way that is a simulation of the PCF game is the "VR Power Trainer", a project developed by R/GA Offices, that was developed with the objective to achieve the most real Virtual Reality training platform possible and adapt its hardware to the wheelchair so it can be used by the players in their homes[52].

Notice that the focus of this dissertation is not entirely related with the simulation itself because what is purposed to achieve is a PCF game that is fun to play, yet help the user to learn how to manoeuvre the wheelchair. Therefore, it can be seen as a game with some simulation properties, but not a full-fledged simulator.

Some important facts to retain regarding simulation for electric wheelchairs in a game context are the use of a first-person perspective, the scenario as an approximation to the real world and the physics. All of these are important factors for the development of the PCF game since these guarantee to cause immersion on the game and a way to provide a wheelchair conduction approximate to reality.

2.1.3 Enjoyment Factor

As previously discussed, serious games have been causing an extremely important role in learning, however, many of these are developed with a tight budget and suffer from poor game design and presentation[59]. Regardless of their successful educational component, many serious game have been reported for their low enjoyment.

A study made by Cuihua Shen, Hua Wang and Ute Ritterfeld[59], with the objective of identifying the enjoyment factor of some selected serious games, interviewed a player with 10 years of experience on playing digital games in all genres and relevant content areas and, using both quantitative and qualitative approaches, they evaluated the enjoy-ability of seven serious games by asking this player to play each of them and report his enjoyment experiences afterwards.

The experience concluded that, in a 100-point scale, where 0 means not at all enjoyable, 50 average and 100 very much enjoyable, five out of seven games ranged between 30 and 70 and could therefore be placed roughly around average.

Although this experience was positive, 2 out out of the 7 games were badly rated (both with a score of 20), furthermore they only analyzed 7 serious games that are by no means representative of all the serious games currently on the market.

Either way, it was sufficient to conclude some key point to make the game acceptable or playable, namely, the game has to meet certain thresholds in terms of technological capacity, aesthetic presentation, and game design elements. For example, the game must be stable, intuitive (player can move around easily and the objectives must be clear) and have good graphics quality as well as good sound design.

In a retrospective, serious games might be a boring experience for the player, therefore, many users might stop playing them in the early game which, generally, ends up by not providing the necessary educational factor. Therefore, using a full-pledged simulator to train the conduction of the powered wheelchair might not be enjoyable to the user. In order to counter this, the development of a PCF game with simulation properties might provide better results. A simulated conduction teaches the user on how to manoeuvre the wheelchair and a stable, intuitive and well designed PCF video-game provides enjoy-ability.

2.2 Powerchair Football

This chapter introduces the Powerchair Football game definition, classification and regulation. It will cover topic's such as equipment used in the modality, dimensions of the field, chair control, and some others which are vital for the development of the video-game logic and regulation.

2.2.1 Definition

Powerchair Football, also known as powersoccer is a modality sport where players with low mobility can play football using a PWC. This activity appeared in France in the late 70's [32], and it was originally proposed for disabled children so that they could learn how to manoeuvre the PWC in a more fun way.

In October 2005, as an initiative of France, representatives of the United States, Canada, Japan, England and Portugal, created officialy the modality of the Powerchair Football, therefore creating the Fédération Internationale de Powerchair football (FIPFA), the International Federation of Powerchair football [22].

Powerchair Football is an "adapted" modality of the football sport itself. There exists two teams of four players, each player uses a special foot-guards attached to a powered wheelchair and uses them to kick a large ball throughout the field. Players must be at least 5 years old to play and must have adequate control of their own PWC's.

The objective of the game is to manoeuvre the ball over the goal line of the opposite team while preventing the opposite team from doing the same [21].

While in most disability sports the chairs are controlled by the participants upper bodies, in PWCF, all the chairs are motorized due to the physical abilities of players. The majority of the player suffer from congenital disabilities such as muscular dystrophy, spinal muscular atrophy, arthrogryposis and cerebral palsy, but some of them have acquired disabilities such as multiple sclerosis and quadriplegia.

Since players are only allowed to use motorized wheelchairs, mainly controlled by a joystick and cannot exceed the speed limit of 10km/h, performance differences associated with gender or physical ability are greatly reduced or almost null.

2.2.2 Game elements

This subsection will not only talk about equipment used by the players but also about all the physical details that compose the game. This is an important subsection because here lies many of the information necessary to build the scenario of the game and approximate it to the reality, for instance, know the real dimensions of the field and the ball weight are crucial for the use of scales and delimitation of physics in the game.

All of the following subsections follow the laws of the game, approved at December 2010[21] by FIPFA.

2.2.2.1 The Equipment

Regarding the equipment of the players, each must be wearing a jersey or a shirt and a short or warm-up pants. Generally, if one is playing by a team, he must be wearing the team's shirt and corresponding short/pants as well as a clear and visible number, corresponding to the player's team number, however, the goalkeeper should be wearing colours that distinguish him from the other players.

In order to play the game it is absolutely necessary to use a powered wheelchair. Therefore, the wheelchair must fulfil some requirements, namely:

- The PWC must have four or more wheels (3 or 4-wheeled scooters or similar equipment is not allowed).
- The maximum permitted speed is of 10km/h either forwards or reverse.
- Backpacks, bags or other equipment are not allowed to be attached to the wheelchair unless they are essential to the player, some examples are oxygen mask, feeds, ventilator etc.
- Equipment that might interfere with other's PWCs are also not allowed.
- If necessary, additions should be placed on the PWCs that might trap, hold or ride over the ball in order to prevent this eventualities. ...

These are very important because they are responsible for the player to play/shoot the ball along the game. A typical foot-guard is shown in the following image.

Figure 2.1: Example of a foot-guards used by players

The foot-guards are attached to the front of the wheelchair and may vary from wheelchair to wheelchair, however they also must fulfil some requirements, namely:

- They must consist of a unbreakable material and be securely attached to the PWC.
- The use of it must not prevent the player of maintaining the eye contact with the ball.
- The surface must be solid and not angled in order to hit the ball upwards.
- The dimensions of the foot-guards must not exceed the one presented in figure 2.2.

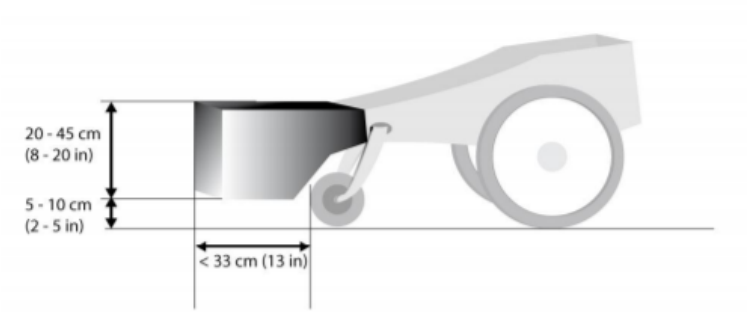


Figure 2.2: Foot-guard dimensions

Regarding the ball used for the match, it is very similar to a football ball thus a little bigger, it is spherical and must be fulfilled with the adequate pressure so as to minimise bouncing yet prevent PWCs from riding over it. It weights 1kg and has a diameter of 33cm.

2.2.2.2 The Field

The field of play where the game will be played has the size of a standard basketball field (28 m x 15 m), however, the dimensions may vary a little. The following table and image represents the specific dimensions and lines of the field[21].

	Maximum	Minimum
Length	30m	25m
Width	18m	14m

Table 2.2: Field dimensions [21]

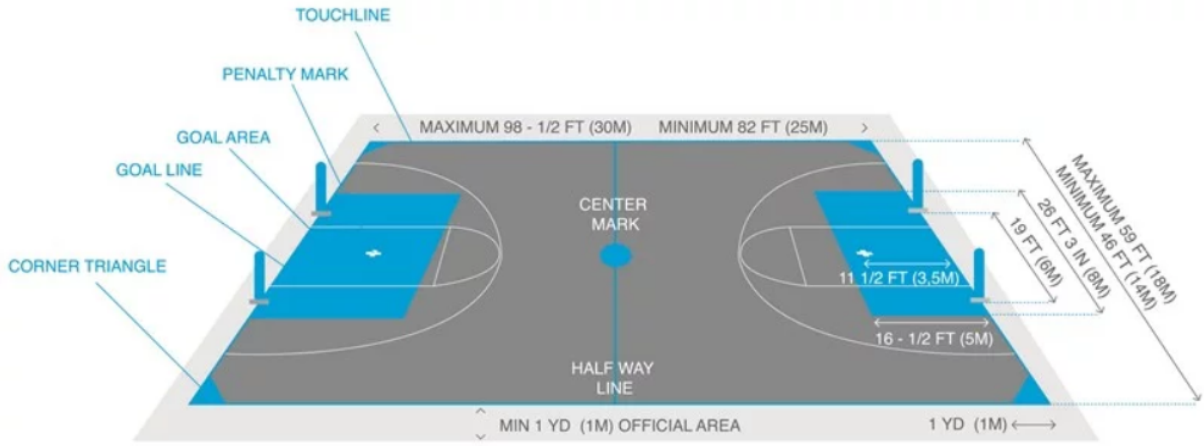


Figure 2.3: Powerchair football field dimension and regulation

The field is divided into two halves by a halfway line with a centre mark that indicates the midpoint of it. At each half, there exists a longer boundary line called touch line and a shorter one called goal line. The goal lines delimit the goal areas.

Goal areas are marked at the centre of each end of the field and are 8m wide and 5m deep. 3.5m away from the goal line and equidistant from each goalpost, there exists the penalty mark. The penalty mark is normally delimited with an "X" or a line taped onto the floor.

Two upright posts separated by 6m and placed on the extremities of the goal line delimit the goals. This posts might be pylons or cones.

Finally, an area of at least 1m wide exists around the entire perimeter of the field and allows the manoeuvre of the officials. Then, outside of the official areas lies the technical area. The technical area starts on the extremity of one of the goal lines and ends in the other, however, at the center of it, lies the scorer's table.

2.2.3 Regulation of the match

Powerchair football is very similar to football itself. The main objective is to score the maximum number of points while the game is running by passing the ball throughout the opposite team goal line, however, comparing with football, there are some similar and some different rules.

Since players are unable to kick the ball in the air, artificial space has to be created, therefore, two of the most important rules are 1) the "two-on-one" rule, and 2) the 3-in-the-goal-area violation.

1. "two-on-one": Only a player and an opponent are allowed to be less than 3 meters from each other. The non-compliance of this rule may lead to a fault and an indirect free kick to the opposing team.

2. "3-in-the-goal-area": The defending team is only allowed to have two players in their own goal area. Again, the non-compliance of this rule may lead to a fault.

An official match is composed of two equal periods of 20 minutes. Between these, there exists an interval of 10 minutes so that players may rest. Therefore, the game match starts after tossing a coin. This will decide which team starts attacking in the first half, and, as in football, the other team starts attacking on the second half. The kick-off is set on each team's side at the centre of the field.

If the ball goes outside of the field of play, the the ball is out of play. If the ball has been kicked from a player from team A, then the team B will kick the ball starting the place where the ball crossed the boundary line.

Sometimes, the game is unpredictable and some accidents may occur. When one happens, the game may stop leading to an allowance for time lost which is at the discretion of the referee. The referee as also the authority to take disciplinary sanctions such as using an yellow or red card. Depending on the sanction, the game may resume by placing the ball at the same place where the accident happened and kick the ball directly or indirectly. A penalty kick may also be attributed by the referee if necessary.

In case of a draw, competition rules may allow extra time of game or kicks from the penalty mark to determine the winner of a match.

2.3 Machine Learning

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. For example, these can be used for creating intelligent agents in video-games that can learn to play the game by themselves therefore, this subsection is an important piece in the state of the art of this dissertation.

In a summarized way, Machine Learning can be broadly defined as computational methods using experience to improve performance or to make accurate predictions.[39]

The following sections focus mainly in more concrete machine learning sub-topics, namely RL and IML. These are presented in a way to give a general context to the reader of what they are. Furthermore, some frameworks/toolkits regarding training environments will be discussed in order to understand the importance of these in this dissertation.

2.3.1 Reinforcement Learning

Reinforcement Learning can be seen as an area of machine learning which its major objective is learning what to do, how to map situations to actions, as well as to maximize a numerical reward signal. There exists a learner, which can be an agent, and he is not told which actions to take. He must discover which actions provide the most reward by trying them.

RL cannot be seen as supervised learning, which is a type of system in which both input and output data are provided and it is expected that a given algorithm learns with it[61]. In a similar way, RL cannot be seen as unsupervised learning, which is typically about finding structure hidden in collections of unlabeled data. Instead, RL should find a set of behaviours that optimize its reward function.

Although there exists many different approach techniques for RL, a simple RL process can be expressed as a Markov Decision Process [42].

In a very succinctly way, a MDP is a discrete time stochastic control process. It models decision making in situations where outcomes may be random and partly under control of a decision maker[50], therefore, it can make decisions in stochastic environments.

Like a Markov chain[60], the model attempts to predict an outcome given only information provided by the current state. However, the Markov decision process incorporates the characteristics of actions and motivations. At each step during the process, the decision maker may choose to take an action available in the current state, resulting in the model moving to the next step and offering the decision maker a reward[15].

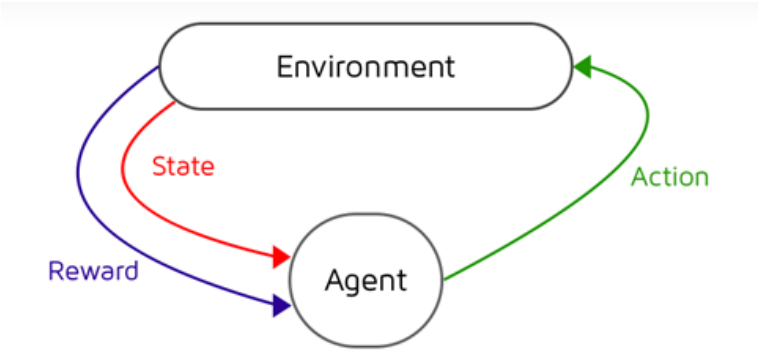


Figure 2.4: Markov Decision Process [15]

One advantage, or key, of RL techniques is that an agent learns a good behavior or, in other words, an agent is able to modify or acquire new behaviors and skills incrementally [24]. Another important aspect is that RL uses trial-and-error experience, therefore, it is not essential to have complete knowledge or control of the environment, only needs to be able to interact with it and collect information [24].

In a retrospective, RL seems to be a good candidate for the development of the agents in the game, since the environment of the game seems to be stochastic.

2.3.2 Imitation Learning

The demand of intelligent agents being capable of mimicking human behaviour has grown over the years. Advancements generally in robotics and communication have risen to potential applications that require artificial agents that can either make intelligent decisions and perform realistic motor-actions, decisions that require ability to behave like a human.

In Imitation Learning, a Teacher or Expert can be seen as an experienced agent or a human that knows how to perform successfully a required action. The same way, there exists a learner which is an agent that tries to learn with the teacher. The idea is that the learning agent observes the actions of an experienced agent as well as the corresponding consequences in a way that the observations give the learner information or clues on how to successfully perform the required task.

Generally, IML is useful when it is easier for an expert to demonstrate the desired behaviour rather than to specify a reward function which would generate the same behaviour or to directly learn the policy.

Some examples of IML applied are navigational problems, which typically employ vehicle-like robots, with relatively lower degrees of freedom. These include flying vehicles [13] or ground vehicles [12]. Other applications focus on robots with higher degrees of freedom such as humanoid robots [10]. Although the majority of applications are applied in robotics, IML has been applied to simulations and even computer games [63][53]

The next subsection introduces some well-know IML techniques.

2.3.2.1 Behavioral Cloning

Behavioral cloning is a technique for creating agent behaviors. This technique focus in replicating patterns of behavior observed in humans and/or other agents. Instead of trying to explain to the agent what the teacher knows, teachers simply perform tasks. These tasks are then somehow recorded and machine learning algorithms are used to create a model which is used to produce agent behaviors[2].

In behavioral cloning, we aim simply to learn the policy via supervised learning[31]. Specifically, we will fix a policy class and aim to learn a policy mapping states to actions. One notable example of this is ALVINN[47], which learned to map from sensor inputs to steering angles.

In a more specific way, behaviour cloning assumes that there exists a subject of some kind that is under control of a human operator or/and a super intelligent agent. The subject may be a physical system or a simulation, still, for pragmatic reasons, behavioral cloning has mostly been implemented and tested in simulation environments. Either way, the subject must be instrumented so that it is possible to capture the state of the system, including all the control settings. Thus, whenever the operator performs an action, we can associate that action to a particular state.

One brief example, suppose that one wants to create an agent to control a pole and cart system by applying a left push or a right push in the appropriate time. Whenever a control actions of left/right is performed, one should record the action as well as the values of the four state variables

at that time. In this proper example there exists four states, x (position of cart), v (velocity of cart) t (angle of pole) and w (angular velocity of pole). Therefore, each of these records can be viewed as an example of a mapping from state to action. This kind of approach can be seen as direct Learning or direct controllers[55] due to its nature of mapping situations to actions.

Although direct controllers work quite well for systems that have a relatively small state space, when one is presented with a more complex system, action rules tend to produce very brittle controllers. To face this problem, successful methods decompose the learning task into two stages: learning goals and learning the actions to achieve those goals, therefore, this can be seen as indirect learning or indirect controllers.

A controller is 'indirect' if it does not compute the next action directly from the system's current state but, in addition, uses some intermediate information[55]. In other words, this can be seen as the inverse of the usual problem of controller design, that is, given the actions in an operator's trace, find the goal that these actions achieve. Therefore, the next subsection explains IRL in a way to deal with this problem.

2.3.2.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning, also known as inverse optimal control [34], inverse planning [4] or structural estimation of MDPs Rust [54] is a technique that focus in the learner trying to recover a reward function from a policy (or demonstrations of a policy).

The original IRL algorithms were introduced by Ng and Russell (2000) and Abbeel and Ng (2004). These algorithms basically formulate the IRL problem as a linear programming procedure with constraints corresponding to the optimal condition.

There are mainly three cases existed in the IRL problem formulation[25]:

- Finite-state MDP with known optimal policy.
- Infinite-state MDP with known optimal policy.
- Infinite-state MDP with unknown optimal policy, but demonstrations are given. Being this one the closest to practical problems because, usually, only the expert's demonstrations are available rather than the explicit policy.

The main idea of IRL is to learn the reward function of the environment based on the expert's demonstrations and then find the optimal policy by using RL.

A generic and simple way of methodologically structuring IRL is the following[14]:

1. Start with a set of expert's demonstrations (assuming these are optimal) and then try to estimate the reward function, that would cause the expert's behavior/policy.
2. Update the reward function parameters.

3. Solve the reinforced learning problem, that is, given the reward function, find the optimal policy.

4. Compare the newly learned policy with the expert's policy.

5. Repeat step 2, 3 and 4 until converge and find a good enough policy.

Depending on the actual problem, there can be two main approaches of Inverse Reinforcement: the model-given and the model-free approach. Both of these are described at 2.3.3.1.

In summary, IRL can be considered as a reverse procedure of RL problems. The assumption is that the expert's demonstration is an optimal policy p*, which is derived according to some reward function R*. The objective of IRL is to learn this unknown R*[25].

2.3.3 Related Algorithms

This subsection aims to explain some relevant algorithms related to RL and IML in order to understand if they are good candidates or not for the development of the agents in this dissertation.

2.3.3.1 Q-learning

There are two main learning approaches that can be adopted, the model-free approach, which consists of learning an action policy directly and the model-based, which consists of first learning the environment model and then use that to learn a policy.

The Q-learning algorithm belongs to the family of model-free approaches and was firstly introduced by Chris Watkins[69] in 1989. A convergence proof was also presented by Watkins and Dayan in 1992[68]

A simple description of the Q-learning algorithm is as follows:

1. Agent senses its environment, using this information to determine its current state.

2. Agent takes an action and obtain a penalty or reward.

3. Agent senses its environment again - to see what effect its chosen action had.

4. Agent learns from its experience (and so makes 'better' decisions next time)

A brief description in pseudo-code of how the algorithm works in run time is presented in the following fig. 2.5:

A major limitation of Q-learning is that it only works in environments with discrete finite states and action spaces, furthermore, occasions where the Q function (i.e. reward function) is too complex to be learned, Q-learning tend to fail. Therefore, Q-learning might not be suitable for the development of the agents in this dissertation.

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Figure 2.5: Q learning algorithm pseudo-code [15]

2.3.3.2 Trust Region Policy Optimization

Trust region methods are a class of methods used in general optimization problems to constrain the update size. They work in a way that first define a region around the current best solution, in which a certain model can do some extent approximation to the original objective function.

That said, Trust Region Policy Optimization (TRPO) is a scalable on-policy region method algorithm for optimizing policies in reinforcement learning by gradient descent [56][44].

TRPO can be used for environments with either discrete or continuous action spaces, therefore, do not require access to a model of the environment and often enjoy better practical stability.

TRPO updates policies by taking the largest step possible to improve performance, while satisfying a special constraint on how close the new and old policies are allowed to be.

2.3.3.3 Proximal Policy Optimization

Proximal Policy Optimization (PPO) can be seen as a motivation of the TRPO algorithm that emphasizes: how can one take the biggest advantage step on a policy using the data we currently have, without stepping so far that we accidentally cause performance to collapse?

TRPO tries to solve this problem with a complex second-order method while PPO, being a family of first-order methods, uses a few other tricks to keep new policies close to old.

In a few words, PPO is an on-policy algorithm that can be used for environments with either discrete or continuous action spaces.

The main objective is to perform not just one but multiple minibatch gradient steps with the experience of each iteration. Therefore, reusing data to make more progress per iteration, while stability is ensured by limiting the divergence between the old and updated policies. There exists two primary variants of PPO algorithm, PPO-Penalty and PPO-Clip. [57][44]

PPO-Penalty approximately solves a KL-constrained update like TRPO, but penalizes the KL-divergence in the objective function instead of making it a hard constraint, and automatically adjusts the penalty coefficient over the course of training so that it's scaled appropriately.

PPO-Clip doesn't have a KL-divergence term in the objective and doesn't have a constraint at all. Instead relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy.

Due to the efficient and reliable results that these algorithm has been providing, PPO seems to be a good candidate for the development of the agents in this dissertation.

2.3.4 Training Environments

2.3.4.1 TensorFlow

Figure 2.6: Tensorflow Logotype

Developed by Google Brain team mainly with the purpose of being used internally by the company, Tensorflow was realeased as an open-project at November 9 2015.

TensorFlow is an open source machine learning library applicable to a wide variety of tasks. It is a toolkit with the main purpose of creating and training neural networks to detect and decipher patterns and correlations.[27]

It runs on nearly everything: GPUs and CPUs—including mobile and embedded platforms—and even tensor processing units (TPUs), which are specialized hardware to do tensor math on[67].

2.3.4.2 OpenAI Gym

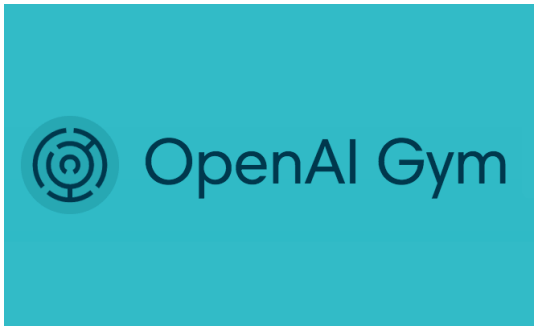


Figure 2.7: OpenAI Gym Logotype

OpenAI is a non-profit research institution in artificial intelligence, which aims to promote and develop friendly AI with the main objective to benefit humanity as a whole. The organization seeks to "collaborate freely" with other institutions [26] and with researchers making their patents and research open to the public.[36]

As a result of this project, OpenAI Gym is a toolkit/library for developing and comparing RL algorithms. It makes no assumptions about the structure of the using agents, and is compatible with any numerical computation library, such as TensorFlow, described at 2.3.4.1 or Theano.[1]

More specifically, the gym library is a collection of test problems — environments — that one can use to train RL algorithms, furthermore, these environments have a shared interface, allowing users to write general algorithms.

2.3.4.3 Unity ML-Agents

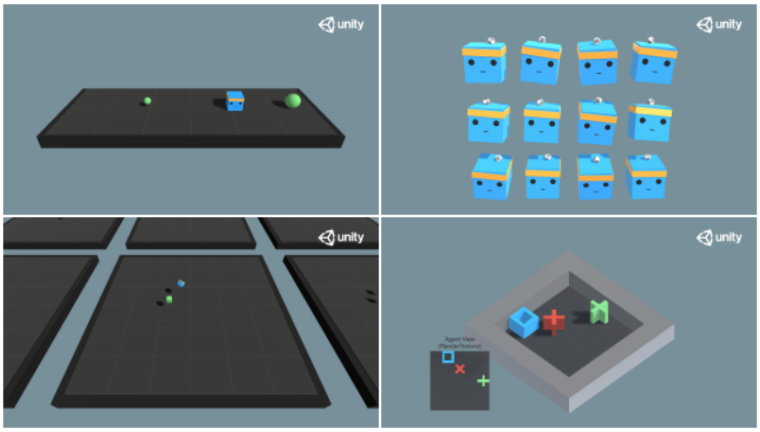


Figure 2.8: Unity ML-Agents Framework

Unity ML-Agents stand for Unity Machine Learning Agents Toolkit and is an open-source Unity plugin that enables games and simulations to serve as environments for training intelligent agents.

By using Python API, this plugin allows users to train agents using RL, IML, neuro-evolution and other machine learning methods. It also provide implementations (based on TensorFlow2.3.4.1) of state-of-the-art algorithms to enable game developers and hobbyists to easily train intelligent agents for 2D, 3D and VR/AR games.

These trained agents can be used for multiple purposes, including controlling NPC behaviors, automated testing of game builds and evaluating different pre-released game design decisions. In addiction, the ML-Agents toolkit is mutually beneficial for both game developers and AI researchers as it provides a central platform where advances in AI can be evaluated on Unity's rich environments and then made accessible to the wider research and game developer communities. [33]

2.4 Summary

With the closing of this chapter, we may conclude that the development of a PCF video-game in a context of a serious game must fulfil some requirements in a manner that it can be an enjoyable video-game and, if possible, teach user to drive the powered wheelchair.

We have seen that the simulation is a powerful manner of educating the user on how to control/manoeuvre the powered wheelchair, therefore, the development of a PCF video-game in the context of a serious game must, at least, simulate the conduction of the powered wheelchair. A first-person perspective must be implemented for the same reason of realism, as well as, the approximate, real physics of the powered wheelchair and the world.

Furthermore, the enjoyment factor is a very important key to achieve this goals.

Chapter 3

Intell Power Soccer

The following sections aim to explain, in detail, all the architecture and steps done to achieve this video-game. This sections are composed by four main groups: "Game Elements", responsible to explain how the environment of the game was developed, mainly game physics, 3D models and colliders, "Game Logic", which is the core of the video-game and all the rules applied to the game and his mechanism, "Agents", which explains how all the Machine Learning behaviours were developed and assembled and, finally, "User-interface" which explains how all the game menus such as the pause menus and other were created.

3.1 Game Elements

3.1.1 Game Engine

As video-games industry increases, the ease of developing video-games increases as well. Nowadays, there exists many software to help develop video-games, furthermore, there exists a gigantic community in internet willing to help people who wants to learn.

One of the most used frameworks/video-games engine is Unity 3D. It packs a ton of features together and is flexible enough to make almost any game, either 3D or 2D games, furthermore, it has a open and friendly community which is great for people who are starting developing video-games. Therefore, Unity is the one chosen to be the engine of this video-game.

In order to create this game it is necessary to answer one main question, should the game be in 3D or 2D? As said before at 1.3, because it is expected to achieve realism, 3D is more proximate to the reality, therefore, the game will be developed in 3D.

3.1.2 Wheelchair

Has mentioned before at 2.2, a Powerchair football game is composed by two teams of four player, each player rides a powered wheelchair and tries to score a goal. By the end of 40 minutes, the

team who had scored more goals wins the match.

Because 3D games requires 3D Models, to develop this video-game it is strictly necessary to have a Powerchair-football wheelchair 3D Model, therefore, the following model in fig. 3.1 was chosen to be the representative model of a real powered wheelchair.

Figure 3.1: Real Scale 3D Model chosen to represent the Powerchair-football wheelchair

This model is made at real scale, which was found necessary in order to achieve a realistic behaviour when controlling the wheelchair. We will discuss the need of a real-scale model further.

One of the main problems found in simulating the wheelchair behaviour is how to correctly allow the caster wheels to move freely as they should when they contact with the floor. In order to understand this, it is necessary to know how a powered wheelchair works.

A powered wheelchair usually have 4 wheels. Two back wheels connected directly to a motor and two caster wheels respectively connected to a caster, a junction that connects the wheel to the body of the wheelchair and is responsible for the wheel to spin over itself. Figure 3.5 illustrates how a caster wheel should be working when facing a frontward direction.

When the motor of a wheelchair starts, the user can control the wheelchair by using a joystick. If he pushes left, the right back wheel starts spinning frontward, and the left back wheel starts spinning backwards. This behaviour results in the wheelchair rotating over himself to the left.

While the wheelchair moves to any pointed direction made by the user, the caster wheels freely rotate about 360^{o}, thus enabling the wheel to roll in any direction. This makes it possible to easily move the vehicle in any direction without changing its orientation.

3.1.2.1 Hinge Joints

To start simulating the wheelchair behaviour, it was necessary to cut each of the parts of the model, specifically, separate the wheels from the casters and the casters from the body in a way that Unity

recognizes a wheel or a caster as separated objects from the whole wheelchair model.

This was obligatory because the strategy adopted to make the wheels spin, as well as the casters, was to connect each of the parts with an Hinge-Joint Component.

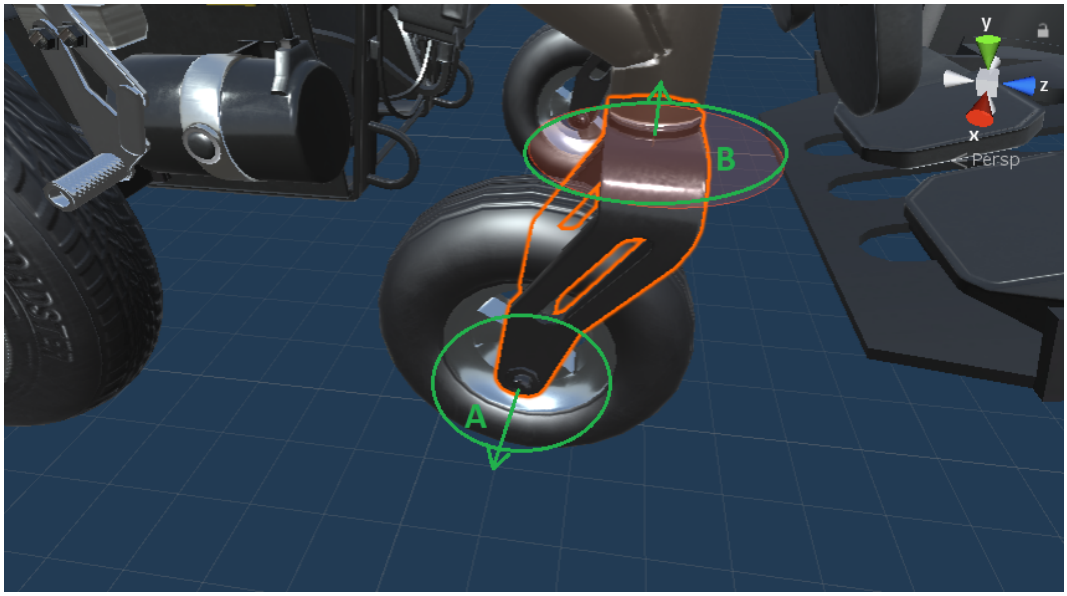


Figure 3.2: Front caster wheels Hinge-Joint assembly

Hinge Joints are components that groups together two Objects, constraining them to move like if they were connected by a hinge. It is perfect for doors, but can also be used to model chains, pendulums, etc. [66] Fig. 3.2 represents how the Hinge Joints are connected and actuate. The figure highlighted by ***Orange*** is the Caster, responsible for the wheel to spin over itself in the Y Axis. ***A*** is the Hinge-Joint connection that allows the wheel to spin over herself around the X Axis. The same applies for ***B***, which allows the caster to spin freely around the Y Axis.

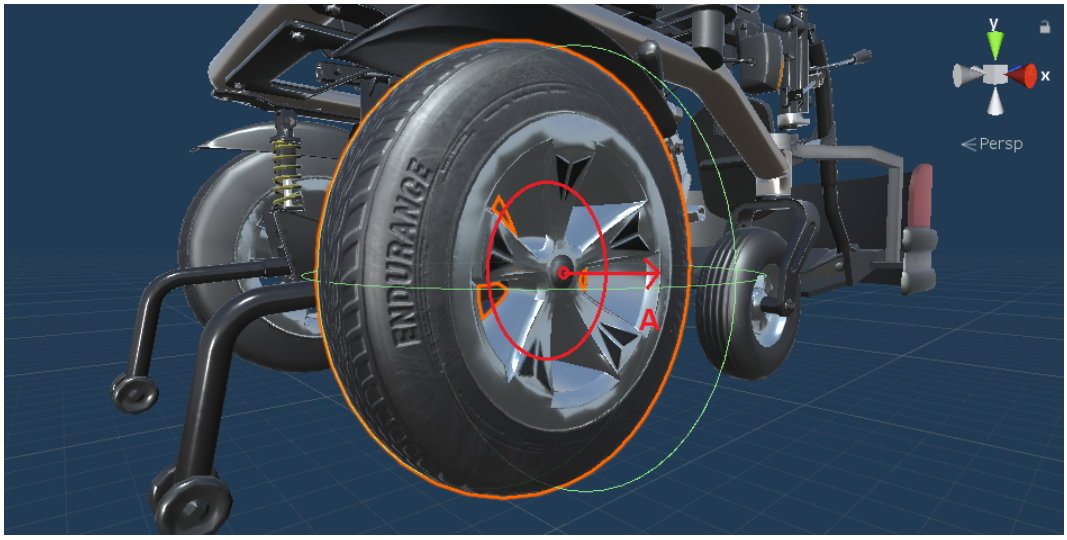


Figure 3.3: Middle big wheels Hinge-Joint assembly

Figure 3.4: Support back wheels assembly

The same strategy was applied to the two back big wheels and the other two support back wheels, as represented in fig.3.3 and fig.3.4, respectively. In each of the figures its possible to observe a sphere in green as well. These spheres are named Colliders and they are attached to each of the wheels. We will discuss colliders in the next section.

Hinge Joints are composed by some parameters. These parameters are values that can be changed in order to achieve more complex behaviours. The following table 3.1 specifies the ones that were changed in order to achieve a more realistic behaviour of the chair being controlled.

Connected Body	Object that the joint is dependent upon.
Spring	The force the object asserts to move into the position.
Damper	The higher this value, the more the object will slow down.
Break Force	The force that needs to be applied for this joint to break.
Mass Scale	The scale to apply to the inverse mass and inertia tensor of the body.
Connected Mass Scale	The scale to apply to the inverse mass and inertia tensor of the connected body.

Table 3.1: Hinge Joint parameters [66]

Although these parameters represent specific things, because unity is not a perfect world of physics, all this values should be changed by trial and error.

This was precisely one of the more time consumption part of the work because it is not an easy task to find the correct values to allow the wheels of the casters to work as expected, furthermore, if the model being used is not correct scaled and with the appropriate masses, find these values may be very hard or even impossible. This is the main reason why a real scaled model was chosen to represent the wheelchair.

In the following figure 3.5 it is possible to observe how a caster wheel should rotate correctly when facing a forward force and how it should not, which happened many times when changing

the values and the masses.

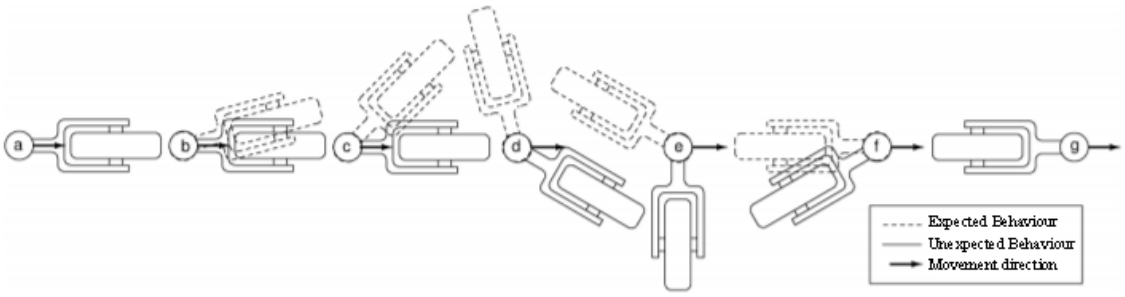


Figure 3.5: Wheelchair Caster expected behaviour vs unexpected behaviour (Adapted from [11])

3.1.2.2 Colliders

Collider components define the shape of a Game Object for the proposes of physical collisions and when they interact with each other, their surfaces simulate the properties of the material they were given. When colliders interact, their surfaces need to simulate the properties of the material they are supposed to represent. For example, a material created to represent ice will be slippery with almost no friction, on the other hand, a football ball will offer a lot of friction and will be very bouncy. Although the shape of colliders is not deformed during collisions, their friction and bounce can be configured using Physics Materials.[65]

In a brief, colliders represent a component responsible for detecting collisions. For instance, if two of these objects collide with each other, a collision event will be triggered and none of these objects will pass through each other, unless desired. In this situation, these colliders will be responsible for the rotation of the wheels when they are in contact with the floor of the field. All the six wheels of the whellchair have one of this attached to itself and their corresponding Physics material is the default rubber material pre-defined by Unity. The following figures represent the remaining colliders and their use.

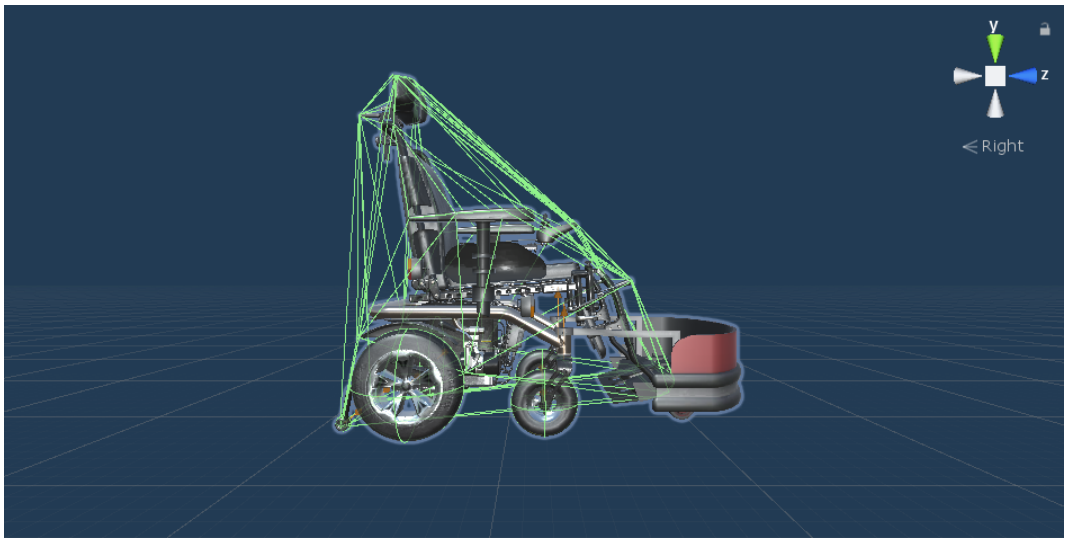


Figure 3.6: Wheelchair body collider

Fig.3.6 is the body collider. This collider is mainly used to detect collisions between different players and agents, it avoids them to pass through each other and in case the ball collides with it, the ball is projected to its equal force and opposite direction.

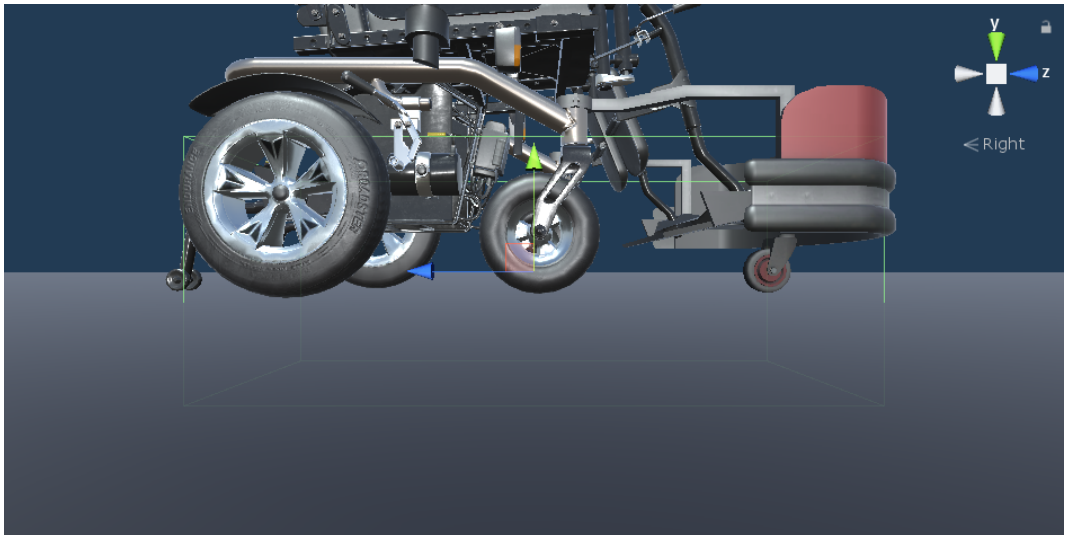


Figure 3.7: Wheelchair tracking collider

Fig.3.7 is a special collider, it is not colliding with anything nor have a physics material attached. As we will see in 3.1.3.2, this collider is not running by its default behaviour, it is somehow disabled. This is being used to detect the position of the agents or players in the field, this way, the main game logic script keeps the track of the players.

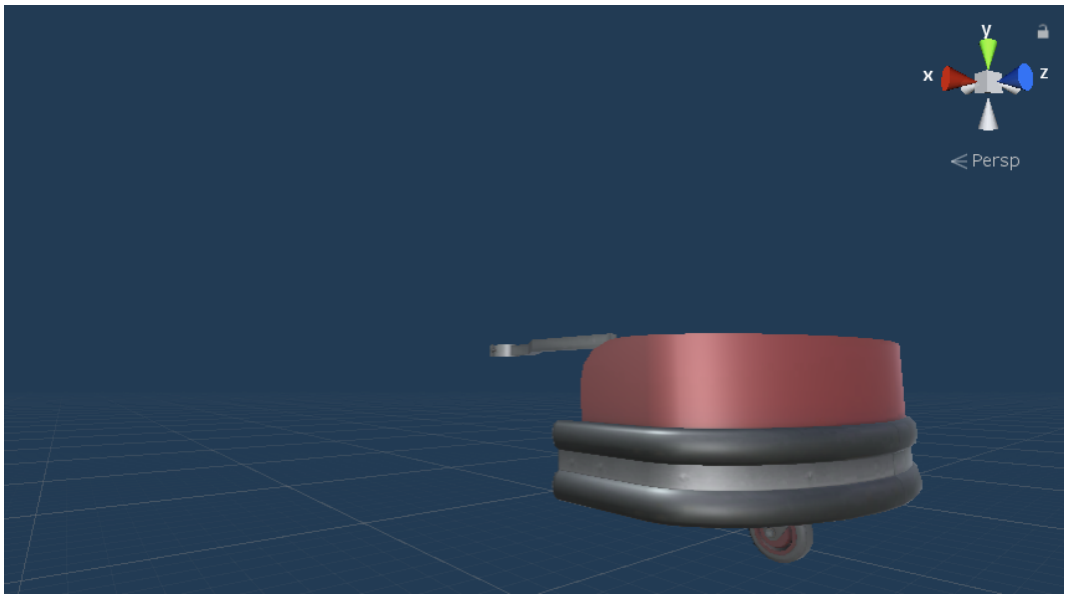


Figure 3.8: Wheelchair feet model

Fig.3.8 is the model of the Feet or Foot-guards. As previously referenced at 2.2, the wheelchairs used in Powerchair football need to have a Feet which comes in need to players kick the ball. These feet model have two versions, a red one, which is represented in the figure, and a blue one. These versions are used to distinguish players between teams. A red feet is a player who plays in the red team and a blue one is a player that plays in the blue team.

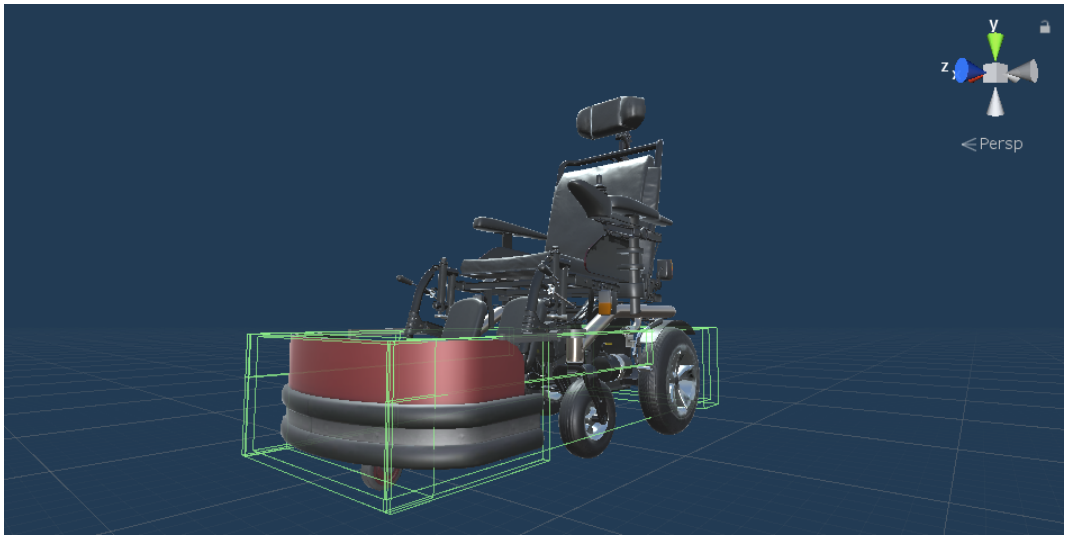


Figure 3.9: Wheelchair feet colliders

As usual, every feet in a wheelchair has its own collider, fig. 3.9 represents this. The Physics material attached to this one is the default metal material pre-defined by Unity.

3.1.3 Field

The field is an important piece, it is not only necessary because users need to know where they physically are but it highly contributes to the game logic. This next section aims to explain how the field of play is assembled and how it is functioning along with the game logic.

3.1.3.1 Areas

The field of play is composed by 5 major areas, the red area, the blue area, the small red area, the small blue area and the out-of-bounds area.

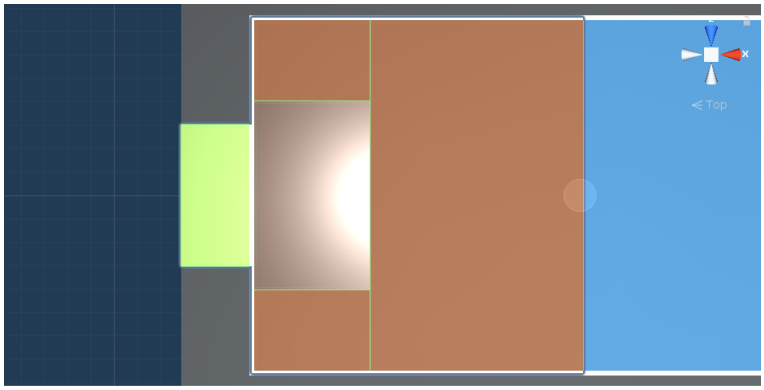


Figure 3.10: Red Team Area

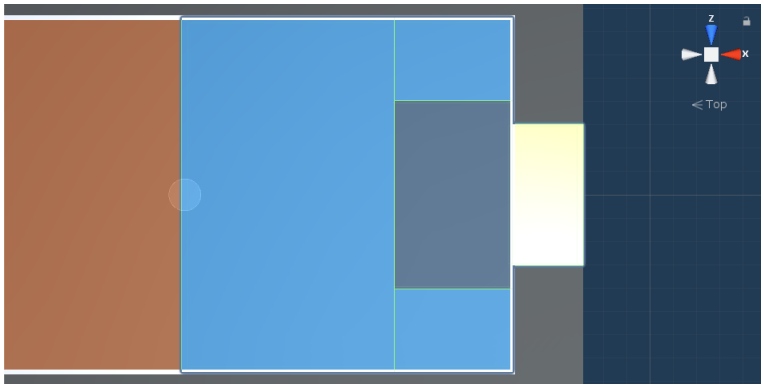


Figure 3.11: Blue Team Area

Figures 3.10 and 3.11 are the red and blue team areas, respectively. They are composed by each team own side of the field and correspondent small areas. This is easily observable at fig. 3.11: the lighter blue color stands for the Blue Team big area and the stronger blue color is the Blue Team small area. The same ideology applies to fig. 3.11. In this figure it is also observable at green color the goal area. These areas are used to keep track of the ball, the agents and to understand if a goal was scored.

Figure 3.12: Field Out-of-Bounds area

The green outlined lines at Fig. 3.12 represent the whole area where the ball should not be played in. When there exist a collision with this and the ball, the game enters in a ball-out-of-bounds period.

It is noteworthy that this observable model and the colors at correspondent areas in the figures are not the final model of the field neither the model chosen to represent the final field of play, they were created only with the objective of development. This field will serve as a mask for the final project.

3.1.3.2 Tracking Mechanism

Each of the areas observed before at 3.1.3.1 have colliders attached to it, and, in junction with the previously referenced, Wheelchair tracking collider 3.7, they make a simple system that keeps the track of the ball and the players in field. What happens is the following.

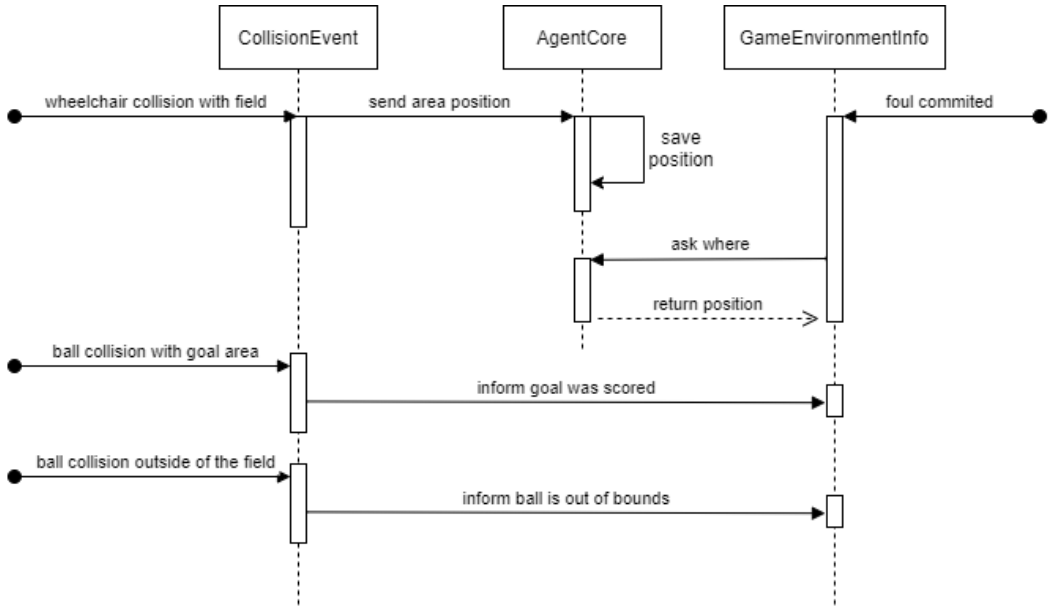


Figure 3.13: Field Event Collision Sequence Diagram

Every time a player enters on one of these areas a ***CollisionEvent*** is triggered. This happens because the Wheelchair tracking collider 3.7 collides with the floor of the field. Then, this event registers the information that a given player X is at the Y area and sends its area position to the ***AgentCore*** class, where it is saved. This becomes useful because in the ***GameEnvironmentInfo*** Class, explained further in this dissertation, there exists a List of all the players that are constantly being accessed in order to keep the game running.

For instance, this is useful in a situation where a foul is committed. Since the difference between a penalty or a regular foul is the place in field where the foul happened, the tracking system comes in hand. Furthermore, this mechanism checks if the ball is out of bounds or if a goal is scored by triggering the ***CollisionEvent*** when the ball is colliding with the outside field or the goal area respectively. If any of this events happen, ***CollisionEvent*** informs ***GameEnvironmentInfo*** that the ball is out of bounds. Figure 3.13 illustrates this mechanism.

3.1.4 Ball

The ball as a game element is a simple sphere modeled by unity. Its has a RigidBody Component attached and a Sphere Collider. The Sphere Collider has a Rubber Physics Material with values that were edited to look like a real football ball. Once more, find the correct values for the physics material took some time because of the trial-and-error approach that is necessary.

3.2 Game Logic

The next section aims to explain in detail everything that was necessary to create the mechanism of the game, or, in other words, the game logic. Game Logic refers to the internal mechanism of a

game in order to perform all the tasks needed for it to work as desired. It might be seen as all the programming logic applied to games such as collision detection, AI, animation, and other.

3.2.1 Controller

There exists a wide variety and types of wheelchair. They may be regular wheelchairs composed by 4 wheels and no motor. They may be specific to particular activities, as seen with sports wheelchairs and beach wheelchairs. They might have 6 wheels or even be adapted to the type of disability. Either way, the most widely recognized distinction is between powered wheelchairs, where propulsion is provided by batteries and electric motors, and manually propelled wheelchairs, where the force that allows the chair to move is provided either by the user pushing the wheelchair with his own hands or an attendant pushing from the rear.

This video-game focus in powerchair football modality, therefore, wheelchairs used in this type of game, as referenced at 2.2.3, are obligatory to be powered wheelchairs and, although many individuals might not have the motor skills to effectively operate it, a powered wheelchair is usually controlled by a joystick. Since this video-game tries to be realistic operating the wheelchair, it was decided that a joystick should be the primary controller while playing the game, being the "AWSD" keys of the keyboard, the secondary.

When a wheelchair is motorized, the back wheels are usually directly attached to the electric motor and this is manipulated by the joystick. A generic explanation of what happens is the following: When a user inputs the joystick to the left, the motor commands the right wheel to spin forwards and the left wheel to spin backward. This comportment results in the wheelchair spinning over himself to the left. The same happens on the contrary when the user pushes right. If the users pushes front, both back wheels spin frontwards and the wheelchair moves frontwards. If the user pushes back, both back wheels spin backwards and the wheelchair moves backwards. In order this to happen, it is necessary to map the joystick to the desired behaviour of the motor.

[19] explains three different ways of achieving this. We will focus on two of them.

The first is a relatively simple mapping where the steering and power mapping is achieved through the use of both x and y in all instances and the wheelchair can rotate around itself when the joystick is pushed to the sides.

Assuming that the joystick is represented in a Cartesian coordinate system, x and y are the correspondent axis and vary between -1 and 1 and this coordinates can be used to determine the distance to the center of the joystick or resting position, which is point (0,0), and L is the desired speed of the left wheel and R the desired speed of the right wheel represented by normalized values (between -1 and 1), this mapping can be calculated as follows:

$$\begin{cases} R = y - x \\ L = y + x \end{cases}$$

In the second method, the distance of the handle to the center of the joystick (ρ) is proportional to the maximum wheel speed and the angle (θ) of the joystick relating to the vector (0, 1). This

way, the y axis, determines how the speed is distributed to the wheels and in order to keep ρ minor or equal to 1, the value is clipped when ρ is above one. Assuming that x and y are normalized and vary between -1 and 1, the control follows these conditions:

$$R = \begin{cases} \rho, \text{ if } 0 \leq \theta \leq \frac{\pi}{2} \\ \rho \cdot \frac{-\theta + 3.\pi/4}{\pi/4}, \text{ if } \frac{\pi}{2} \leq \theta \leq \pi \\ -\rho, \text{ if } -\pi < \theta < -\frac{\pi}{2} \\ \rho \cdot \frac{\theta + \pi/4}{\pi/4}, \text{ if } -\frac{\pi}{2} \leq \theta \leq 0 \end{cases}$$

$$L = \begin{cases} \rho \cdot \frac{-\theta + \pi/4}{\pi/4}, \text{ if } 0 \leq \theta \leq \frac{\pi}{2} \\ -\rho, \text{ if } \frac{\pi}{2} \leq \theta \leq \pi \\ \rho \cdot \frac{\theta + 3 \cdot \pi/4}{\pi/4}, \text{ if } -\pi < \theta < -\frac{\pi}{2} \\ \rho, \text{ if } -\frac{\pi}{2} \leq \theta \leq 0 \end{cases}$$

$$\text{where } \rho = \sqrt{x^2 + y^2} \text{ and } \theta = \operatorname{atan} 2(-x, y)$$

Although the first method seems to be good, it has quite a big problem for the user: Steering the wheelchair requires accurate precision in the joystick. For instance, the wheel speed variations that derive from changes in joystick direction changes are quite steep, and when moving forward, little variations on the joystick horizontal axis, result in a big adjustment of the direction. Furthermore, a great space area available in the joystick becomes useless.

On other hand, the second method becomes more useful and smooth. The speed variation is extended towards the corners of the joystick, and clipping is minimized. However, this method has a problem too. Suppose a user is directing the joystick to vector (0,1), the wheelchair will be now spinning at full speed to the right but at the small change of the vector, and because humans may be quite inaccurate, a small tremble may lead to a vector like (-0.01,1). What will happen in this situation is that, now, the wheelchair is spinning and going a little backwards. This is a situation unpleasant because the user of the wheelchair did not want this to happen, it just did because users may be inaccurate due to many external factor. This problem will be called the User unexpected joystick handling (UUJH).

That said, what it is proposed in this dissertation to deal with the controller mapping problem is to use the second method proposed by [19] with the addiction of creating gaps in the extremities of the Cartesian coordinate system, more in concrete, in the (-1,0) and (0,1) axis, to attenuate the UUJH problem.

3.2.1.1 Controller Class

In order to achieve the desired objective of providing an enjoyable controller to the user and avoid the UUJH problems, the Controller class was created.

This class is full responsible for creating a operational controller for the video-game taking in count the [19] second method, reference previously at 3.2.1.

The class receives both of the big back wheel Objects, which are the wheels that are directly attached to the motor of the wheelchair, that are further used to provide the correct angular velocity, thus, allowing the wheelchair to move. It also receives the user input of the joystick, as a Vector of two dimensions, and the maximum velocity that the wheelchair can reach. After this, the distance of the handle to the center of the joystick (ρ) is calculated as well as (θ), the angle of the joystick related to the vector (0, 1).

Finally, the second method algorithm is done with the small exception of the gaps. The following expression shows them:

$$R = \begin{cases} \rho, \text{ if } 0 \leq \theta \leq \frac{\pi}{2} \\ \rho \cdot \frac{-\theta+3.\pi/4}{\pi/4}, \text{ if } \frac{\pi}{2} + \frac{11.\pi}{90} \leq \theta \leq \pi \\ -\rho, \text{ if } -\pi < \theta < -\frac{\pi}{2} - \frac{11.\pi}{90} \\ \rho \cdot \frac{\theta+\pi/4}{\pi/4}, \text{ if } -\frac{\pi}{2} \leq \theta \leq 0 \\ \rho, \text{ if } \frac{\pi}{2} \geq \theta \geq \frac{\pi}{2} + \frac{11.\pi}{90} \\ -\rho, \text{ if } -\frac{\pi}{2} - \frac{11.\pi}{90} \geq \theta \geq -\frac{\pi}{2} \end{cases}$$

$$L = \begin{cases} \rho \cdot \frac{-\theta+\pi/4}{\pi/4}, \text{ if } 0 \leq \theta \leq \frac{\pi}{2} \\ -\rho, \text{ if } \frac{\pi}{2} + \frac{11.\pi}{90} \leq \theta \leq \pi \\ \rho \cdot \frac{\theta+3\cdot\pi/4}{\pi/4}, \text{ if } -\frac{\pi}{2} - \frac{11.\pi}{90} \\ \rho, \text{ if } -\frac{\pi}{2} \leq \theta \leq 0 \\ -\rho, \text{ if } \frac{\pi}{2} \geq \theta \geq \frac{\pi}{2} + \frac{11.\pi}{90} \\ \rho, \text{ if } -\frac{\pi}{2} - \frac{11.\pi}{90} \geq \theta \geq -\frac{\pi}{2} \end{cases}$$

$$\text{where } \rho = \sqrt{x^2 + y^2} \text{ and } \theta = \operatorname{atan}2(-x, y)$$

Figure 3.14: Proposed Controller Mapping Method TODO

The observable changes to the algorithm previously proposed by [19] are very simple. We notice that the

$$\frac{11.\pi}{90}$$

was added to some of the conditions as well as four new expressions. The objective of this is to force that when the joystick is handled to a position as such, ρ is positive for one wheel and negative to the other. This way, if the joystick is handled to the following gaps, $\frac{\pi}{2} \geq \theta \geq \frac{\pi}{2} + \frac{11.\pi}{90}$ and at $-\frac{\pi}{2} - \frac{11.\pi}{90} \geq \theta \geq -\frac{\pi}{2}$, the wheelchair keeps spinning to the correspondent side, which helps users not to easily mistake from changing the direction of the wheelchair, therefore, preventing

going frontwards or backwards when the desired behaviour is rotate to the left or to the right. These gaps are better observable in the following picture 3.15:

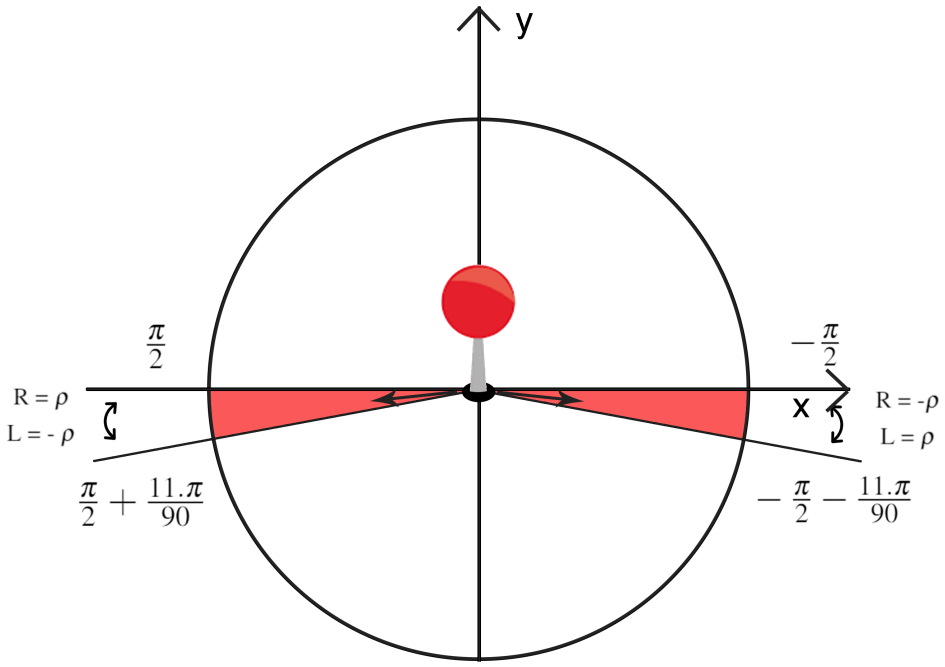


Figure 3.15: Joystick Gaps

3.2.2 Ball-Out-of-Bounds Mechanism

The next section aims to explain how the game logic handles the ball out of bounds. In order to understand this it is necessary to know how a ball is considered to be out of the bounds.

A ball is considered to be out of bounds when it is out of the field of play, thus, as in football, refers to being outside the playing boundaries of the field. Due to the chaotic nature of play, it is normal in many sports for players and/or the ball to go out of bounds frequently during a game, therefore, when a player kicks/passes/touches the ball and then the same goes out of the field of play, the opposite team is awarded with a free kick against the team who put the ball out of bounds.

When this happens, there exists three scenarios observable in the following figure 3.16:

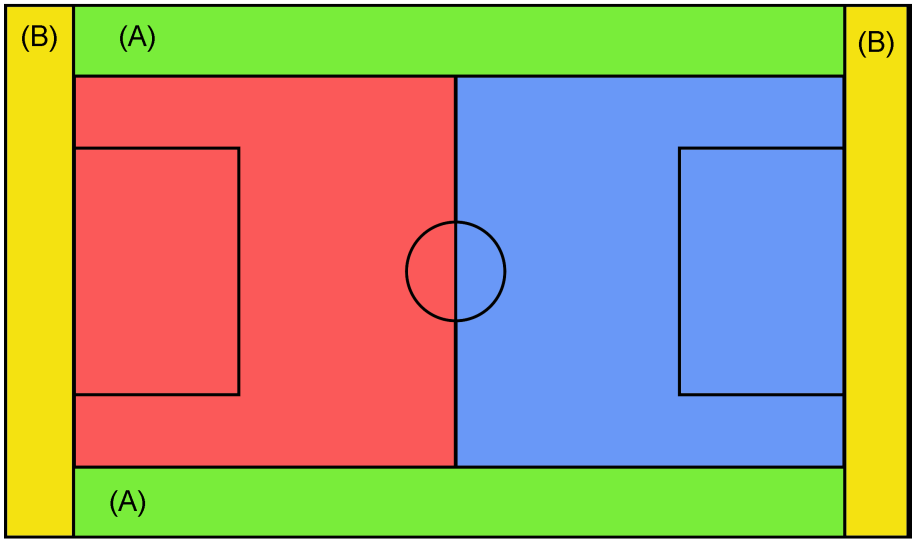


Figure 3.16: Field Out of Bounds areas

Scenario 1: If the ball goes out of bounds at area A, the team awarded with a free kick must kick the ball from the last position relative to the boundary line where it went out.

Scenario 2: If the ball goes out of bounds at area B and the correspondent area is the correspondent team area of the last player touching the ball, then, the team awarded with a free kick must kick the ball from the closest corner of that area where the ball went out. This kind of approach is known in football as a corner kick.

Scenario 3: If the ball goes out of bounds at area B and the correspondent area is not the correspondent team area of the last player touching the ball, then, the goalkeeper of the team awarded with a free kick must kick the ball from his own small area. This kind of approach is known in football as a goal kick.

With this reviewed, let's start discussing the approached technique to find this in the videogame.

While the game is running, there exists a variable at the main game class (GameEnvironmentInfo) that is responsible for tracking the last player who touched the ball. This is done in a very simple way. Every time that a player touches the ball, this variable is updated with the agent that touched it.

As previously discussed at 3.1.3.2, there exists a collider attached to the ball that, when it is in contact with the Out Of bounds area, triggers a CollisionEvent. Posteriorly, this Event sends a message to the main game class (GameEnvironmentInfo) informing that a Ball Out of Bounds occurred.

At GameEnvironmentInfo the information is processed and the game enters in a state where the time counter pauses, all the players are re-positioned in field and the player who kicks the ball

is chosen.

The player that is chosen to kick the ball is usually the player from the team that the kick was awarded and is closer to the ball. A second player from the same team is positioned near the player who kicks the ball in order to be the player who receives it. This was chosen to be this way because, since agents are trained with machine learning, and the train that was made in order to achieve the pass behaviour was relative to the closest teammate, this strategy benefices the pass behaviour. This approach will be seen in more detail at **??**. The remaining agents/players in field are re-positioned.

Re-position in this context means that, all the players in exception of the two referenced before, need to position themselves in field in such a way that seems natural and benefices both of the teams when a out-of-bounds period happens. For example, if team A is awarded with a out of bounds free kick, the team B must position in places where they can defend better they're own goal area, the contrary for team B, that must position themselves in order to attack the team A area.

Although it would be interesting to create a dynamic positioning mechanism to the players in field when a ball out-of-bounds occur, the approach to deal with this was to define multiple sections of static positioning for every player relative to pre-defined areas. More in concrete, outer field was divided in 8 areas, and, for each of them, there exists a pre-defined position for every player, in exception of the two ones referenced before, namely, the player kicking the ball and the player receiving it.

3.2.3 Two-on-One Foul Mechanism

As discussed before at 2.2.3, there exists two main fouls at this modality, the two-on-one and the three-in-the-goal. Two-on-One foul declares that there cannot be more than two players of the same team confronting directly one adversary which has the ball in its possession between a radius of 3 meters, therefore this section aims to explain how the two-on-one foul mechanism works in the game. We will discuss how a foul is detected in run-time and what happens next if a foul occurs.

3.2.3.1 Foul detection Algorithm

In order to detect if a foul as occur, it is necessary to check constantly if we are in a presence of a it while the game is running, therefore, there exists some methods provided by the Unity Mono-Behaviour API that are specifically used for this kinds of approach. These functions, that we will discuss in a brief, are the most commonly used function to implement any kind of game script, they are named Update, LateUpdate and FixedUpdate.

They are mainly responsible for being called every-time that a frame in the game is processed, being the main differences between the three, the following:

While Update() function is called every frame, LateUpdate() is called after all Update functions have been called. This is useful to order script execution, for example, a script that follows a

camera should always be implemented in LateUpdate because this tracks objects that might have been moved inside Update. FixedUpdate is called every fixed framerate frame, therefore, a good use of the FixedUpdate is to use it when dealing with Rigidbody, which are physic components. For example, when adding a force to a rigidbody, you have to apply the force every fixed frame inside FixedUpdate instead of every frame inside Update [64].

That said, the update function seems to be better for this propose, therefore, every-time that a frame is processed, the algorithm we will now discuss, is called. What is presented here is an algorithm that, in a brief, detects this foul and the players involved. The following pseudo-code 1 explain the algorithm.

Algorithm 1 Two-on-one Foul detection algorithm

```
1:  order AllPlayerList by ball.distance                                   ▷ (1)
2:  if (AllPlayerList[0].distanceToBall ≤ 1.5) then                        ▷ (2)
3:      playerWithBall ← AllPlayerList[0]
4:  else
5:      exit
6:  end if
7:  for (i = 1; i ≤ AllPlayerList.length; i++) do                          ▷ (3)
8:      if (AllPlayerList[i].team ≠ playerWithBall.team) then
9:          if (AllPlayerList[i].distanceToPlayerWithBall ≤ 3) then
10:             FirstOpponent ← AllPlayerList[i]
11:             for (j = i+1; j ≤ AllPlayerList.length; j++) do            ▷ (4)
12:                 if (AllPlayerList[j].team ≠ playerWithBall.team) then
13:                     if (AllPlayerList[j].distanceToPlayerWithBall ≤ 3) then
14:                         SecondOpponent ← AllPlayerList[j]
15:                         call FoulPeriod()                              ▷ (5)
16:                     end if
17:                 end if
18:             end for
19:         end if
20:     end if                                                             ▷ (6)
21: end for
```

(1) First, the list of all the players in field **AllPlayerList**, is ascendant ordered by the distance to the ball.

(2) Second, the algorithm runs a verification to understand if there exists a player with the ball in its possession, thus, checks if the first position of the list is the player who has the ball, named **PlayerWithBall**. This is necessary because if there is no player with ball, suppose it has been kicked and its away from any player, a foul cannot ever occur. The function that analysis this is done by checking if there exists any player in field which the distance to the Ball is smaller than 1.5 meters. If there's none, the function exits.

(3) Then, this list is iterated and, if there exists a player that is from the other team and is at a distance smaller than 3 to the **PlayerWithBall** we find that there exists an **FirstOpponent** which is confronting directly the **PlayerWithBall**.

(4) Again, if exists, the **AllPlayerList** is iterated starting by the position in the list of the **FirstOpponent** + 1 and the step 3 is repeated with a small change that, if there exists a player that is from the other team and is at a distance smaller than 3 to the **PlayerWithBall**, we have found the second opponent (SecondOpponent), which leads to a foul and a period of game foul, named **FoulPeriod**.

(5) At this point, we are in a presence of a two-on-one foul, thus, **FoulPeriod()** function is called. This function is responsible for initializing the period of a foul, which we will discuss further.

(6) After this point, point **3** is repeated.

3.2.3.2 Foul Period

As referenced before, when a two-on-one foul is detected, the **FoulPeriod()** function is called. These function is responsible for starting the Foul period.

The foul period is the period of the game where a foul is being handled, this means that, first, it is necessary to identify if the foul is a regular foul or a penalty, then, the time of the game needs to be paused until the ball is kicked, the player who will kick the ball must be chosen and the remaining players must be re-positioned in the field.

The following steps aims to explain how this information is handled.

(1) First, the game time counter that is being decremented in the update function is paused. This is achieved by setting a Boolean variable to true.

(2) Second, the player who is chosen to be the player kicking the ball is always the player who suffered the foul. Therefore, the player kicking the ball is the **PlayerWithBall**.

(3) As previously discussed at 3.1.3.2, there exists a tracking mechanism that keeps track of the players and the ball positions in field, therefore, we just need to check if the ball at the small area of the team who suffered the foul, and, if it is, the re-positioning of the players in field is done having in count that we are in a presence of a penalty. These re-positioning is done very similarly as the re-positioning previously discussed at Ball-Out-of-Bounds mechanism 3.2.2. All the players in exception of the one who is kicking the ball, position themselves statically in a pre-defined position strategy. These strategy can be observed at 3.17.

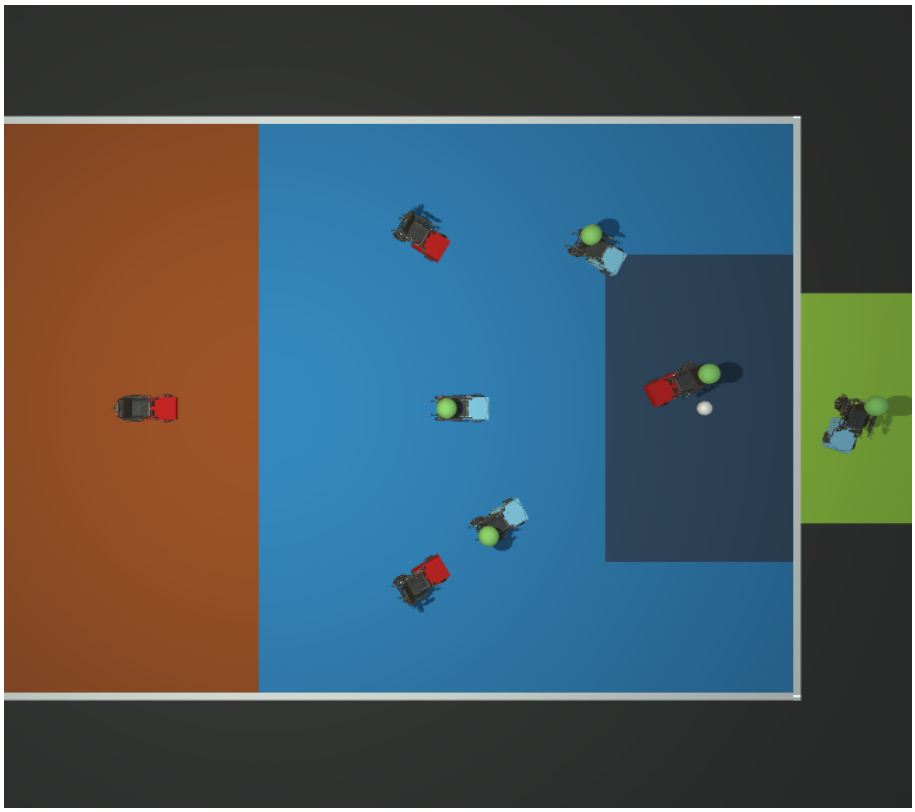


Figure 3.17: Penalty static player positioning in field

(4) If we are not in a presence of a penalty, then the re-positioning is handled just like the Ball-Out-of-Bounds mechanism 3.2.2. The unique difference is that, instead of diving the outer area in 8 parts, we divide the inner area of the field (big areas and small areas) in 8 parts. Then, we just need to check where the foul occur and set the specific pre-defined positions for that part of the field to each player.

(5) These period ends when the **PlayerWithBall** touches the ball for the first time. The game time and agents behaviours resume.

3.2.4 Three-in-the-goal-area Mechanism

As discussed before at 2.2.3, there exists two main fouls at this modality, the two-on-one and the three-in-the-goal.

We are in the presence of a Three-in-the-goal-area foul if a player is in possession of the ball and, at the small area of the opposite team, there exists more than two players of the same opposite team.

Again, in order to detect if a three-in-the-goal-area foul occurred it is necessary to implement a mechanism to detect it, therefore, a simple algorithm was designed. The following pseudo-code explains in detail how this is handled.

(1) The algorithm starts by checking if there's any player with the ball in its possession (**playerWithBall**), if there's none, the function exits.

Algorithm 2 Three-in-the-goal-area Foul detection algorithm

```
    if (playerWithBall = null) then                                          ▷ (1)
2:      exit
    end if
4:  if (playerWithBall.team = red) then                                      ▷ (2)
        if (playersAtSmallAreaBlue.Count ≥ 3) then                           ▷ (2)
6:          for (i = 0; i ≤ playersAtSmallAreaBlue.Count; i++) do
                if (playersAtSmallAreaBlue[i].team ≠ playerWithBall.team) then   ▷ (3)
8:                  teamMembersInAreaCounter ← teamMembersInAreaCounter + 1
                end if
10:         end for
        else
12:         exit
        end if
14:     if (teamMembersInAreaCounter ≥ 3) then
            call FoulPeriod()                                                ▷ (4)
16:     end if
    else                                                                     ▷ (5)
18:     if (playersAtSmallAreaRed.Count ≥ 3) then
            for (i = 0; i ≤ playersAtSmallAreaRed.Count; i++) do
20:             if (playersAtSmallAreaRed[i].team ≠ playerWithBall.team) then
                    teamMembersInAreaCounter ← teamMembersInAreaCounter + 1
22:             end if
            end for
24:     else
            exit
26:     end if
        if (teamMembersInAreaCounter ≥ 3) then
28:         call FoulPeriod()
        end if
30: end if
```

(2) Then, checks which team the **playerWithBall** is from. If the **playerWithBall** is from the RED team, the algorithm counts how many players are at the blue team's small area (**playersAtSmallAreaBlue**). If there's more than 3 players, all of these agents are iterated.

(3) These agents need to be iterated because we need to check if all the players that are at the area are actually from the opposite team of the **playerWithBall**, other way, we are not in the presence of a foul. For each player that is at the small area and its from the opposite team of the **playerWithBall**, a counter named **teamMembersInAreaCounter** is incremented.

(4) After all the **playersAtSmallAreaBlue** being iterated, there exists a condition to check if the **teamMembersInAreaCounter** ≥ 3. If it is, we are in the presence of a Three-in-the-goal-area Foul, and, therefore, the **FoulPeriod()** function, which is the same as referenced before at 3.2.3.2, is called.

(5) The same steps **(1), (2), (3), (4)** are done for the blue team (lines 18 to 30).

3.3 User Interface

A clear and well organised gaming scene makes gaming more interesting. Either if a user is new or expert, both of them need a game that is highly interactive and well guided. Good visual experience generates pleasurable video gaming experience. Usability attracts and retains more gamers. Adaptability to new technology and components makes it easy for gamers to adapt easily to new components and technologies and a highly responsive UI keeps the gamers to enjoy the game throughout. It is clear that the user interface contributes to the gaming experience.

The following sections aims to explain how the user interface was assembled for the IntellPowerSoccer. It is divided in three main categories, the Main Menu, the Pause Menu and the In-game screens and Interface.

3.3.1 Main Menu Screen

Every video-game requires to have a main menu. This is usually the menu that allows the user to choose what to do before playing the game. For instance, if we are playing a football game, there exists the play button, and when it is pressed, it takes us to a new scene where we can choose the team we want to play with. That said, the IntellPowerSoccer explores this kind of menu. When the user runs the game, he is presented with a main menu that has three buttons, a play button, a rules button and an exit button. Furthermore, and in order to beatify the user interface, there exists a wheelchair model that is spinning around itself. The following figure A.1 shows the final design.

Figure 3.18: Main Menu - User Interface

When one presses the play button, it is taken to a new scene with a transition animation where one can choose the team to play with. The user can play either with the red or the blue team. Figure A.3 shows this.

Figure 3.19: Main Menu: Choose Team - User Interface

When one presses the rules button, a menu with the three main rules is popped-up. Figure A.3 shows this.

Figure 3.20: Main Menu: Rules - User Interface

Although many video-games provide a settings menu usually to allow users to change graphics and controllers settings, since unity takes care of setting by itself, it was not found necessary to provide such menu in the IntellPowerSoccer.

Finally, the exit button closes the application.

3.3.2 Pause Menu Screen

While the game is running, there exists a pause menu that can be accessed by pressing the Escape keyboard key. Users may need to take a break from the game, therefore, this becomes very useful since it allows to pause the game, thus, pausing the timer increment and the the whole game elements and logic. The following figure A.5 represents the pause menu.

Figure 3.21: Pause Menu - User Interface

By observing fig. A.5 we may notice that there exists three buttons and a toggle box.

Resume button, as the name stands for, it resumes the game, thus, resumes the timer increment, game elements and logic.

Restart button restarts the game. This becomes very useful because users may not be happy with their score and may want to restart the game without closing the application. Furthermore and, since the game is new, it is susceptible to bugs, therefore, having a fast restart button may come in hand.

Main Menu button takes users to the main menu again, this way, they can change the team or look for the rules again.

Finally, there exists a toggle look around checkbox that is unset by default. This checkbox is responsible for locking or unlocking the first person camera controller. This controller allows users to look around by using the mouse-pad or the right controller joystick. The reason why the checkbox is unset by default is because users that almost do not play games find it harder to control the powerchair.

3.3.3 In-game Screens and Interface

We may notice in games that, many times, when a user makes the transition of a menu to the game itself, they are presented with a loading screen.

This loading screen is a traditional solution to the problem that loading the game data from HDD or optical drive or flash takes a long time, especially if the game is content-rich. This solution prevents user from getting impatient or thinking that the game has crashed when they see an empty screen.

Therefore, and since IntellPowerSoccer is a 3D game with some medium/high quality graphics, it was found necessary to provide a loading screen before game starting. The following fig. 3.22 shows the IntellPowerSoccer implemented loading screen.

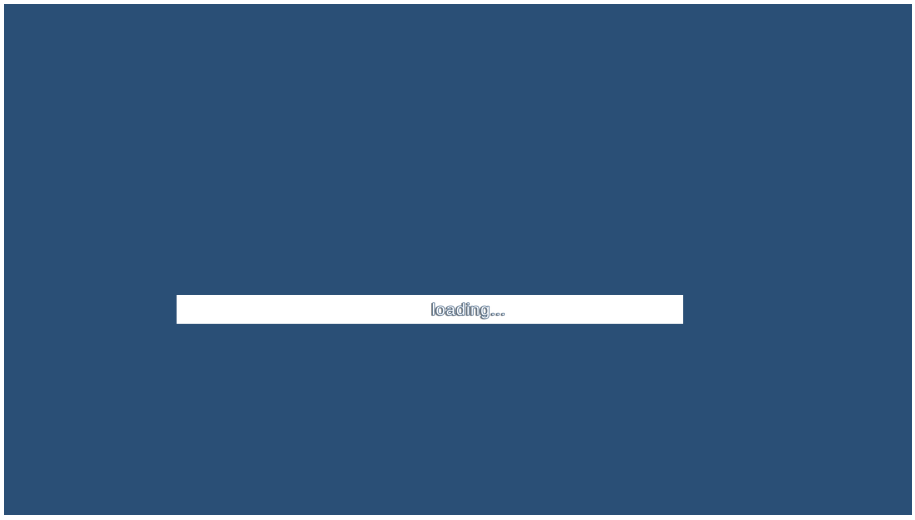


Figure 3.22: Loading Screen - User Interface

After the loading screen, the game starts, and, if we look to the top left part of the screen we may notice that there exists an UI component. Following figures 3.23 and 3.24 represent this.

Figure 3.23: Game view of the Score Board - UI Component

Figure 3.24: Score Board - UI Component

This component is responsible for presenting the actual score board of the game where the red square stands for the red team score, the blue square stands for the blue team score, and the white square marks the elapsed game time.

After 20 minutes of game-play marked by the counter in the score board, which are not real 20 minutes of gameplay because the clock is time-forwarded and, therefore, corresponds to real 5 minutes, the first part of the game ends and the following screen 3.25 is presented to the user.

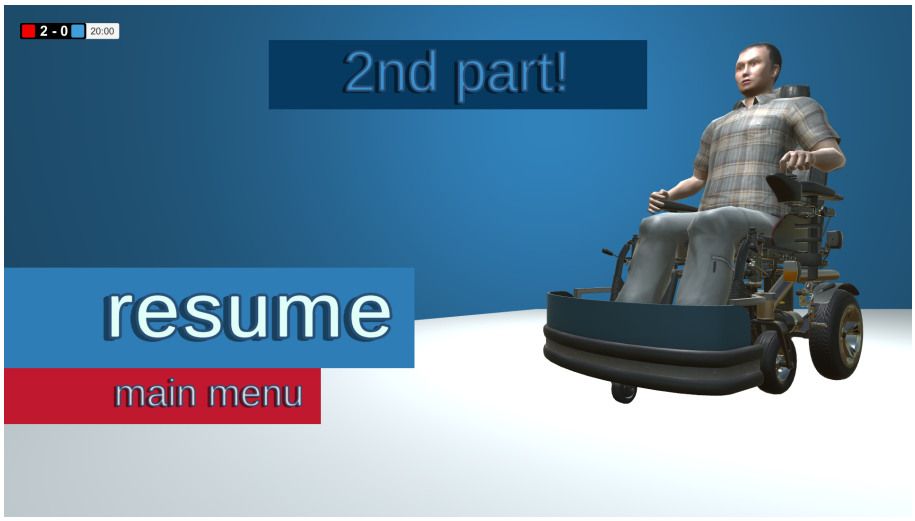


Figure 3.25: End of first Part - User Interface

Just like the end of the first part, by the end of 40 minutes of game-play, the game ends and the user is presented with the final game screen. Figure 3.26 represents this.

Figure 3.26: End of game - User Interface

3.4 Summary

By closing this chapter we were able to understand the importance of developing the game logic as well as the methodology adopted to achieve the game we have so far.

We have explored a strategy to allow the wheelchairs in game to rotate freely with they're own caster wheels, presenting a natural movement and realism.

All the models used for the wheelchairs, ball and field were presented, as well as all the algorithms used for developing the logic of the game, such as the 3-in-the-goal area foul mechanism,

the two-on-one foul mechanism, the ball out-of-bounds mechanism and the player tracking system.

It is now possible to understand how the controller of a real motorized wheelchair works and how it was assembled for this video-game.

Finnaly, we have seen the user interface implemented as well as all the possible interaction a user can have with the game.

Chapter 4

Agents

As discussed before at 2.3, in order to make the game more entertaining and enjoyable, it is necessary to develop agents that somehow know how to play the game. Many games use Artificial Intelligence algorithms to achieve desired behaviours but since we want to achieve some realism and because the essence of the powerchair football is to freely move the chair and rotate the wheelchair to strike or pass the ball, we found more interesting to use machine learning, therefore, what is proposed in dissertation is to develop agents that will learn to play by using machine learning.

Before discussing how the behaviours will be approached, it is necessary to understand that there exists many different algorithms to achieve machine learning behaviours. The ones we will focus the most are based in reinforcement learning and imitation learning. We will discuss this in the following sections.

4.1 Lower-Level Behaviours

Developing machine learning behaviours can be quite a challenging task, either because the training environment is quite complex, leading to a very slow learning rate, or because the approached algorithms and configuration values are not good for developing the desired behaviour.

Therefore, in order to achieve the desired behaviours it was proposed to divide this problem in two sub-problems. Instead of trying to develop a big behaviour with all agents at the same training environment, it is proposed to develop small and simple behaviours and then make use of them at a higher level behaviour where all the small behaviours will be handled.

Furthermore, and since this has been proved to be a good API while developing Machine Learning behaviours in Unity, we will use Unity ML-Agents for developing all the following behaviours.

Before start discussing, it is necessary to understand some ML-Agents important definitions such as: what is an agent (1), what are observations (2) and what are actions (3).

(1) An agent is an entity that collects observations within an environment, decide on the best course of action using those observations, and execute those actions within its environment. When

creating agents, there exists two important aspects for them to successfully learn a desired behaviour, namely, the way the agents collect the observations and the reward that one assigns to estimate the value of the agent's current state toward accomplishing its tasks.

(2) Observations are data that the agent can collect from its environment. Collecting this data can be done in several ways, for instance, if we consider an environment in 2D, and we want to teach an agent to move to a point in the field, we can simply provide the point to go as a Vector of two dimensions (x,y), on other hand, if we are dealing with 3D environments, and we want to teach an agent to move a block from one point to another, there exists the need of the agent to see the object he is learning to move, therefore, in this scenario providing Ray-casts may be a better option.

(3) Actions stand for what exactly the word means, they are allowed actions that an agent can perform in such environment. For example, if we want to teach an agent to move to a point, we need to provide the agent a controller in a way he can move to the desired point, in other words, we most provide the mechanism that allows the agent to move forward, backward ans so on.

4.1.1 The Pass Behaviour

At Powerchair football, although there exists multiple ways of kicking the ball to a desired place, either scoring goals or passing it, the most common way of kicking is to make the wheelchair spin and kick the ball with the foot-guards (or feet). This way the ball is projected with more velocity leading to longer passes and stronger strikes.

When watching a powerchair football game, we can clearly observe that players make use the most of their footguards, which makes sense because it is the only way allowed of kicking/dribble the ball. Furthermore, many times, these players use a common strategy which is to pass the ball to a desired teammate and this teammate takes advantage of the ball velocity and then kick it again to score. This strategy is very useful because players can shoot the ball with more velocity and provide a harder task to the goalkeeper. That said, the first behaviour we will discuss is the Pass behaviour.

The main objective of this behaviour is to pass the ball to a teammate in a way that seems natural, therefore, the agent should be able to rotate the wheelchair while aiming at the ball and kick it targeting the teammate, just like the real powerchair football users do it.

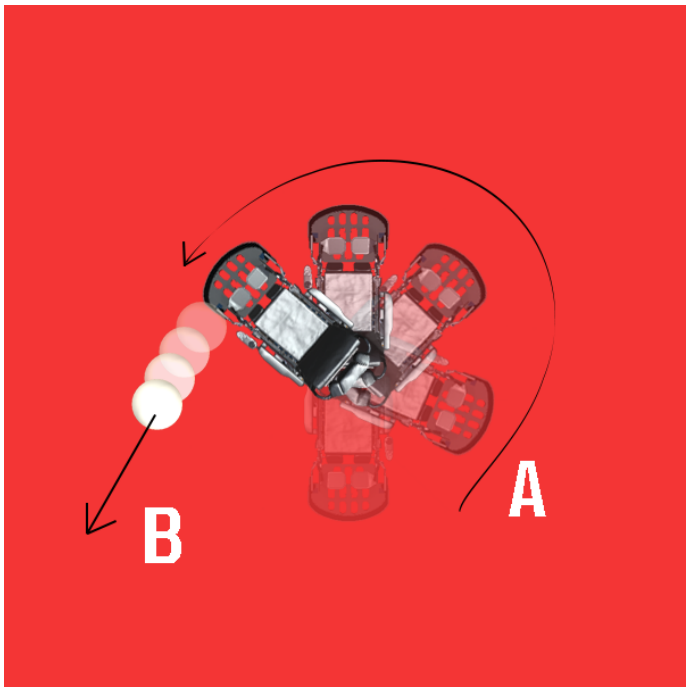


Figure 4.1: Demonstration of a real powerchair football kick

Figure 4.1 is a demonstration of how a real powerchair football player kicks the ball with his foot-guards. **A** is the rotation of the wheelchair while aiming at the ball and **B** represents the kick and the ball projection when the wheelchair touches it.

The proposed solution for training this behaviour is to use Reinforcement Learning and Imitation Learning algorithms. We will discuss this in the following sections.

4.1.1.1 Environment Observations

The training environment is defined by the field of play, two powered wheelchairs and the Ball. One of the agents is the one to be trained and the other is the agent who will receive the ball, therefore, is teammate. The field was already discussed at 3.16 as well as the wheelchair 3.1, which are, in fact, the model of the agents.

To achieve this desired behaviour, the agent will collect 6 different observations, thus, the observation vector size is 6. This collected observations are described as follows:

AgentAngularVelocity	Agent current angular Velocity in float.
AgentDistanceToBall	Agent distance to the ball in float.
AngleBetweenAgentAndBall	Angle between the agent and the ball in degrees.
AngleBetweenAgentAndTeammate	Angle between the agent and the teammate in degrees.
AgentDistanceToTeammate	Agent distance to teammate in float.
NumberOfTouches	Number of touches the agents did at each training episode.

Table 4.1: Pass the ball observation vector values

AgentAngularVelocity, as referenced at 4.1 is the agent current angular velocity, more in concrete, the angular velocity of the wheelchair. This is a necessary observation because it helps the agent know the exactly angular velocity he should apply to spin the wheelchair and kick the ball further or closer. A good example where this is very useful is, for instance, if the teammate distance to the agent is closer, then, the agent should kick the ball with less force than if the teammate is further away. Furthermore, in previous experiments training this behaviour, it was found that the agent didn't know how to spin the wheelchair correctly.

AgentDistanceToBall is the agent distance to the ball. This has proven to help the agent understand if the wheelchair is closer or further away when kicking the ball. **AgentDistanceToTeammate** follows the same ideology, providing the distance of the agent to the teammate.

AngleBetweenAgentAndBall is the angle that the wheelchair footguards do with the ball. This has proven to be useful because in previous training sessions the agent took too long to find the right angle to kick the ball to the desired player and by adding this observation, the agent was faster to understand how to kick. The same happened by adding the **AngleBetweenAgentAndTeammate**, which is the angle between the agent wheelchair footguards and the teammate.

Finally, **NumberOfTouches** is responsible for providing the number of thouches the agents did with the ball. This was proven to be very useful because in previous training sessions the agent started to learn to dribble the ball instead of passing it. Therefore, in order to contradict this behaviour, the agent is penalized equally as the number of touches he does. If he kicks the ball only one time and the ball is received by the teammate, then, the agent wins the maximum reward, on other hand, he wins a smaller reward proportional to the number of touches if he touches the ball more than one time.

4.1.1.2 Ray Perception Sensor Component

Although all of this observations seem to be enough to make the agent learn the desired behaviour, the training session was proven to be more reliable when adding a Ray-cast component as well.

At Unity ML-Agents, Rays-casts work as an independent observation vector, therefore, there is no need of adding it to the agent observation vector. Furthermore, what they do is to work as a sensor that detects colliders, in other words, they project a ray that when colliding, for example, with the collider of a wheelchair, provides observable information to the agent. The following image describes how the Ray-cast was configured for the agents.

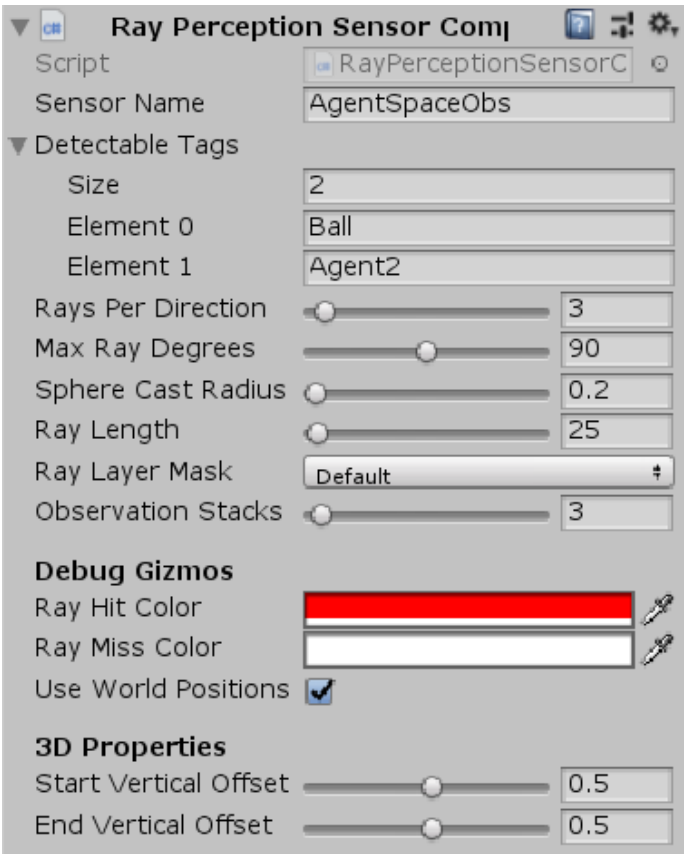


Figure 4.2: Ray-cast Configuration

At the following figure 4.2 there exists multiple parameters such Detectable Tags, rays per direction, max ray degrees and ray length. These parameters are configuration parameter that, in its majority, are responsible for positioning the rays in the desired object.

For instance, Rays per direction are the number of rays that one desires to project. Max Ray Degrees is the angle of which this Rays per direction spread themselves. Sphere radius is the size of the sphere of the ray when colliding with an object.

Either way, these parameters are mainly responsible for positioning the "sensors" in the desired position at the wheelchair. In this case, it was chosen to apply two Ray Perceptions sensors to achieve better results, one was positioned at the level of the rears, facing frontwards, and the other was positioned at the foot-guards of the wheelchair.

This was chosen to be this way because, this way, when the agent is almost touching the ball with his foot-guards, he can "see" where the ball is. This happens because the ray-casts are being projected from its own foot-guards. On other hand, the ray-casts that are being projected from the rears are mainly used to detect objects that are further away from the agent, for instance, a teammate agent or the ball.

The following figure 4.3 aims to give a clear perspective of how these "Sensors" are working along the training session. The red ray represents a ray-cast hitting an object, thus, collecting observable information while a white ray represents that there's no other object along the field

that is important to be collected as an observation. Notice that there exists longer white rays than others. This is precisely the difference between the two Ray Perception Sensors Components, the less longer rays are the footguards sensors and the the longer ones are the rear sensors.

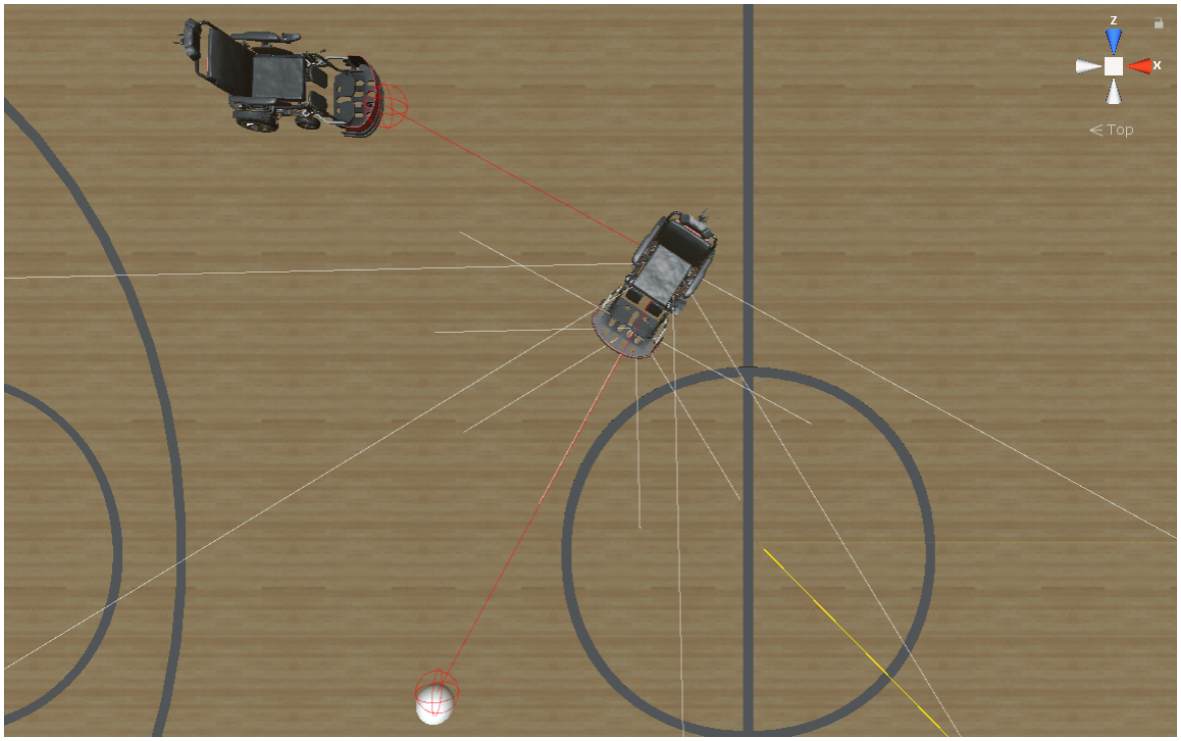


Figure 4.3: Ray-cast Hit example

As we will see in the next sections, these sensors will be used for all the behaviours.

4.1.1.3 Reward System

As previously discussed at 2.3, reinforcement learning can be seen as an area of machine learning which its major objective is learning what to do, how to map situations to actions, as well as to maximize a numerical reward signal. Therefore, in order to train an agent it is necessary to develop a reward system.

At this behaviour, every episode of train is responsible for executing the following steps:

(1) First, the agent and the ball spawns randomly inside the field of play.

(2) Then, the teammate spawns in a point randomly inside of a circle which has center in the agent and radius of 6 unity meters.

(3) Every-time that the agent touches the ball, the **NumberOfTouches** is incremented and a timer, named **timeOfFullPass**, starts counting. This timer is important for calculating the final reward, the greater this value, the smaller the reward. This way, agent becomes stressed and tries to accomplish his task faster.

(4) If the ball distance relative to the teammate is smaller than 1, then the pass was successful, thus, a reward is provided. This reward is calculated as follows:

$$\frac{10}{timeOfFullPass} - 0.2.numberOfTouches$$

(5) Other negative rewards were added in order the agent did not go outside of the field or kicked the ball to undesired positions such as the out of bounds area.

4.1.1.4 Demonstration Recorder Results

Although many Reinforcement learning algorithms provide good results, they tend to be very slow achieving a desired behaviour, they take too many episodes to start learning something and this can be extended to days/weeks/months or even years, depending on the complexity of the environment, observations and actions.

The firsts tries running this Pass the ball training sessions were done using Reinforcement Learning only, therefore, it did not took to long to notice that after hours of training the agent did not learn anything satisfying. This can be observed in the following diagram 4.4.

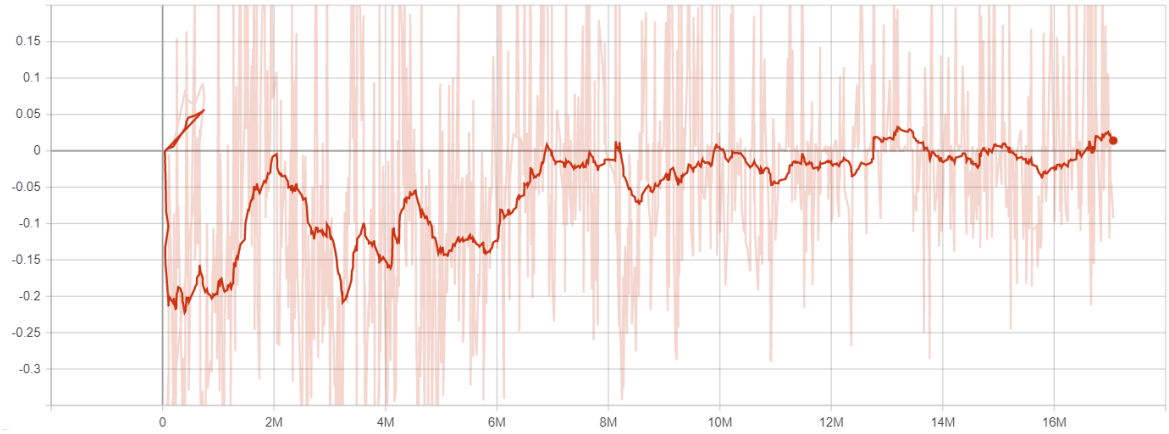


Figure 4.4: Cumulative Reward of the Pass the ball Train first run

In the following diagram, that represents the cumulative reward over the time for the reinforcement learning approach in the pass the ball behaviour, we can observe that, for a number of steps of 16 million, the reward oscillates to much over -0.2 and 0.05. 16 million steps are equivalent to 2 days of training running. That said, by observing this, we clearly understand that the agent is not being able of learning the proposed task. Although this happened, it does not mean that the agent would never learn anything with this first approach, it just told us that it might took too long to learn anything. Precisely because the time-consumption of this, it was decided to use Imitation Learning in addiction.

As referenced at 2.3, the Imitation Learning idea is that the learning agent observe the actions of an experienced agent as well as the corresponding consequences in a way that the observations give the learner information or clues on how to successfully perform the required task.

Unity Ml-Agents provides a Component for recording in-game demonstration. This component is named Demonstration recorder and its responsible for recording all the observable values as well as actions done by one controlling the agent, thus, an expert. This is precisely what was

done. 30 Episodes where recorded by one driving the wheelchair and kicking the ball to the teammate and therefore used to serve as an expert to run a new train using PPO, Behavioural Cloning and Gail algorithms. The following parameters were used to configure the training session:

```yaml
PassingTheBallImiReinf:
    summary_freq: 30000
    time_horizon: 128
    batch_size: 128
    buffer_size: 2048
    hidden_units: 512
    num_layers: 2
    beta: 1.0e-2
    max_steps: 5.0e9
    num_epoch: 3
    behavioral_cloning:
        demo_path: Assets/Demonstrations/PassTheBall.demo
        strength: 0.5
        steps: 150000
    reward_signals:
        extrinsic:
            strength: 1.0
            gamma: 0.99
        curiosity:
            strength: 0.02
            gamma: 0.99
            encoding_size: 256
        gail:
            strength: 0.01
            gamma: 0.99
            encoding_size: 128
            demo_path: Assets/Demonstrations/PassTheBall.demo
```

Figure 4.5: Parameters used for PPO, Gail and Behavioural Cloning algorithms

The result observable after some hours of train are quite noticeable, the agent cumulative reward is almost exponentially growing at the first 1.35 million step and the cumulative reward by the end of 8 million steps almost hit the maximum allowed reward which is 4. This means that the agent learned to do the desired behaviour. The following diagram proves this:

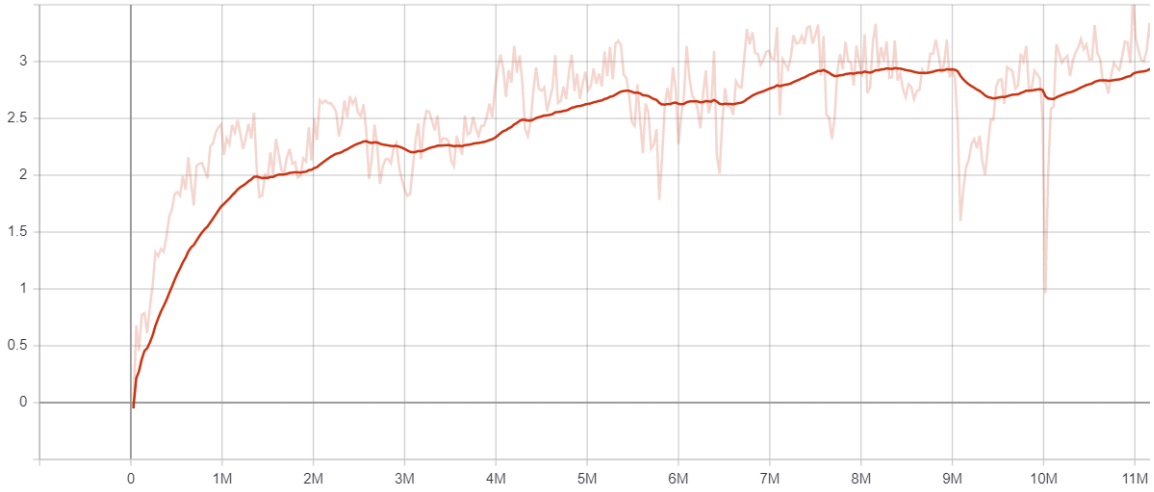


Figure 4.6: Cumulative reward of the Pass The Ball Train with Gail, BC and PPO

4.1.2 Strike Behaviour

At Powerchair football players must score goals. The team with more goals by the end of 40 minutes of playing, wins the match. With this, comes the need of agents learn how to kick the ball to the goal area, therefore, this sections aims to explain how the behaviour for striking the ball was approached.

The following behavior is very similar to the last one because explores the rotation of the wheelchair to kick the ball, the difference is that we do not want the agent to pass the ball but instead kick it to the goal area. Once again, by observing figure 4.1, we can better understand how real powerchair football players use the foot-guards to kick the ball.

4.1.2.1 Environment Observations

Once more, the training environment for this behaviour is similar to the one in Pass The Ball with the small exception that there is no teammate, therefore, the environment is composed by the field, the agent and the ball.

The observation vector for this behaviour has size 4 and is composed by the following observations:

AgentAngularVelocity	Agent current angular Velocity in float.
AgentDistanceToBall	Agent distance to the ball in float.
AngleBetweenAgentAndBall	Angle between the agent and the ball in degrees.
NumberOfTouches	Number of touches the agents did at each training episode.

Table 4.2: Strike Behaviour observation vector values

Each observation parameter follows exactly the same strategy as 4.1. Furthermore, the same Ray Perception Sensor Components, previously referenced at 4.1.1.2, were added as well.

4.1.2.2 Reward System

As discussed at 4.1.1.3, every behaviour that runs with Reinforcement Learning algorithms needs a Reward system. This behaviour was trained with the following one:

(1) The agent spawns randomly inside one of the small areas, as well as the ball. He is not allowed to spawn in other place rather than inside one of the small areas. This was chosen to be this way because makes the train less complex. For instance, if the agent spawns in a random place far away from the opponent goal area, he has to learn first to dribble the ball to a position closer to the opponent goal area and then learn to strike and score.

(2) As discussed before at 4.1.1.3, every-time that the agent touches the ball, the **NumberOfTouches** is incremented and **timeOfFullPass** starts counting. This is necessary because, since it is a new behaviour, we need that the agent learns how to kick the ball again.

(3) If the agent touches the ball, then, we start checking if a goal was score. If it was, a reward is calculated and provided as follows:

$$\frac{10}{timeOfFullPass} - 0.2.numberOfTouches$$

(4) Once more, other negative rewards were added in order the agent did not go outside of the field or kicked the ball to undesired positions such as the out of bounds area.

4.1.2.3 Results

Since we have proved that imitation learning speeds the process of learning, in order to achieve this desired behaviour it was used Demonstration Recorder as well, thus, allowing the use of imitation learning in junction with reinforcement learning.

Again, one played the game in this environment and recorded 35 episodes with the demonstration recorder of an agent kicking the ball and scoring a goal. PPO, Gail and Behavioural Cloning were the algorithms used to train this behavior. The parameters used are the same ones discussed at 4.5.

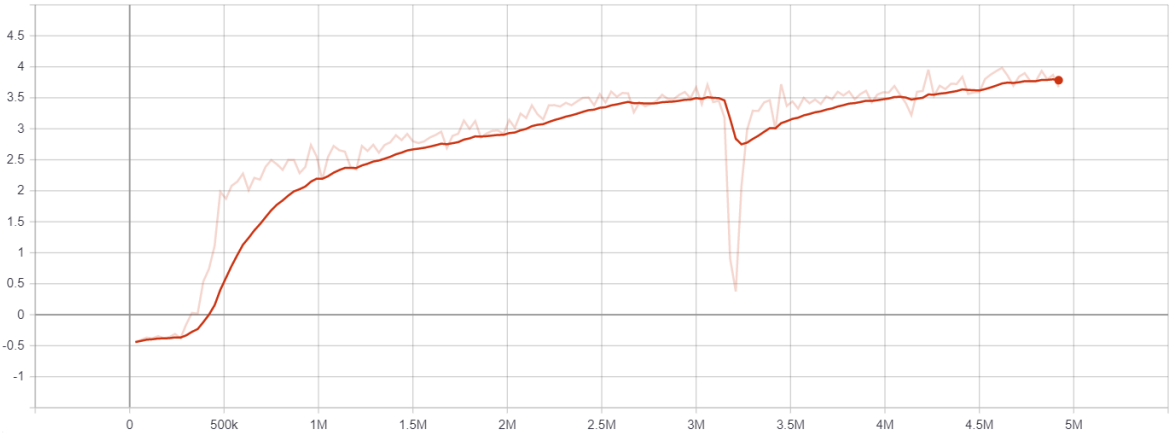


Figure 4.7: Cumulative reward of the Strike Train with Gail, BC and PPO

By observing the following diagram, we can clearly notice that the cumulative reward after 5 million steps approximates to the maximum reward which is 4, therefore, proves that the agent's task has been successfully learned.

4.1.3 Intersect-the-ball Behaviour

While a powerchair football game is running, there exists the need of one trying to intersect the ball in field when his team is no longer in possession of the ball, therefore, and since each team are always trying to have the ball in their own possession, players must pursue the ball along the field and try to acquire it. One good way of stealing the ball from an opponent is, for instance, to take advantage of a pass that one does and try to intersect the ball along. This is precisely what this behaviour we will now discuss focus to achieve.

4.1.3.1 Environment Observations

The Environment of training for this behaviour is composed by the field of play, an agent with the Pass behaviour active, an agent without any behaviour active, the agent to be trained and the ball. The first agent, **passAgent**, is the one that will pass the ball to the second agent, which is the **receiveAgent**, thus, his teammate. The agent to be trained, **agent**, which is from the opponent team, will try to intersect this pass in order to steal the ball.

The following table 4.3 describes the used observation values.

passAgentPosition	passAgent current position in-field.
receiveAgentPosition	receiveAgent current position in-field.
agentDistanceToBall	agent distance to the ball.

Table 4.3: Intersect-the-ball behaviour observation vector values

The same Ray Cast Sensor components used in the Pass Behaviour were used as well for this behaviour training approach.

4.1.3.2 Reward System

The following steps aims to describe in a chronological way how the Reward System for the Intersect-the-ball behaviour was approached.

(1) The **agent** spawns randomly at the field.

(2) **passAgent** and **receiveAgent** spawn at a distance between 3 and 5 of each other, at a random position in field as well. This spawn strategy comes with an exception, it makes sure that both do not spawn near the **agent**, more in concrete, they must spawn at least at a distance greater than 3 from the **agent**.

(3) After spawn, the ball is then positioned near the **passAgent**, inside a radius of 2 with center at the **passAgent**. By combining this step and step **(2)** exception, we make sure that the **passAgent** has the ball in its possession.

(4) While training, if the **agent** approximates too much from the **passAgent** or the **receiveAgent**, he is penalized with a reward of -0.01 for each second he is near them.

(5) Finally, if the agent touches the ball, he is rewarded with a score of 5, thus, he completes successfully his task.

4.1.3.3 Results

By using PPO, Gail and Behaviour Cloning algorithms, and by providing 30 episodes of Demonstration Recorders, the results of both behaviour training sessions can be observed in the following diagram.

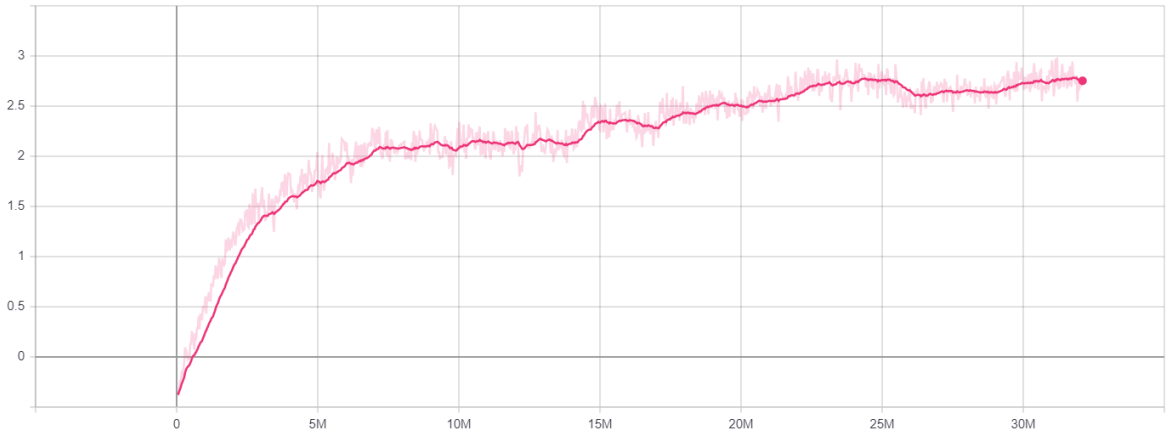


Figure 4.8: Cumulative reward of the Intersect-The-Ball Train with Gail, BC and PPO

By observing the following diagram we can notice that, compared with other previously discussed behaviour results, the train did not approximate too much to the maximum cumulative allowed reward, which is 5 for this reward system.

We can also observe that the training session ended around step 32 million, which for this train, corresponds to 3 days of running time.

Although there's no conclusive explanation for this results, we can notice that the training environment is very complex when compared with other in this dissertation, therefore, one hypothesis for this bad results may be not having enough training time, on other words, this train may needed more running time to approximate closer to the maximum allowed cumulative reward. Other hypothesis is that the observation values or the approached reward system is not sufficient to teach the agent what to do.

Even though this seem bad numbers, when watching the behaviour in the game running inference, the results do not seem so unsatisfactory. The **agent** is capable of intersecting the ball in it's majority of time with the exception that he approximates closer to the opponents than desired. Either way, he is capable of stealing the ball to the opponent and, although not perfect as desired, he full-fits the requirements.

4.1.4 Goal-Keep Behaviour

Every powerchair football game requires to have a goalkeeper at each team, therefore, it is necessary to develop a behaviour where an agent can successfully defend a strike from an opponent. The following behaviour we will discuss approaches the training session, mainly, with the objective of defending a goal in a penalty situation. It was chosen to focus this train in a penalty situation because, since a goalkeeper can move in the field too and any player can try to avoid a goal, the more specific situation where a strike should be defended by the goalkeeper is at a penalty situation.

That said, lets review some necessary rules applied to a penalty situation:

(1) The goalkeeper must defend the penalty at the back of the goal line.

(2) The striker, which is the player kicking the ball in a penalty situation, should kick the ball from the penalty mark.

4.1.4.1 Environment Observations

After discussing how a penalty situation should be marked, let's now see how the training environment will be handled .

The training environment will be represented by the field of play, the ball, the agent to be trained and the striker. In this context, the striker, as previously discussed at 4.1.2, is a trained agent responsible for kicking the ball and score a goal, therefore, this agents will have its correspondent behaviour active.

The following table represents the collected observations from the environment.

AgentDistanceToBall	Agent distance to the ball in float.
AngleBetweenAgentAndBall	Angle between the agent and the ball in degrees.
DistanceToKicker	Distance from the agent to the kicker in float.
AngleBetweenAgentAndKicker	Angle between the agent foot-guards and the striker footguards.

Table 4.4: Goal-Keep Behaviour observation vector values

We may notice that some observation values have been maintained from the previous behaviours. This is mainly due to the results of previous behaviours, which have demonstrate that these are good observations.

AgentDistanceToBall continues to be very important for the agent in this train in order to understand where the ball is at field.

AngleBetweenAgentAndBall helps the player find the correct rotation angle with the ball to defend the strike.

DistanceToKicker is responsible for providing the distance of the player who will kick the ball, this helps the agent understand if the striker is ready to be kicked or not and, furthermore, helps position himself better.

AngleBetweenAgentAndKicker as proven to be useful for the same reason as **DistanceToKicker**, helping the agent understanding where he should position himself with the better rotation at the moment of the kick.

4.1.4.2 Reward System

Although an agent only needs to learn how to defend his own goal from a strike, it was found more reliable to teach the same agent either to defend is goal or the opponent one. This is manly because if we approached this problem by teaching each goalkeeper to defend his own goal, it would be necessary to train two behaviours separated, which would have been more time-consumption and even lead to different results.

(1) That said, it was proposed to teach the agent to defend both of the goals. Therefore, firstly, the agent is positioned either at the blue goal or the red goal. This is decided randomly, 50% chances of defending the blue or the red goal. Since the rules obligate to it, the agent is positioned behind the goal line of the correspondent goal. At the same time, the striker is positioned at the correspondent penalty mark.

(2) After agents positioning, the striker enables his behaviour and starts kicking the ball to the goal. When the striker first touches the ball, a boolean flag, named **oponentStriked**, is activated. This flag is responsible for allowing the goal-keeper agent to check if he has successfully defended the ball. This prevents the goal-keeper agent to go chase the ball before the striker shoot it, which is not allowed.

(3) After **oponentStriked** flag is set to true, the goal-keeper agent starts checking if he has touched the ball and, if so, a reward of 4 is awarded.

(4) Even thou the goal-keeper touches the ball after an opponent strike, a goal may not be avoided since the trajectory of the ball may lead to a goal. That said, it was necessary to do something else to effectively indicate the agent that he had successfully avoided a goal. The solution proposed is the following:

(4.1) If the agent touches the ball after a strike, he wins a final reward of 5, but, if, after that, a goal is scored, the reward is penalized by adding a reward of -2, therefore, is final reward is 3. Although it's a positive reward, its not the maximum reward possible, thus, the agent will eventually find that he can win more than 3, leading him to explore how to maximize this. This is precisely what is desired, to have a goal keeper that can learn how to effectively defend a penalty.

Although this solution does not directly indicate the agent he successfully avoided a goal, it teach the agent to maximize is reward, therefore, it indirectly indicates that he is doing a good or a bad job.

(5) Other negative rewards were added in order the agent did not go outside of the goal area.

4.1.4.3 Results

Just like the previous behaviours, Demonstration recorder was used to speed up the process of training. 35 episodes were recorded of one defending a kick from the striker agent and used in

Gail and Behavioural Cloning algorithms. PPO was also used.

The following diagram 4.11 compares two training runs for the same reward system and algorithm values, with the small exception that one uses a small value in the curiosity value and other uses a bigger one.

By observing the following diagram we can clearly understand that the red graph which has a curiosity value greater provides better results than the red one.

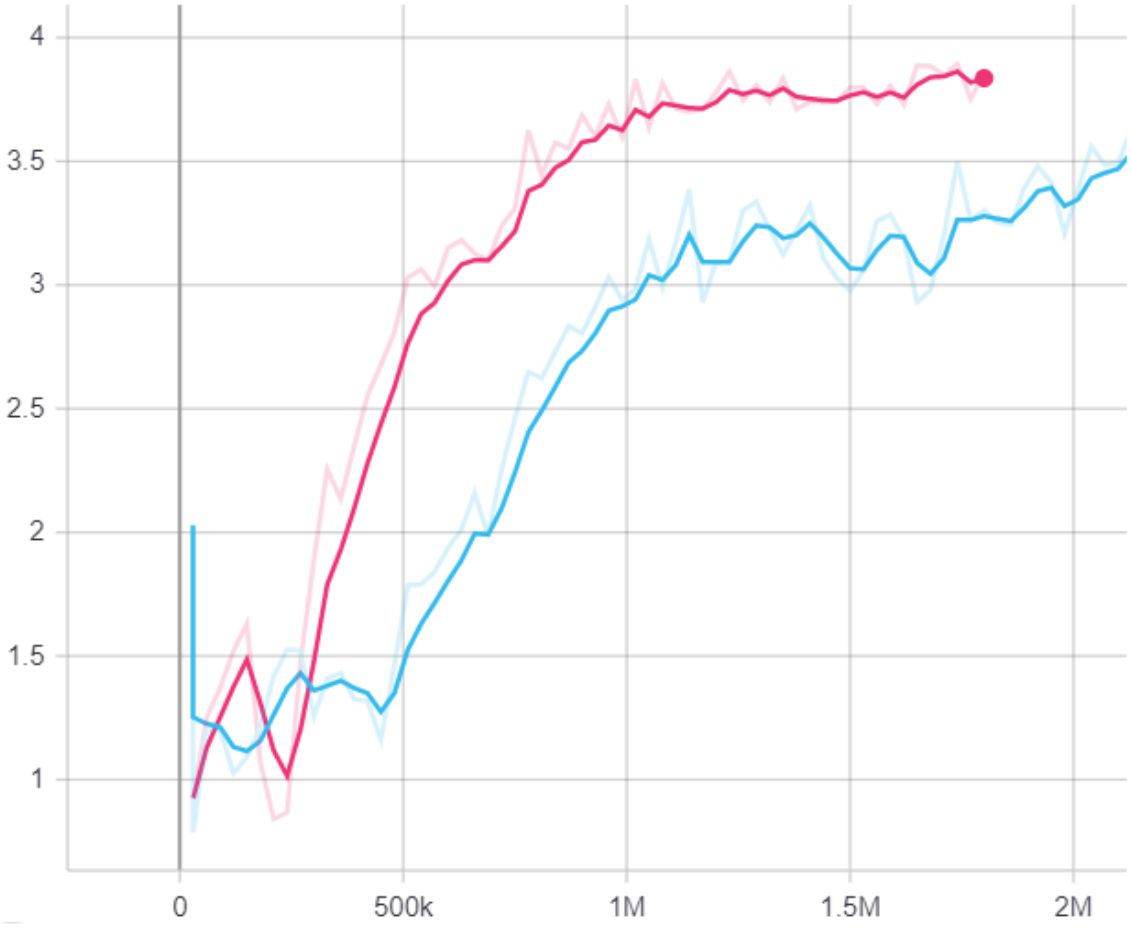


Figure 4.9: Cumulative reward of the Strike Train with Gail, BC and PPO

Although the unique difference between this two trains is the curiosity value, it does not necessarily mean that this was the great difference for achieving better results in one comparing to the other. This is said because the train has an random factor, therefore, sometimes, even for the same environment, variables, configurations, one train may provide different results than another.

That said, the train represented by the red graph was the one chosen to represent the Goal-Keep Behaviour since is the one that is closer to tha maximum allowed cumulative reward.

4.1.5 Dribble Behaviour

Although many of the powerchair football players use the pass as a manner of keep the ball circulating in game, dribbling is another great and effective way of playing. The player can use the foot-guards to dribble the ball or even score a goal, therefore, there exists the need of creating an agent behaviour where one can successfully dribble the ball by using it's own foot-guards.

If we focus in nowadays video-games, more specifically in football video-games, we can notice that a player or agent dribbles the ball in a way that the ball seems to be locked to its foot. Some projects even claim to use delaunay triangulation diagrams to facilitate the pass from a player to another. For instance, one common strategy, is to use the edges of the delaunay as the path of the ball for the pass between players, while the players are the vertex of each edge intersections.

Although this strategy seems to be indicated for the majority of the football video-games, when compared with this project, where exists the need of a player to simulate wheelchair driving and it is expected that a player is be able of performing a pass by rotating the wheelchair as desired, this strategy doesn't seem to good enough. We rapidly conclude that the ball cannot be locked in the wheelchair foot neither its useful a delaunay triangulation for the pass. It is precisely because of this that the previously discussed Pass Behaviour 4.1.1 was approached has it was.

That said, we will discuss now a strategy for training the dribble by using points in the field that represent the point where the player should take the ball.

4.1.5.1 Environment Observation

What is proposed to handle this behaviour is to teach an agent to go to a desired point in the field of play while dribbling the ball to it.

In addiction, it was chosen to approach this behaviour as follows because the Higher Behaviour that we will discuss in more detail further away in this dissertation will need a way of command the agents in the field to create a dynamic positioning system, in other words, provide a mechanism that is constantly updating the position of the agents in a way that gives advantage to the team. Therefore, in order to minimize problems in the future, it was found a good policy to train an agent to learn to dribble the ball to any desired point in the field.

That said, let's discuss how the environment of train was handled. The environment is represented by the field of play, the agent to be trained, the ball and a point, which is a 3D sphere object with a collider. Table 4.5 represents the observable values an agent can take from it's environment.

DistanceFromBallToPoint	The distance from the ball to the point to go.

Table 4.5: Dribble Behaviour observation vector values

The observation vector is composed by only one value which is the DistanceFromBallToPoint, responsible for providing the distance from the ball to the point where the agent must take the ball.

This behaviour only needs this observation value because the raycasts from the wheelchair provide the remaining important information. The raycast, in this context, aims to detect where

the point to go and the ball are at the field. We will discuss this in more detail in the following Reward System section.

4.1.5.2 Reward System

The following steps represent the approached solution for the reward system:

(1) First, the agent spawns in a random point inside the field of play. The pointToGo, which is the 3D Sphere Object that represents the point where the agents should learn to go while dribbling the ball, is then spawned in a random point inside the field of play. Finally, the ball spawns in a random point inside of the field as well. Figure 4.10 represents an example where the agent, the ball and the poinToGo might be and provides an example path the agent must learn in order to take the ball to the pointToGo.

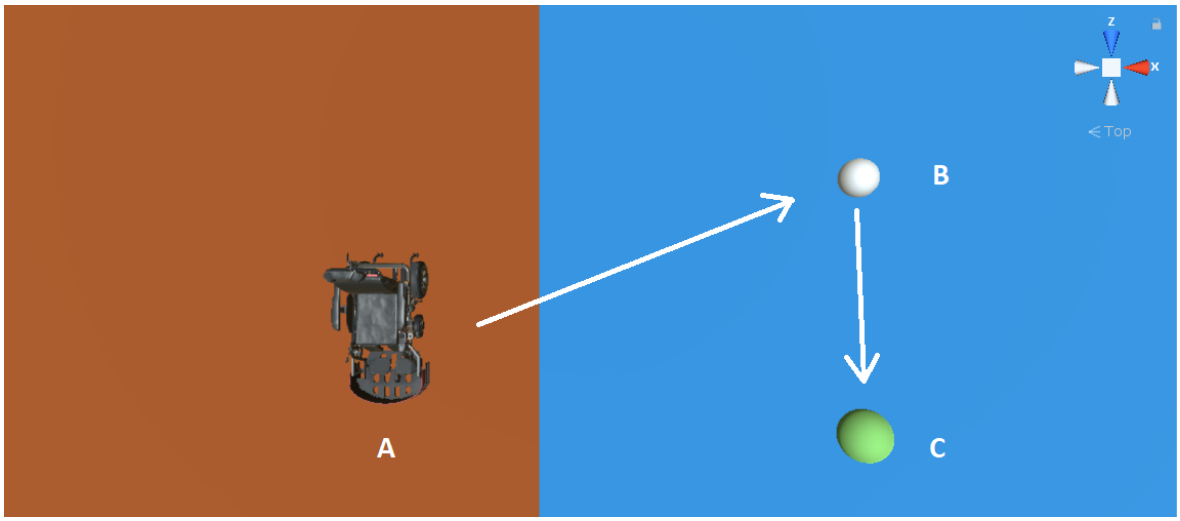


Figure 4.10: Dribble the ball random object spawn and path example

Point A represents the agents to be trained. Point B is the ball. Point C is the pointToGo where the ball must be taken. The arrows represent the path and direction an agent should follow in order to take the ball to the desired point.

(2) Every time that the agents touches the ball, he receives 0.01 reward values. These values are posterior added to the final reward. This was found to be good for accelerating the training session because encourages the agent to keep touching the ball and, eventually, forces him to dribble the ball to the desired pointToGo. If this was not used, the agent could, for instance, start learning to kick the ball to the pointToGo, which is not desired.

(3) If the ball is at a distance smaller than 0.5 from the pointToGo, then, the agent successfully took the ball to the pointToGo and recieves a reward of 10.

4.1.5.3 Results

The following diagram presents the results of the cumulative reward after 50 million steps of training. Once more, the algorithms used to achieve this results are based in the PPO, Behavioural

cloning and Gail. 20 episodes were recorded with the demonstration recorder.

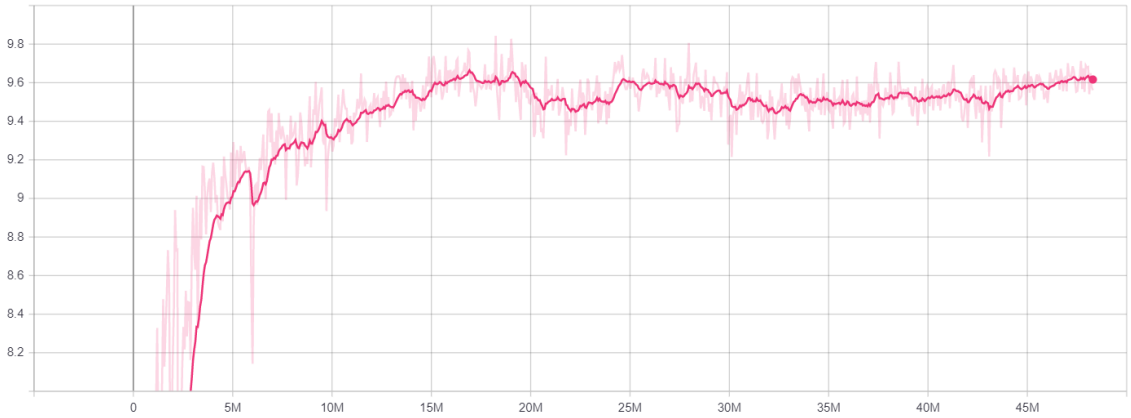


Figure 4.11: Dribble the ball behaviour Cumulative Reward

Although there exists some oscillation of values, which is normal due to the randomness and curiosity values used for this train, after 25 million steps, there exists a tendency of 9.6 reward cumulative values, therefore, we can conclude that it approximates the maximum reward possible (10), thus, the task was successfully learned.

4.1.6 Follow-a-point Behaviour

Follow-a-point Behaviour comes in need of the problem that is faced in the Higher Behaviour section. There exists the need of creating a system that allows player dynamic positioning, therefore, it is absolutely necessary to define a mechanism that orders agents to move to a desired point in the field. We will discuss this in detail further in this dissertation.

In order to face this, it was found a good policy to teach an agent to learn to go to any point in the field of play. This way, when coordinating the positioning system, it is possible to order an agent to move to the desired calculated point of the dynamic positioning system.

Although there exits some other good options to make an agent go to a desired point, for instance, by providing a force to it and the correct rotation angle, teaching an agent to go to a point with machine learning seems to be a better option because it will seem more natural than the first option. The first option will seem a robotic behaviour which is not well desired.

4.1.6.1 Environment Observations

This behaviour is very similar to the previously discussed Dribble the ball 4.1.5 since the objective is to move to a point. The difference is that the Dribble the ball aims to dribble the ball to the pointPointToGo while Follow-a-Point aims only to teach the agent to move to the pointPointToGo.

That said, the environment is composed by the field of play, the agent and the pointToGo.

The following table 4.6 provides the observation vector values in exception of the ray-cast, which are also used in this behaviour.

DistanceFromAgentToPoint	The distance from the agent to the point to go.

Table 4.6: Follow-a-point Behaviour observation vector values

DistanceFromAgentToPoint provides the distance that the agents is from the pointToGo. Once again, this helps the agents understand if he is closer or further away from the desired pointToGo.

4.1.6.2 Reward System

The reward system follows the same ideology than the previously discussed Dribble Behaviour 4.10. The following steps represent the reward system solution:

(1) First, the agent spawns in a random point inside the field of play. The pointToGo, which is the 3D Sphere Object that represents the point where the agents should learn to go while dribbling the ball, is then spawned in a random point inside the field of play as well.

(2) If the agent is at a distance smaller than 0.5 from the pointToGo, then, the agent successfully accomplished his task and its reward with 10.

4.1.6.3 Results

Due to the simplicity of this behaviour and non-complex environment, the results were very fast to be obtained. This can be observed at the following diagram 4.12. Gail, Behavioural Cloning and PPO algorithms were used as well. 10 episodes were recorded by the Demonstration Recorder Component.

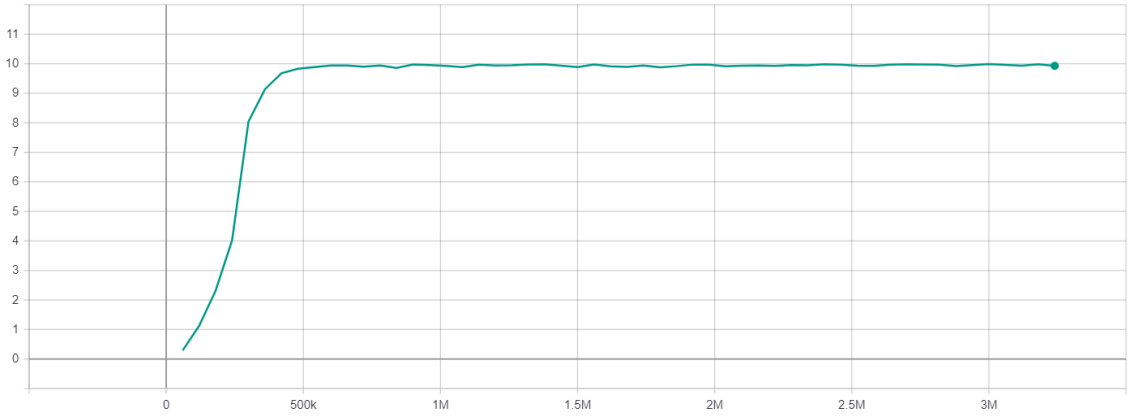


Figure 4.12: Follow-a-point behaviour Cumulative Reward Results

By observing the diagram, it is very clear that the agent learned very fast to move to the desired point. After 500k steps, which is equal as 25min of training, the cumulative reward almost converged to its maximum allowed reward. Furthermore, there was no need of smoothing the diagram, which had to be done in the previous behaviour results, to understand better his tendency.

4.2 Higher-Level Behaviour

Although we have seen in the previous sections how the agents behaviours were developed and proved that each one of them were able to learn the provided task, there's still the need of command when one should be or not enabled while the game is running.

Agents must know when is appropriate to make use of one behaviour. For instance, when referencing to the Pass the Ball Behaviour, it is not expected that one agent that is in front of an open goal area from the adversary team, to pass the ball to a teammate. It is expected, for example, to strike it to the goal.

Therefore, this sections aims to explain in detail how the behaviours were mapped to different in-game situations.

The following sections aim to explain the two main mechanisms of the Higher Behaviour: The Agents Behaviour handler and the Agents Positioning in Field.

The first one is responsible for handling each one of the behaviours an agent has available, in other words, it is responsible for enabling or disabling each and map them to different game situations.

The second one is responsible for the dynamic positioning of the players in field, more in concrete, responsible for providing the positions one agent must move to. Furthermore, it aims to explain how a path is provided to an agent that is with the ball and should use it to approximate to the desired goal.

4.2.1 Agents Behaviour Handler

This section aims to explain in detail how the in-game situations are mapped to the agent behaviours. More in concrete, this handles the information that is obtained from the **GameEnvironmentInfo** class and chooses the more adequate behaviour for the corresponding situation.

In order to achieve this, it is necessary to develop a mechanism that handles the different behaviors, this means, a mechanism that is responsible for enabling and disabling them, since they cannot be all active at once.

That said, the strategy adopted is the following. Each agent has attached to it a GameObject with all the corresponding behaviours as well as all the corresponding ray-cast sensors and observation vectors from the previous training sessions. This GameObjects are then, at the beginning of the game, disabled and, while the game is running and new game situations are being presented, there exists an algorithm that maps the in-game situations to the appropriate behaviour.

The following sequence diagram 4.13 helps to understand how this is handled in a high-level perspective.

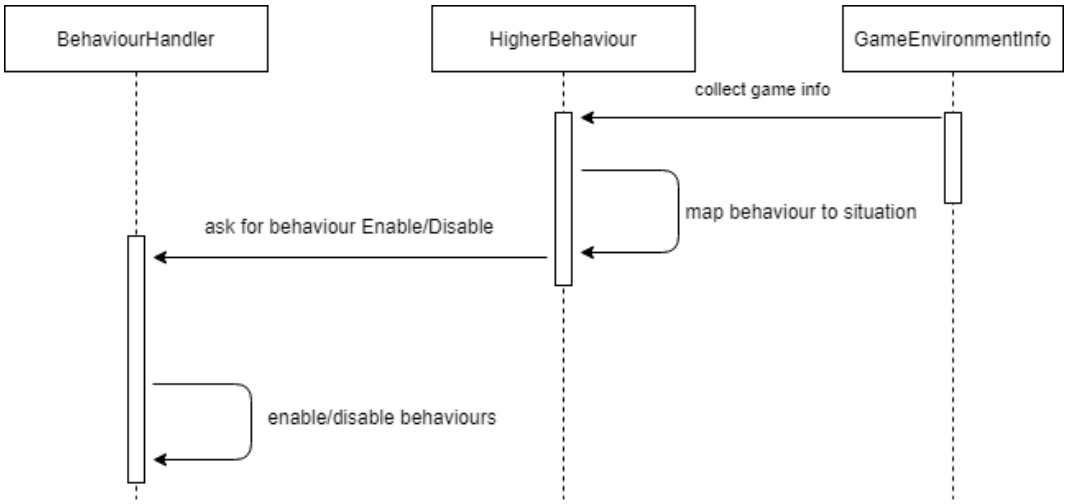


Figure 4.13: Higher Level Behaviour Sequence Diagram

By observing the diagram we can rapidly conclude that there exists three main classes responsible for the Higher Behaviour Handler.

GameEnvironmentInfo is constantly providing all the in-game information such as: a ball is out-of-bounds, a foul has been occurred and so on. This information is then sent to the **HigherBehaviour** Class.

HigherBehaviour is responsible for mapping the situations to the behaviours, in other words, this class chooses the most adequate behaviour for each agent in the field, at that moment, for that specific situation, thus, it handles the Mapping algorithm. It then asks **BehaviourHandler** to deal with the process of enabling or disabling the behaviour for that specific agent.

4.2.1.1 Mapping Algorithm

As referenced before, the mapping algorithm is mainly responsible for mapping in-game situations to behaviours.

At a certain point of the game, if an agent is using the dribble behaviour and he loses the ball, his behaviour should change, he must now, for instance, dispute the ball with the opponent, thus, enable the intersect the ball behaviour. This is exactly what the mapping algorithm aims to do, decide when behaviours should be enabled or disabled.

That said, lets start discussing how the mapping algorithm works. The following pseudo-code describe the approached solution.

Before the algorithm start, it is important to reference that the **nearestPlayer** and **nearestOpponent** are pre-defined, respectively, has the nearest player to the ball, thus, the player/agent with the ball in its possession, and the nearest opponent to the **nearestPlayer**. It is also important to reference that each agent has a pre-defined original position (**originalPos**)), which is the positions from the starting of the game of each agent in field. This positions are very important because they are the positions that are used to make the attraction and repulsion system. The following algorithm is running inside Update function of unity. That said, the algorithm runs as follows:

Algorithm 3 Agents Behaviour Handler - Mapping Algorithm

```
    for (i = 0; i ≤ redTeamAgents.Count; i++) do                                   ▷ (1)
 2:     if (redTeamAgents[i].isOutOfBounds()) then                                 ▷ (2)
            redTeamAgents[i].setFollowAPointBehaviour(originalPos)
 4:     else if (redTeamAgents[i].isGoalKeeper() ⋂ redTeamAgents[i] ≠ nearestPlayer) then ▷
    (3)
            if (redTeamAgents[i].isInOriginalPos()) then
 6:             if (checkGoalKickThreat()) then
                    redTeamAgents[i].setGoalKeepBehaviour()
 8:             else
                    redTeamAgents[i].disableAllBehaviours()
10:             end if
            else
12:             redTeamAgents[i].setFollowAPointBehaviour(originalPos)
            end if
14:     else if (redTeamAgents[i] == nearestPlayer) then                           ▷ (4)
            if (redTeamAgents[i].distanceToOpponentGoalArea() ≤ 3.5) then
16:             redTeamAgents[i].setStrikeTheBallBehaviour()
            else
18:             redTeamAgents[i].setDribbleTheBallBehaviour(getVoronoiPoint())
            end if
20:     else if (redTeamAgents[i] == nearestOpponent) then                         ▷ (5)
            redTeamAgents[i].setIntersectTheBallBehaviour()
22:     else                                                                       ▷ (6)
            redTeamAgents[i].setFollowAPointBehaviour(AttractionRepulsionNewPoint())
24:     end if
    end for
26:
    for (i = 0; i ≤ blueTeamAgents.Count; i++) do                                  ▷ (7)
28:     (...)
    end for
```

(1) The algorithm starts by iterating the whole list of the red team's agents.

(2) If the agent being analyzed is out of the boundary lines, which means he is not inside of the field, the agent's follow-a-point behaviour is activated for its original position and agent returns to its original position (**setFollowAPointBehaviour(originalPos)**). This is necessary because if an agent is outside of the bounds, he is doing nothing to help his teammates.

(3) If step 2 is not true, we check if the agent being analyzed is the red team's agent goalkeeper. We also check if he is not with the ball in its possession. If both of this are true, and if this agent, which is the red team's goalkeeper, is at his original position, then, we may now check if it is being threatened by a goal kick from an opponent, which is done by calling **checkGoalKickThreat()**. If the goalkeeper is being threatened, then, the agent's goal-keep behaviour is activated (**setGoalKeepBehaviour()**). If not, and since the agent is at his original position already, all agent's behaviour are disabled. If agent is not at his original position, agent's follow-a-point behaviour for its original position is activated (**setFollowAPointBehaviour(originalPos)**).

(4) If step 2 and 3 are not true, we check if the agent being analyzed is the **nearestPlayer**, thus, the player with the ball in his possession. If he is, then, it is necessary to understand if this agent is near or not from the oponnent's goal area, therefore, if the **nearestPlayer** is ≤ 3.5 from the oponents goal area, the agent's Strike-the-ball Behaviour is activated and the agents tries to kick the ball to score. If he is not, the agent's Dribble-the-Ball Behaviour is activated and the agents follows the point attributed by the Voronoi-based attacking mechanism, explained further in this dissertation.

(5) If step 2, 3 and 4 are not true, we check if the agent being analyzed is the **nearestOpponent**. If he is, agent's intersect-the-ball behavior is activated and agent tries to steal the ball to the **nearestPlayer**.

(6) If step 2, 3, 4 and 5 are not true, we are analyzing one of the remaining agents that need to re-position themselves in field in a way that they're positions helps they're team defending or attacking. Therefore, agent's being analyzed follow-a-point behaviour is activated with the corresponding Attraction and Repulsion Mechanism provided position.

(7) Steps 1 to 6 are done equally for the blue team's agents.

4.2.2 Agents Positioning in Field

This next section covers the problem of positioning players in field and decide when and where an agent should move if he has the ball in its possession.

We will discuss two mechanism that deal with this problem, namely, the Voronoi-based attacking mechanism, that explores the Delaunay Triangulation and Voronoi diagram as an answer

to the problem of providing a path from where an agent that is dribbling the ball should go, and the Attraction and Repulsion positioning mechanism that answers the problem of where the agents that don't have the ball should be in the field, in order to assist or pressure other players.

4.2.2.1 Voronoi-based attacking mechanism

In mathematics and computational geometry, the Delaunay triangulation (also known as a Delone triangulation) for a given set P of discrete points in a plane is a triangulation DT(P) such that no point in P is inside the circumcircle of any triangle in DT(P). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation. [16]

A Voronoi diagram is a partition of a plane into regions close to each of a given set of objects. In the simplest case, these objects are just finitely many points in the plane (called seeds, sites, or generators). For each seed there is a corresponding region consisting of all points of the plane closer to that seed than to any other. These regions are called Voronoi cells. Furthermore, the Voronoi diagram of a set of points is dual to its Delaunay triangulation. [9] [37] [62]

Supposing that the center of each of the polygons of a region is an agent in field, we can rapidly define an area that is suitable for the agent to understand if he is closer or further away from all other players. In other words, the bigger the area of the region of one agent, the more further away all the other agents in field are from him. Because of this information, Voronoi diagrams were chosen to deal with the problem of providing a path to the agent that has the ball in it's possession.

That said, in order to draw a Voronoi diagram, that has in the center of each polygon a predefined point that will be our agents in field, we must use the Delaunay triangulation first and then use a Voronoi algorithm to draw the Voronoi regions. For this, it was chosen to use the As3delaunay Library[58][43], which is an open-source ActionScript 3 library that provides Delaunay and Voronoi algorithms. In addition it also provides the convex hull, minimum and maximum spanning trees and several other related geometric entities based on Steven Fortune's C implementation of his sweepline algorithm[23].

The following figure 4.14 represents the Voronoi Diagram with the actual players in field when the game starts.

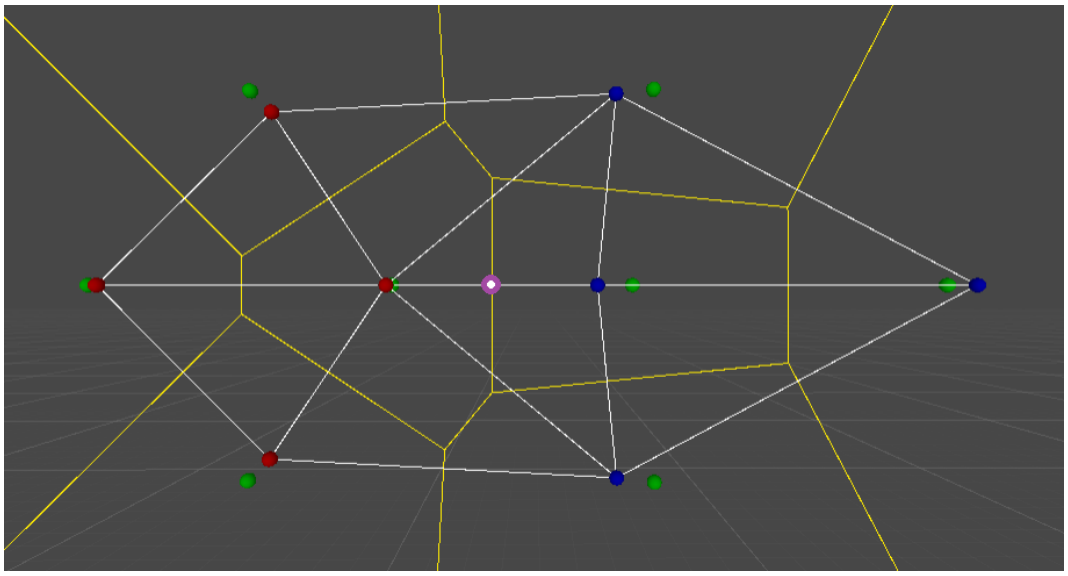


Figure 4.14: Voronoi diagram at the beginning of the game

The white lines represent the Delaunay triangulation edges and the yellow ones the Voronoi diagram edges of each region. Red points are red team's agents positions and blue ones are blue team's agents positions. A purple point with a white dot represents the ball position and green points are the center of mass of each region polygon.

As discussed previously at 4.1, agents learned to Dribble the ball to a desired point, therefore, we can take advantage of both Voronoi and Dribble-the-ball behaviour by providing the Voronoi edge we desire that an agent follows and activate the corresponding Follow-a-Point behaviour.

What is desired, since the agent with the ball will be constantly approached by opponents that want to steal the ball from it, is that the agent moves to a point where the probability of being intersected by an opponent is the lowest as possible, which, in fact, is possible due to the Voronoi regions.

By observing Voronoi region edges we may understand that some edges are points that, in fact, are the solution for this problem. Therefore, the agent with the ball should move to one of the edges of his corresponding Voronoi Region polygon, more in concrete, he should move to the edge that is further away from him and more closer to the adversary goal area. The following figure 4.15 provides an example where one agent can move for two possible points.

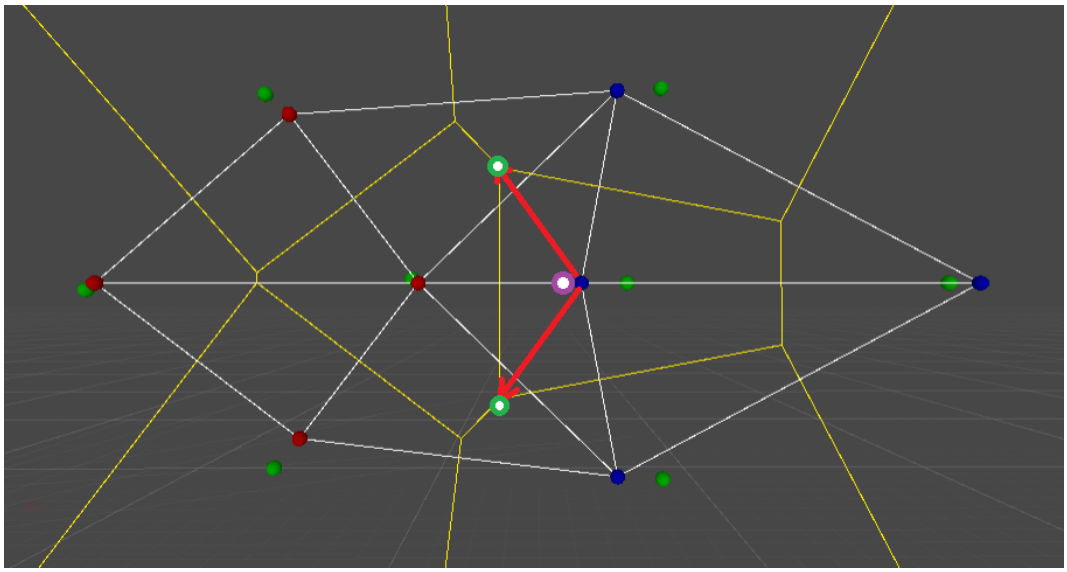


Figure 4.15: Example of the point an agent with the ball must dribble the ball to

The red arrows represent the possible linear path the agent should do, and the green white dots points are the new position chosen for the agent with the ball to go. There exists two possible solutions because the distances of both edges to the agent and the adversary goal area are the same. In this case, the algorithm that computes this decisions chooses randomly one. When there exists plenty edges and multiple directions, the algorithm takes in count the edge from the Voronoi region of the agent that is further away from him and that is closer to the adversary goal area, otherwise, the direction to go could be wrong.

4.2.2.2 Attraction and Repulsion positioning

Although the problem of providing a path to the agent with the ball is solved, there still exists the need of providing positions for the remaining agents in field that makes sense for the development of the game. This means that agents that are not with the ball should move to strategic positions where they can either help they're teammates or make pressure to opponents.

The attraction and repulsion positioning mechanism we will discuss in this section comes in response to this problem.

The solution proposed is to provide new positions an agent can move to by making use of the follow-a-point behaviour. This will be positions that are generated in function of the ball current position. More in concrete, in this scenario, the ball works just like an iman. If the ball moves to the left, the position of the agent moves to the left as well, the same way, if the ball moves to the right, the agent moves too. The following image 4.16 explains clearly how this mechanism should work.

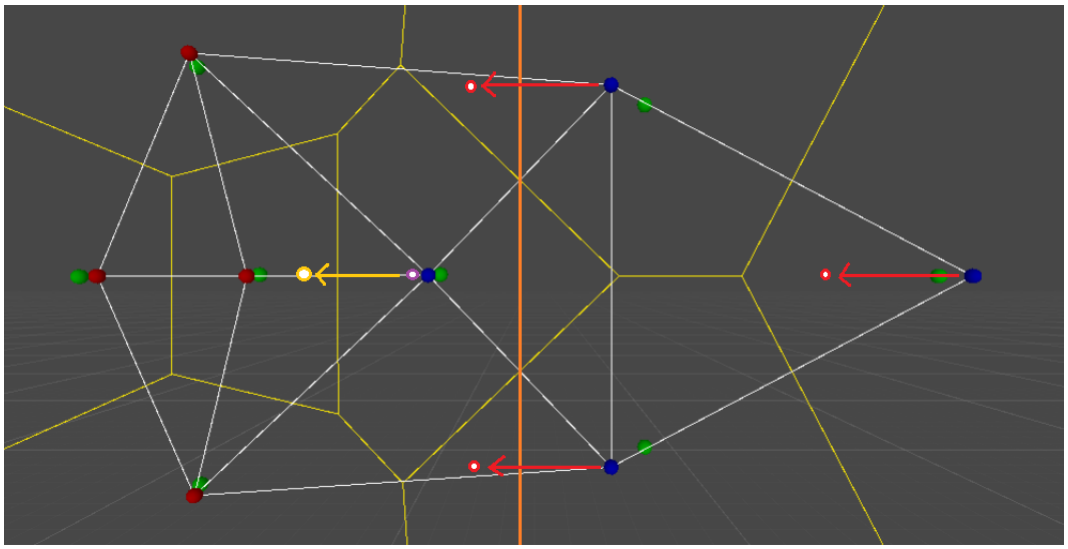


Figure 4.16: Attraction example of the blue team's agents

Just like other previous examples, red points are the red team's agents and blue points are the blue team's agents. The same applies for green dots, they represent the center of mass of each polygon, and the purple dot that represents the ball. The orange line represents the center boundary line of the field, thus, divides the field in two.

By observing this figure we can see an example of attraction. When blue agent moves the ball to the yellow point, represented by the yellow arrow, all other three agents of the blue team should move to the left equally to the red points, represented by the red arrows.

The same way, agents can be repulsed if they intend to defend. We can observe an example of repulsion in the following figure 4.17.

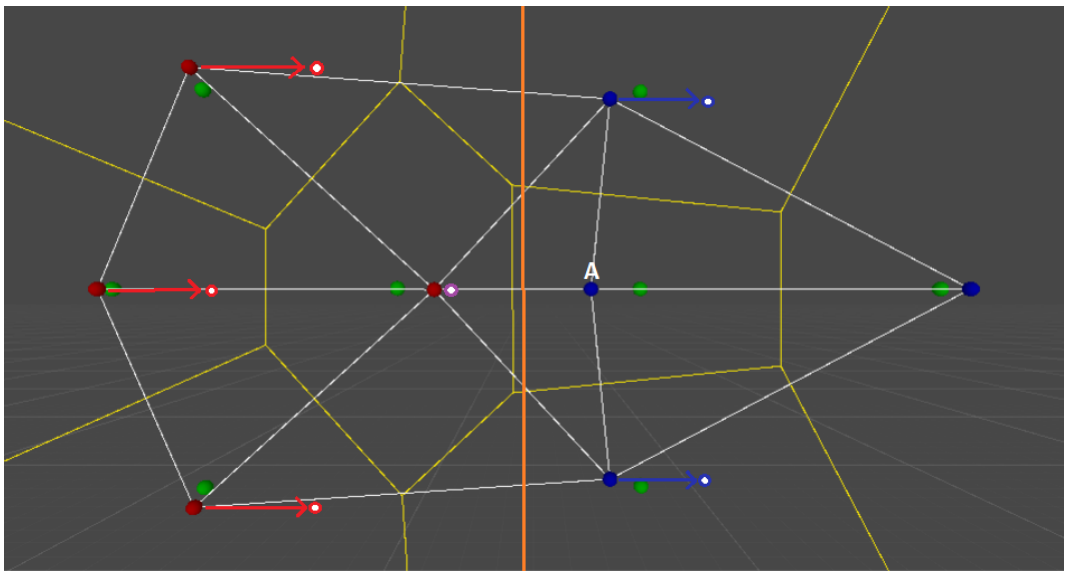


Figure 4.17: Repulsion example of the blue team's agents

When red team is attacking and the red team agent with the ball is going up in the field, blue team's agents are being repulsed by the ball, going backwards in their own area.

In order to achieve this effect of attraction an repulsion, it was used interpolation of vectors, more in concrete, used Vector3.lerp.

What Lerp does is finding a value that is some percentage between two given values. For example, we could linearly interpolate between the numbers 3 and 5 by 50% to get the number 4. This is because 4 is 50% of the way between 3 and 5.

Therefore, and since all the agents have their own initial positions that were called **originalPos** saved, the interpolation was done by making use the most of the **originalPos** of each agent and the ball current position.

Furthermore, and since the ball can be in any place and the agents too, some limits were bounded in order to not allow this mechanism to generate points outside of the field.

4.3 Summary

By the end of this chapter we are able to understand how the agents behaviours were created by dividing this problem in two sub-problems.

The first one explores machine learning areas, such as reinforcement learning and imitation learning to achieve desired low-level behaviours for the agents. It adopts strategies to increase the training sessions accuracy and decrease time consumption. We have seen how six different behaviours for powerchair football can be achieved by making use of Unity ML-Agents framework.

The second one explores the strategy adopted to centralize all this low-level behaviours that we call the High-level behaviour. This introduces a algorithm for mapping the low-level behaviours to in-game situations and defines when one of these six behaviours must be enabled or disabled depending on the current in-game situation. It also explain how players positioning along the field was achieved by making use of Delaunay and Voronoi Diagrams and attraction and repulsion mechanisms.

Chapter 5

Results and Discussion

In order to evaluate the quality of the game and understand if the goals for this dissertation have been achieved, it was chosen to conduct an experiment and use a survey to collect information. The experiment we propose is the following: A user should play, at least, a full match of the powerchair football in the IntellPowerSoccer. The link for the executable of the IntellPowerSoccer can be accessed by https://drive.google.com/file/d/1RNIyIeIg0-A6Ix3gn5-eu22cCyclhWod/view?usp=sharing. He should also memorize the number of goals he scored and suffered, and, finally, answer to the survey. This survey can be accessed by the following link https://forms.gle/h1DVRe8rwKNudBKe8.

This chapter is divided in five sections. The sample characterization, which describes succinctly the sample of individuals that were used to perform this tests. The User Experience with Video-games, that aims to understand the degree of ease a potential user has playing games. The Game Experience Questionnaire Results, which presents the results of a Research approved gaming satisfaction questionnaire. The Assessment of the IntellPowerSoccer Experience, responsible for evaluating personal opinions of the game after the user playing. And, finally, The User's Suggestions, which is an open field of the survey that allows users to make comments or critics to the game.

5.1 Sample Characterization

The experiments were performed with a sample of individuals that in its majority had no motor or cognitive difficulties, with a mean age of 27, a mode of 25 and a range between 20 and 49 years old. The sample is characterized by 21 males, 7 females and 2 individuals that choose not to identify their gender, thus, performing a total of 30 individuals. 90.1% has Portuguese citizenship, 3.3% Russian, 3.3% French and 3.3% Nigerian. Figure 5.1 presents these results.

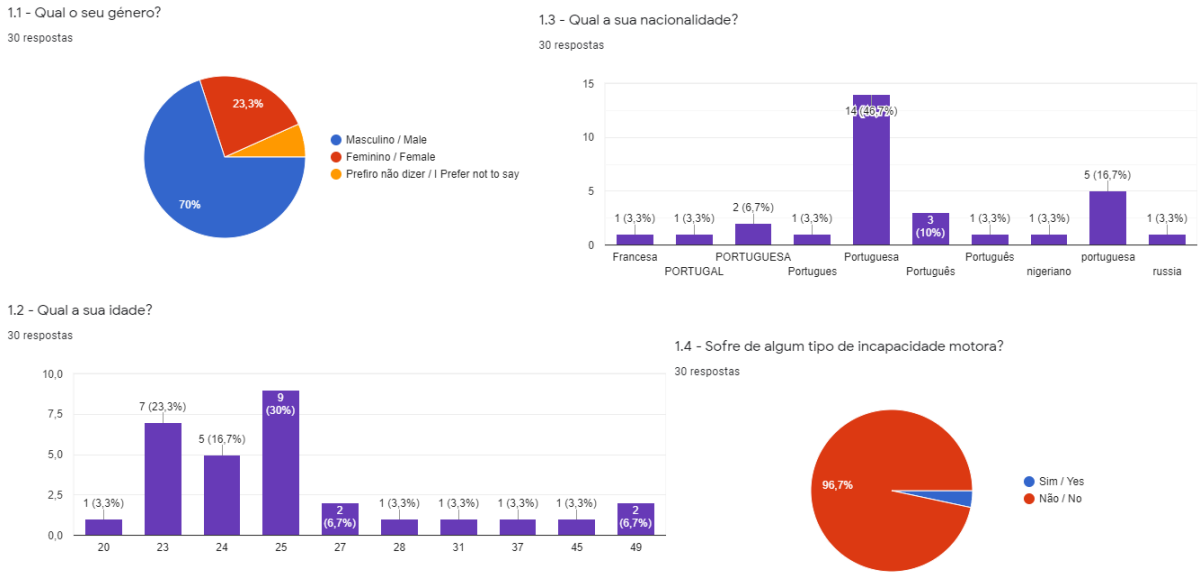


Figure 5.1: Socio-demographic Sample Data Results

In the survey, users were presented with a question and five possible answers that aimed to understand if a user suffered from a physical disability and if he regularly used, or not, a motorized wheelchair.

From a Likert scale, if a user either chooses Muita frequência / Very Frequently, Frequentemente / Frequently or Ocasionalmente / Occasionally, he was taken to a new section where he could evaluate the realism of the controller of the powered wheelchair in the IntellPowerSoccer.

This was chosen to be this way because acquiring this data it's a step to understand if the game simulates a real wheelchair or not, therefore, see if the objective of teaching a user to drive a powered wheelchair was accomplished. Figure 5.2 shows the results acquired.

By observing the following fig.5.2, from the 30 individuals, only one (3.3%) has presented cognitive difficulties and proclaimed to use a motorized wheelchair very frequently. In addiction, he answered that the controller was moderately realist.

Although this answer provides a good feedback for this dissertation goals, this is a small sample and it is not possible to conclude, with statistical evidence, the realism of the wheelchair controller in the IntellPowerSoccer.

Figure 5.2: Sample data results from the Controller realism field

5.2 User Experience with Video-Games

The following section evaluates the user experience playing video-games and his familiarization with the powerchair football modality. This section is important because it helps understanding better the data results obtained in further sections. It is a major key to subdivide the sample characterization in sub-types, for instance, divide the sample in people that play often and people who play rarely. This two are very likely to answer questions very differently since their game-play performance may be very different.

The following fig. 5.3 presents the obtained results from the frequency a user plays video-games question in the survey.

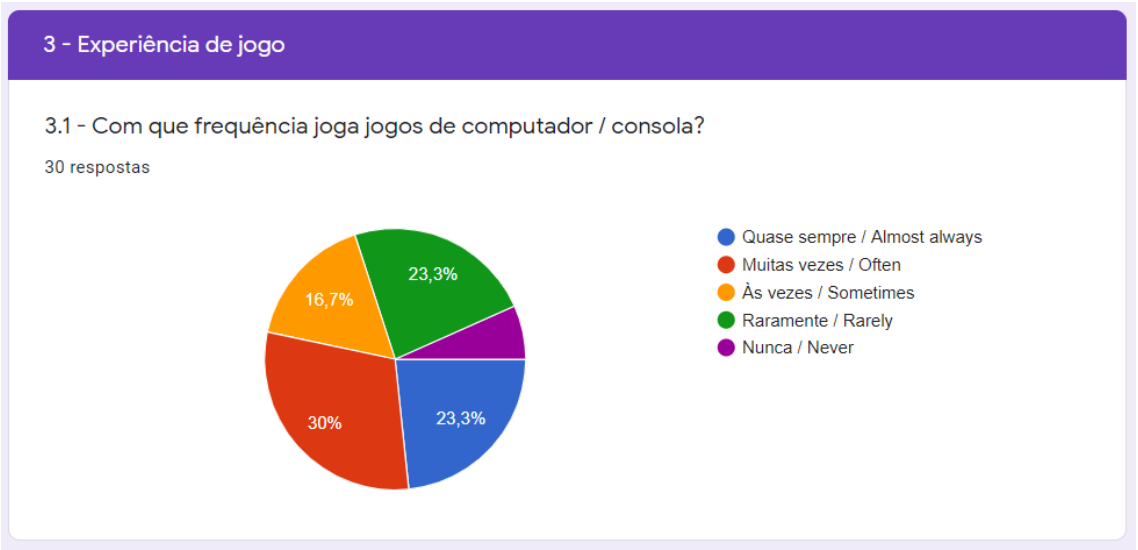


Figure 5.3: Results of the frequency a user plays video-games

By observing it, we rapidly conclude that 30% of our sample rarely or never plays video-games and 70% either sometimes, often and always plays video-game. That said, we have a sample of 30 persons where 9 are not regular video-game players and 21 are frequent video-game players.

The following fig. 5.4 presents the results for the question 3.2 that aims to understand if the users were familiarized with the modality of the powerchair football before playing the game.

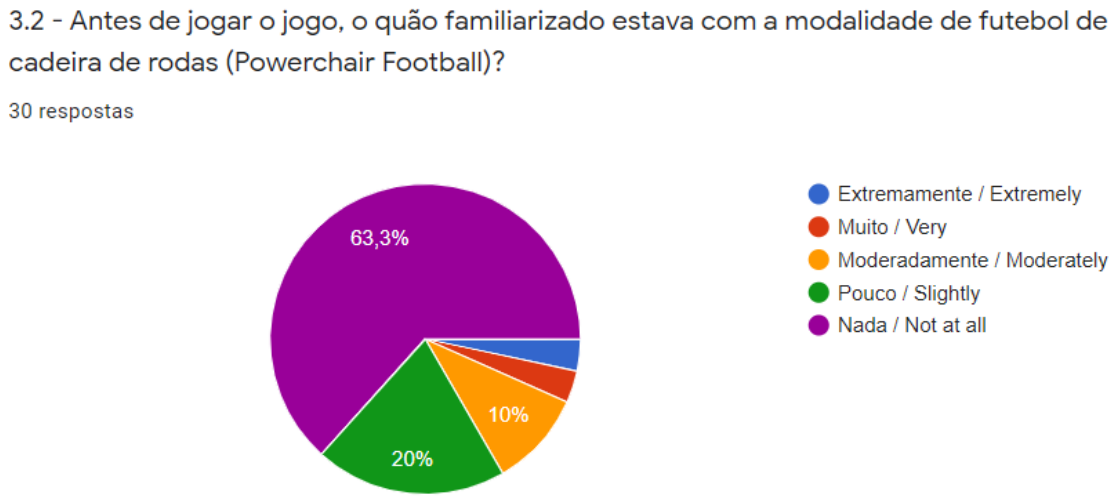


Figure 5.4: Results of the familiarization with the modality

The study suggests that the majority of the people were not familiarized with the game. 63.3% did not know anything about powerchair footbal, 20% slightly, 10% moderately, 3.3% very and 3.3% extremely.

5.3 Game Experience Questionnaire Results

This section integrates an already developed investigation tool based in the survey of the Game Experience Questionnaire, the CEGEQ, or in other words, the Core Elements of the Gaming Experience Questionnaire[30].

This was considered as an important piece of the developed survey because it provides powerful information that helps understanding the global experience users had, either while of after, playing the IntellPowerSoccer.

Although The Game Experience Questionnaire is composed by three main categories, being the first, the evaluation of the user experience while playing the game, the second, the user experience after playing the game, and the third, the social experience, since the IntellPowerSoccer does not have a multiplayer option, the third category was discarded.

Before providing the results, it is important to reference that the CEGEQ makes his evaluation based in the average score obtained by the answers of the survey. This score works as follows:

(1) The first group, relative to the User Experience while playing the game, is composed by 14 obligatory options, each one has a scale that varies between 0 (not at all) and 4 (extremely), which make the score. Therefore, after collecting all data, the 14 options are categorized and an average score of each category and corresponding item is calculated. The following table 5.1 presents this.

Competence	Items 2 and 9.
Sensory and Imaginative Immersion	Items 1 and 4.
Flow	Items 5 and 10.
Tension	Items 6 and 8.
Challenge	Items 12 and 13.
Negative affect	Items 3 and 7.
Positive affect	Items 11 and 14

Table 5.1: CEGEQ in-game scale categories

(2) The same way, the second group, relative to the User Experience after playing the game, is composed by 17 obligatory options and categorized as follows:

Positive Experience	Items 1, 5, 7, 8, 12, 16.
Negative experience	2, 4, 6, 11, 14, 15.
Tiredness	Items 10, 13.
Returning to Reality	Items 3, 9, and 17.

Table 5.2: CEGEQ post-game scale categories

After this, we are now able to understand the results obtained, therefore, the following table 5.3 provides the scores obtained for the first group that evaluates the user experience while one is playing the IntellPowerSoccer.

Category	Obtained Score	Percentage
Competence	2,4 points	60%
Sensory and Imaginative Immersion	2,58 points.	65%
Flow	2,35 points.	59%
Tension	1 point.	60%
Challenge	2,67 points.	68%
Negative affect	0,85 points.	21%
Positive affect	2,65 points.	60%

Table 5.3: CEGEQ in-game scale categories results

By observing this, we can conclude that the results are satisfactory. From a scale of 0 to 4, being 0 the minimum and 4 the maximum, the in-game results provide good results. 60% of positive effect indicates that many users found the game enjoyable while playing it. Sensory and imaginative immersion, with a score of 2.58, and Flow, with a score of 2.35, indicates that the users found the game immersive, thus, it is a great indicator of enjoyment as well. Tension, which might be seen and a negative effect field for this kind of video-game, since this is not an horror/terror video-game, has a low score of 1, which is also a great result. The fact that the users found the game challenging, with a score of 2.67, is a very good indicator that the game was actually challenging. Furthermore, the fact that the score did not crossed the limit of 3 tell us that the game did not became annoying for being too much challenging. Finally, with a score of 0.85, the negative effect, which is, in fact, a field that works in opposition of the positive effect, thus having a lower the score means higher goodness, is very low, therefore, a good result as well.

The following table 5.4 presents the group 2 results.

Category	Obtained Score	Percentage
Positive affect	1,83 points.	46%
Negative affect	0,43 points.	11%
Tiredness	0,42 points.	10%
Returning to Reality	0.73 points.	18%

Table 5.4: CEGEQ post-game scale categories results

Relating the user experience after playing the game, the negative effect and the tiredness, which are both negative effect fields, are marked very low, 11% and 10%, respectively, therefore, both indicate that the user did not found the game annoying neither tiring.

On other hand, the positive effect results, does not seem to be satisfactory as desired, they marked 1.83 points (46%). Although this seems to be bad, this category covers many items of the survey and, from this items, the ones that had a lower average and contributed the most to significantly decrease this average score were the items 7 and 12. This items asked the user if he felt energised and if felt powerful after playing the game. Since both of this answers are

great related to users performance while playing the game, and the percentage of users that play games sometimes/rarely/never is 47%, which is very high for this sample, it is expected that the performance of these individuals is very low, thus, the low score given to items 7 and 12 makes sense. Furthermore, this might indicate that this user's sample have found the gameplay hard.

On a summary, although the results are satisfactory, they also indicate that users that are not used to play video-games find the game challenging and probably hard to play. Therefore, this might indicate that the game needs a tutorial section or practise mode that teaches users the basics.

5.4 Assessment of the IntellPowerSoccer

The following sections aims to describe ans discuss the results collected from the Assessment of the IntellPowerSoccer group of the survey.

While the CEGEQ evaluates the user's experience globally, this field expects to evaluate the user experience from a developer perspective, providing more concrete information on what user's like or hate the most about the game, about agents efficiency and controller preference, thus, providing great information for improvements and future work.

That said, this group starts by asking how many matches one has played before asking the survey and, followed by this, asked the number of goals one has scored and suffered in the last game. The following figure presents the results obtained.

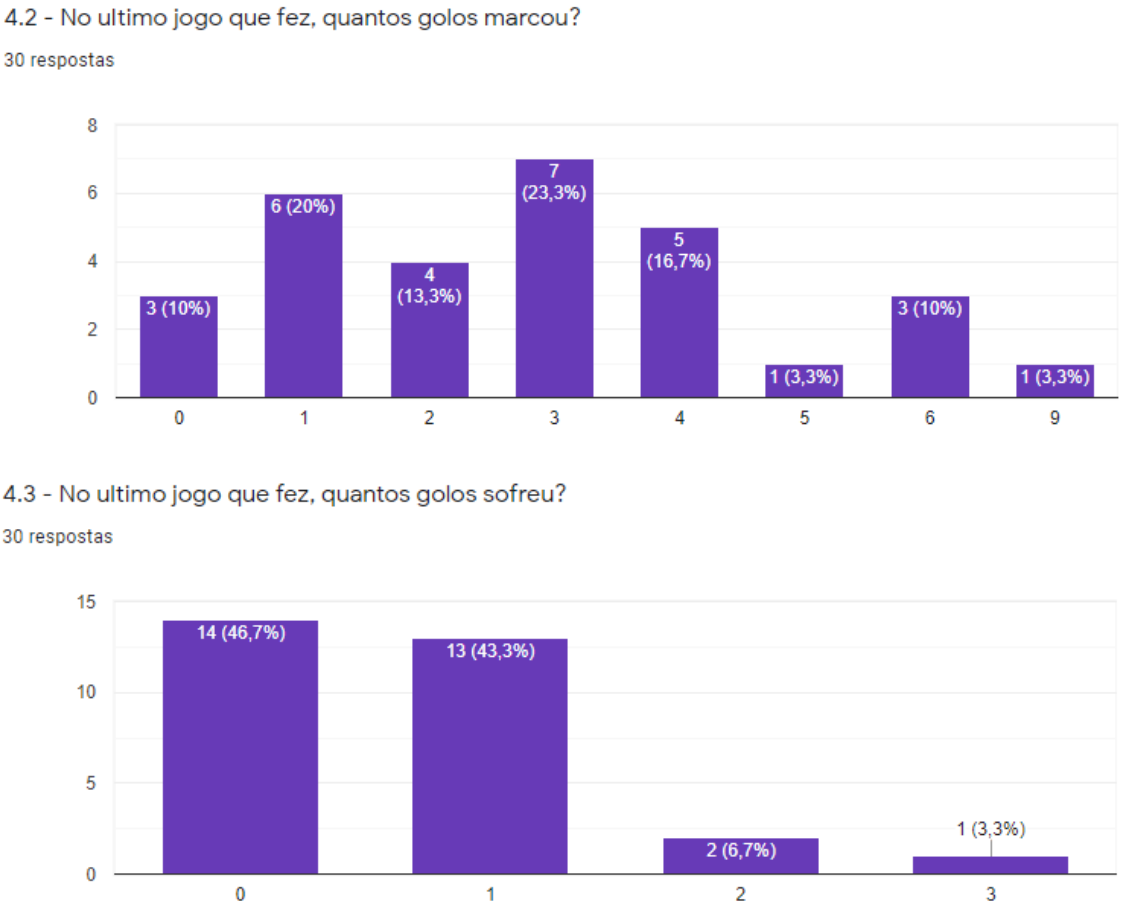


Figure 5.5: Results obtained from the Number of goals scored and suffered

Figure 5.5 represents the scored and suffered goals from the last match user's made. By observing it, although, in average, majority of the sample have scored 2.9 goals with a mode of 3 and suffered 0.67 with a mode of 0, it was found that, in average, players that never play video-games, scored 0.5 goal and suffered 1, player who rarely play games scored 2 goals and suffered 0.85, players who play sometimes scored 3 goals and suffered 0.6, players who play often scored 3.1 goals and suffered 0.66 and, finally, players who always play video-games scored an average of 4.1 goals and suffered 0.42. This parameter observations clearly shows that users with more experience were more well-succeeded than players who rarely or never play video-games, which, in fact, may justify some chosen parameters in the Game Experience Questionnaire we discussed in the previous section. It's more likely that a player that was not well-succeeded to be more frustrated or tired with the game experience as well as more willing to not feel powerful or energised.

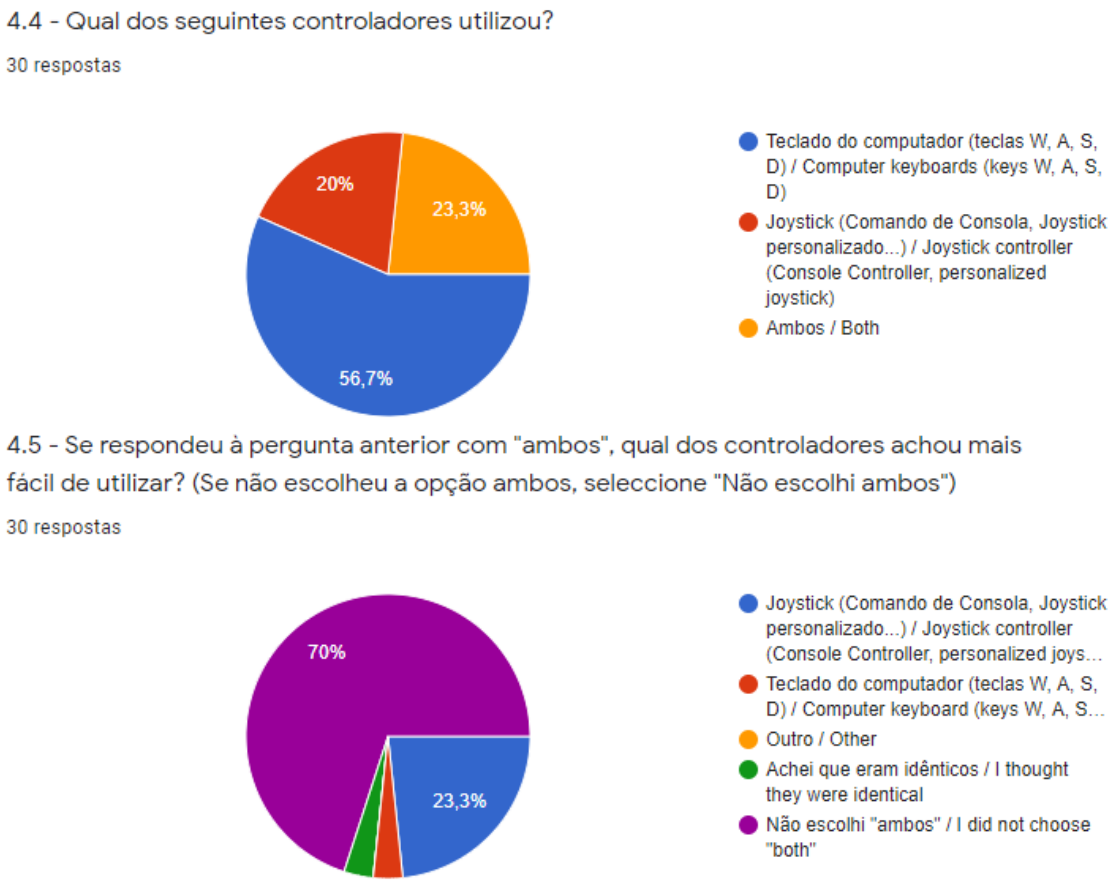


Figure 5.6: Controller Preference results

Figure 5.6 shows the results obtained from the the preference of controller for manoeuvring the wheelchair while playing the game. We can clearly understand that the majority of people that played with both controllers (W,A,S,D Keyboard keys, or console joystick) preferred to use joystick rather than keyboard. Even though this is a fact, it is quite understandable why majority choose the joystick and we will see why in the next section 5.6.

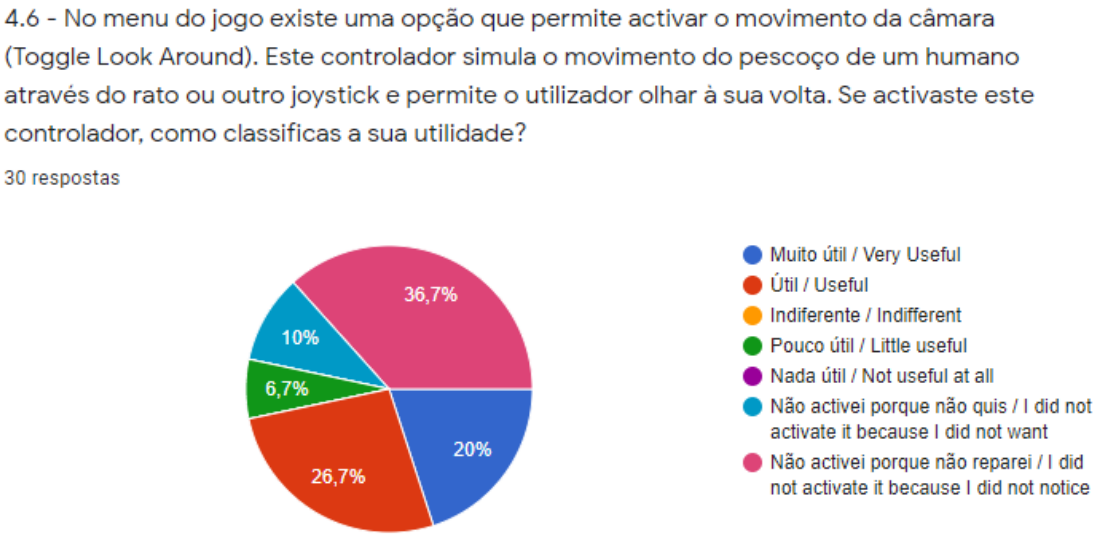


Figure 5.7: Toggle Look-around usefulness results

As referenced at 3.3, it was chosen to provide an option to enable/disable the rotation of the camera that simulates the movement of the neck of a person, while riding the wheelchair. That said, it was decided to ask if the users found the look around useful or not. The results suggests that many players were not able to find the look around button, which may indicate that the positioning of this button in the user interface is not intuitive. Even though, the majority of users that found it and enabled it, found it useful and very useful. We can observe this data possible at figure 5.7.

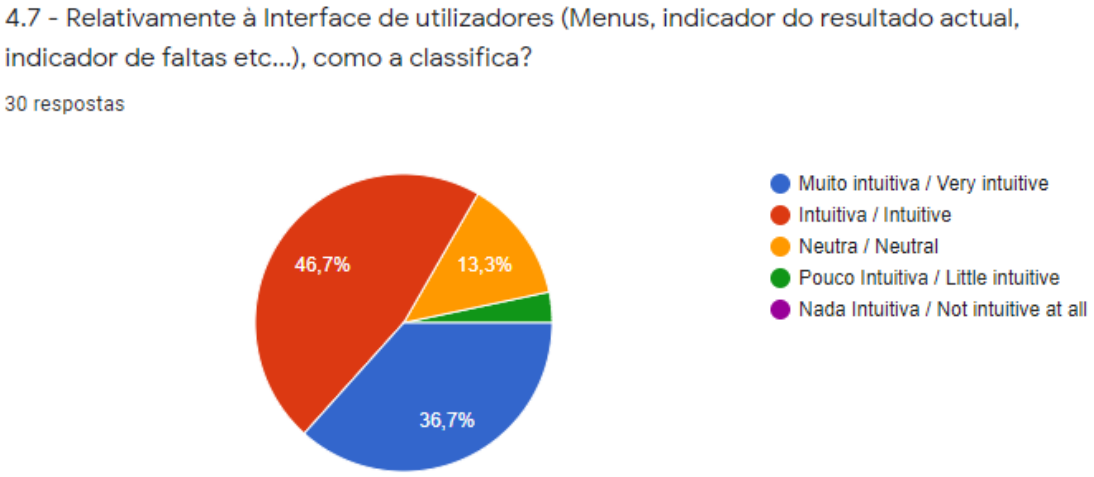


Figure 5.8: User-Interface Intuitiveness results

By observing figure 5.9, we can clearly understand that majority of the sample found the user interface intuitive. Only 1 person found it little intuitive and 4 neutral, therefore, 83.4% of the sample found it intuitive or very intuitive.

Figure 5.9: Agents efficiency results

Finally, and for closing this group, the results relative to the efficiency of the agents in field are quite satisfactory. 10% of the sample found the agents very efficient, 56.7% found it efficient and only 16.7% found it neutral. Only 16.6% reported negative efficiency for the agents behaviour, which tell us that the user interface is very intuitive in general.

5.5 User's Suggestions

The following section presents some critics and opinions from individuals that answer the survey.

Some user's have proclaimed not being able to press the "W" Keyboard key and "A" or "D" at the same time for moving the powered wheelchair frontwards and sideways at the same time. This is a fact well observed because the wheelchair controller in-game is completely optimized for the joystick and not for the keyboard. This was chosen to be this way because one of the goals of this dissertation is to simulate the conduction of the powered wheelchair, therefore, the joystick mapping works just like a real powered wheelchair. The ability of using the keyboard is just an add-on made thinking about users that don't have a joystick to be able to play the game .

One user has reported that the goal areas should be more marked with the corresponding color of the team. This suggests that users may present some difficulties in trying to find the goal area where they should score, therefore, this is something that must be approached as future work improvement.

Some Users have suggested to allow players to play against each other, thus, allow multiplayer, and create tournaments in-game. Furthermore, one individual referenced that it would be very interesting to have a mini-map in the game in order one can easily see where his teammates and opponents are in the field. This is also another great idea that can be implemented at future work.

One person has also referenced that the game could have a difficulty system for the agents to be more or less competitive. Lastly, some people have declared to have found bugs. Some of them were posteriorly corrected.

In a summary, the comments of the game were very constructive and, in it's majority, provide good tips for future improvement of the game. The next chapter analysis this with more detail.

5.6 Summary

By the end of this chapter, and by analysing the data provided by the survey, we can conclude the results have been quite satisfactory.

The agents were proven to be efficient. Even though, some users suggested that the behaviour of these agents could be improved. We will discuss this in the next chapter.

Users have also criticized that they where not being able to press the "W" Keyboard key and "A" or "D" at the same time for moving the powered wheelchair frontwards and sideways at the same time. This problem will be discussed in the next chapter.

Furthermore, and because our sample had insufficient individuals that used powered wheelchairs frequently, we could not prove the approximation to reality for the controller of the wheelchair in the IntellPowerSoccer, which was important to understand if the wheelchair manoeuvring was similar to a real powered wheelchair and fulfill the secondary objectives of this dissertation, to teach users how to manoeuvre a powered wheelchair. This problem will also be discussed in the next chapter.

In conclusion, we have seen that the game was proven to be fun, with good flow, immersive and, globally, it received positive feedback, therefore, we can say that the IntellPowerSoccer has accomplished its major goal for this dissertation, to develop a serious game that has some simulation properties, it's immersive, enjoyable and efficient.

Chapter 6

Conclusion and Future Work

By the end of this dissertation, we have explored realism to achieve one of the main goals of teaching users how to manoeuvre a powered wheelchair. We have been presented with a strategy that allows the in-game wheelchairs to rotate freely with they're own caster wheels, presenting a natural movement and realism. In addiction, all the models used for the wheelchairs, ball and field were presented and made at real scale, which was found necessary. We have seen all the algorithms used for developing the logic of the game, such as the 3-in-the-goal area foul mechanism, the two-on-one foul mechanism, the ball out-of-bounds mechanism and the player tracking system. For the same purpose, the controller was implemented based in a real powerchair joystick, having real mappings and gaps. A User Interface that was proven to be very Intuitive, was also explained and implemented.

Since this dissertation explores machine learning and, in average, majority of video-game requires to have agents that play against the user, a multi-agents system based in Machine learning was also developed. We have discussed the low-level behaviours, that were achieved by exploring multiple reinforcement learning strategies and complemented by training the agents with imitation learning algorithms, as a manner of speeding up the process of learning. Furthermore, we have seen a strategy adopted to centralize all this low-level behaviours that we call the High-level behaviour. This introduces a algorithm for mapping the low-level behaviours to in-game situations and defines when one of these six must be enabled or disabled, depending on the current in-game situation. It also explain how players positioning along the field was achieved by making use of Delaunay and Voronoi Diagrams and attraction and repulsion mechanisms.

As discussed before, by analysing the data provided by the survey, this agents were proven to be efficient. Even though, some users suggested that the behaviour of these agents could be improved, they have criticized that, sometimes, more than 4 agents that have collide to each other became stuck and did not know how to get away of that situation. We have seen in previous chapters that one of the major problems of the machine learning is not being able to adapt to new situations, therefore, as a future work it would be great to improve the agent's behaviour in terms

of adapting to it, for instance, make new training sessions where agents have multiple team players or adversary's doing unexpected things.

Users have also criticized that they where not being able to press the "W" Keyboard key and "A" or "D" at the same time for moving the powered wheelchair frontwards and sideways at the same time. This is a fact well observed because the wheelchair controller in-game is completely optimized for the joystick and not for the keyboard. This was chosen to be this way because one of the goals of this dissertation is to simulate the conduction of the powered wheelchair, therefore, the joystick mapping works just like a real powered wheelchair. The ability of using the keyboard was just an add-on made thinking about users that don't have a joystick to be able to play the game. Either way, for future work, it would be a great idea to create a new controller mapping for the keyboard.

Regarding objectives, because our sample had insufficient individuals that used powered wheelchairs frequently, we could not prove the approximation to reality for the controller of the wheelchair in the IntellPowerSoccer, which was important to understand if the wheelchair manoeuvring was similar to a real powered wheelchair and fulfill the secondary objectives of this dissertation, to teach users how to manoeuvre a powered wheelchair. Therefore, for future work, it would be very good if a new experience was conducted and presented to institutions such as the Portuguese Cerebral Palsy association or other that work with people with motor-disabilities and collect new data more appropriate to evaluate this correctly.

Nevertheless, we have seen that the game was proven to be fun, with good flow, immersive and globally it received positive feedback, and therefore, we can say that the IntellPowerSoccer has accomplished its major goal for this dissertation, to develop a serious game that has some simulation properties, it is immersive, enjoyable and efficient.

References

[1] Openai gym docs. https://gym.openai.com/docs/, February 2020. [Online; accessed at 09-February-2020].

[2] Robert G. Abbott. Behavioral cloning for simulator validation. In Ubbo Visser, Fernando Ribeiro, Takeshi Ohashi, and Frank Dellaert, editors, *RoboCup 2007: Robot Soccer World Cup XI*, pages 329–336, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] Patrick Abellard, Iadaloharivola Randria, Alexandre Abellard, Mohamed Moncef Khelifa, and Pascal Ramanantsizehena. *Electric Wheelchair Navigation Simulators: why, when, how?* 03 2010.

[4] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[5] Rodrigo AM Braga, Pedro Malheiro, and Luis Paulo Reis. Development of a realistic simulator for robotic intelligent wheelchairs in a hospital environment. In *Robot Soccer World Cup*, pages 23–34. Springer, 2009.

[6] Rodrigo AM Braga, Marcelo Petry, Antonio Paulo Moreira, and Luis Paulo Reis. Intellwheels-a development platform for intelligent wheelchairs for disabled people. In *ICINCO 2008: PROCEEDINGS OF THE FIFTH INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATION AND ROBOTICS, VOL RA-2: ROBOTICS AND AUTOMATION, VOL 2*, 2008.

[7] Rodrigo AM Braga, Marcelo Petry, Antonio Paulo Moreira, and Luis Paulo Reis. Concept and design of the intellwheels platform for developing intelligent wheelchairs. In *Informatics in control, automation and robotics*, pages 191–203. Springer, 2009.

[8] Rodrigo Antonio Marques Braga, Marcelo Petry BEng, Luís Paulo Reis, and Antonio Paulo Moreira. Intellwheels: Modular development platform for intelligent wheelchairs. *Journal of Rehabilitation Research & Development*, 48(9), 2011.

[9] P. A. Burrough. *Principles of geographical information systems*. Oxford University Press, Oxford New York, 2015.

[10] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, April 2007.

[11] Félix Chénier, Pascal Bigras, and Rachid Aissaoui. An orientation estimator for the wheelchair's caster wheels. *IEEE Transactions on Control Systems Technology*, 19:1317–1326, 2011.

[12] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, New York, NY, USA, 2007. Association for Computing Machinery.

[13] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 144–151, New York, NY, USA, 2008. Association for Computing Machinery.

[14] Darlene Connoly. A brief overview of imitation learning. https://medium.com/@zoltan.lorincz95/a-brief-overview-of-imitation-learning-25b2a1883f4b, September 2019. [Online; accessed at 07-February-2020].

[15] Deepai. What is the markov decision process? https://deepai.org/machine-learning-glossary-and-terms/markov-decision-process. [Online; accessed at 10-February-2020].

[16] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[17] Brígida Mónica Faria, Luís Paulo Reis, and Nuno Lau. A survey on intelligent wheelchair prototypes and simulators. In *New Perspectives in Information Systems and Technologies, Volume 1*, pages 545–557. Springer, 2014.

[18] Brígida Mónica Faria, Luis Paulo Reis, and Nuno Lau. Adapted control methods for cerebral palsy users of an intelligent wheelchair. *Journal of Intelligent & Robotic Systems*, 77(2):299–312, 2015.

[19] Brígida Faria, Luís Ferreira, Luís Reis, Nuno Lau, Marcelo Petry, and João Couto Soares. Manual control for driving an intelligent wheelchair: A comparative study of joystick mapping methods. 10 2012.

[20] Brígida Faria, Luís Reis, and Nuno Lau. A survey on intelligent wheelchair prototypes and simulators. *Advances in Intelligent Systems and Computing*, 275:545–557, 01 2014.

[21] FIPFA. Powerchair football - Laws of the game. https://fipfa.org/wp-content/uploads/2017/08/FIPFA-Laws-of-the-Game-Approved-December-2010.pdf, 10 2010. [Online; accessed 24-January-2020].

[22] FIPFA. The History of the Development of Powerchair Football. https://fipfa.org/histoire-du-developpement-du-powerchair-football/, 2020. [Online; accessed 24-January-2020].

[23] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2(1-4):153, 1987.

[24] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

[25] Yang Gao, Jan Peters, Antonios Tsourdos, Shao Zhifei, and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 2012.

[26] Dave Gershgorn. New 'openai' artificial intelligence group formed by elon musk, peter thiel, and more. https://www.popsci.com/new-openai-artificial-intelligence-group-formed-by-elon-musk-peter-thiel-an December 2015. [Online; accessed at 09-February-2020].

[27] Google. Tensorflow: Open source machine learning. https://www.youtube.com/watch?v=oZikw5k_2FM, November 2015. [Online; accessed at 09-February-2020].

[28] Sally Hartley. World report on disability (who), 07 2011.

[29] Kevin Hernandez-Ossa, Berthil Longo, Eduardo Montenegro Couto, Maria Romero Laiseca, Anselmo Frizera, and Teodiano Bastos. Desenvolvimento de um sistema de realidade virtual para treinamento de uso de cadeira de rodas motorizada. 10 2017.

[30] W.A. IJsselsteijn, Y.A.W. de Kort, and K. Poels. *The Game Experience Questionnaire*. Technische Universiteit Eindhoven, 2013.

[31] Emma Brunskill James Harrison. Cs234 notes - lecture 7 imitation learning. In *Stanford University: Reinforcement Learning course*, pages 305–313, 1989.

[32] Michael Jeffress. *Communication, Sport and Disability: The Case of Power Soccer*. 05 2015.

[33] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

[34] R. E. Kalman. When Is a Linear Control System Optimal? *Journal of Basic Engineering*, 86(1):51–60, 03 1964.

[35] Fedwa Laamarti, Mohamad Eid, and Abdulmotaleb El Saddik. An overview of serious games. *International Journal of Computer Games Technology*, 2014, 10 2014.

[36] Max Lewontin. Open ai: Effort to democratize artificial intelligence research. https://www.csmonitor.com/Technology/2015/1214/Open-AI-Effort-to-democratize-artificial-intelligence-research, December 2015. [Online; accessed at 09-February-2020].

[37] Paul Longley. *Geographical information systems and science*. Wiley, Chichester, 2005.

[38] David Michael and Sandra Chen. Serious games: Games that educate, train, and inform. 01 2006.

[39] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. Foundations of machine learning. 2018.

[40] Brígida Mónica Faria, Sérgio Vasconcelos, Luís Paulo Reis, and Nuno Lau. Evaluation of distinct input methods of an intelligent wheelchair in simulated and real environments: a performance and usability study. *Assistive Technology*, 25(2):88–98, 2013.

[41] Y. Morère, G. Bourhis, K. Cosnuau, G. Guilmois, E. Blangy, and É. Rumilly. View, a wheelchair simulator for driving analysis. In *2015 International Conference on Virtual Rehabilitation (ICVR)*, pages 100–105, June 2015.

[42] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[43] Atsuyuki Okabe. Spatial tessellations. *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*, pages 1–11, 2016.

[44] openai. Proximal policy optimization - spinning doc. https://spinningup.openai.com/en/latest/algorithms/ppo.html#background. [Online; accessed at 10-February-2020].

[45] Marcelo Petry, A. Moreira, Brígida Faria, and Luís Reis. Intellwheels: Intelligent wheelchair with user-centered design. pages 392–396, 10 2013.

[46] Marcelo R Petry, Antonio Paulo Moreira, Brigida Monica Faria, and Luis Paulo Reis. Intellwheels: intelligent wheelchair with user-centered design. In *2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013)*, pages 414–418. IEEE, 2013.

[47] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989.

[48] CN Pronk, PC de Klerk, A Schouten, JL Grashuis, R Niesing, and BD Bangma. Electric wheelchair simulator as a man-machine system. *Scandinavian journal of rehabilitation medicine*, 12(3):129—135, 1980.

[49] Alain Pruski, Mourad Ennaji, and Yann Morere. Vahm: a user adapted intelligent wheelchair. volume 2, pages 784 – 789 vol.2, 02 2002.

[50] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[51] Luis Paulo Reis, Rodrigo AM Braga, Márcio Sousa, and Antonio Paulo Moreira. Intellwheels mmi: A flexible interface for an intelligent wheelchair. In *Robot Soccer World Cup*, pages 296–307. Springer, 2009.

[52] R/GA. VR Power Trainer. https://www.rga.com/work/case-studies/vr-power-trainer. [Online; accessed 29-January-2020].

[53] Stephane Ross and Drew Bagnell. Efficient reductions for imitation learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 661–668, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[54] John Rust et al. Structural estimation of markov decision processes. *Handbook of econometrics*, 4(4):3081–3143, 1994.

[55] Caude Sammut. *Behavioral Cloning*, pages 93–97. Springer US, Boston, MA, 2010.

[56] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] Alan Shaw. Open-source actionscript 3 library. http://nodename.github.io/as3delaunay/. [Online; accessed at 15-September-2020].

[59] Cuihua Shen, Hua Wang, and Ute Ritterfeld. *Serious games and seriously fun games: Can they be one and the same?*, pages 48–62. 01 2009.

[60] statslab.cam. Markov chain. http://www.statslab.cam.ac.uk/~james/Markov/. [Online; accessed at 10-February-2020].

[61] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[62] Zekai Sen. Spatial modeling principles in earth sciences. 2016.

[63] Christian Thurau, Gerhard Sagerer, and Christian Bauckhage. Imitation learning at all levels of game-ai. 01 2004.

[64] unity. Unity scripting reference. https://docs.unity3d.com/ScriptReference/. [Online; accessed at 11-August-2020].

[65] unity. Unity user manual - colliders. https://docs.unity3d.com/Manual/CollidersOverview.html. [Online; accessed at 05-August-2020].

[66] unity. Unity user manual - hinge joints. https://docs.unity3d.com/Manual/class-HingeJoint.html. [Online; accessed at 05-August-2020].

[67] Amy Unruh. What is the tensorflow machine intelligence platform? https://opensource.com/article/17/11/intro-tensorflow, November 2017. [Online; accessed at 09-February-2020].

[68] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[69] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Appendix A

Appendix

A.1 Game Guiding Tour in Images

The following is a timeline that simulates a user making a game experiment. The following images are all identified with the corresponding in-game situation. This are all real in-game footage.

Figure A.1: Main Menu - User Interface

Figure A.2: Main Menu: Choose Team - User Interface

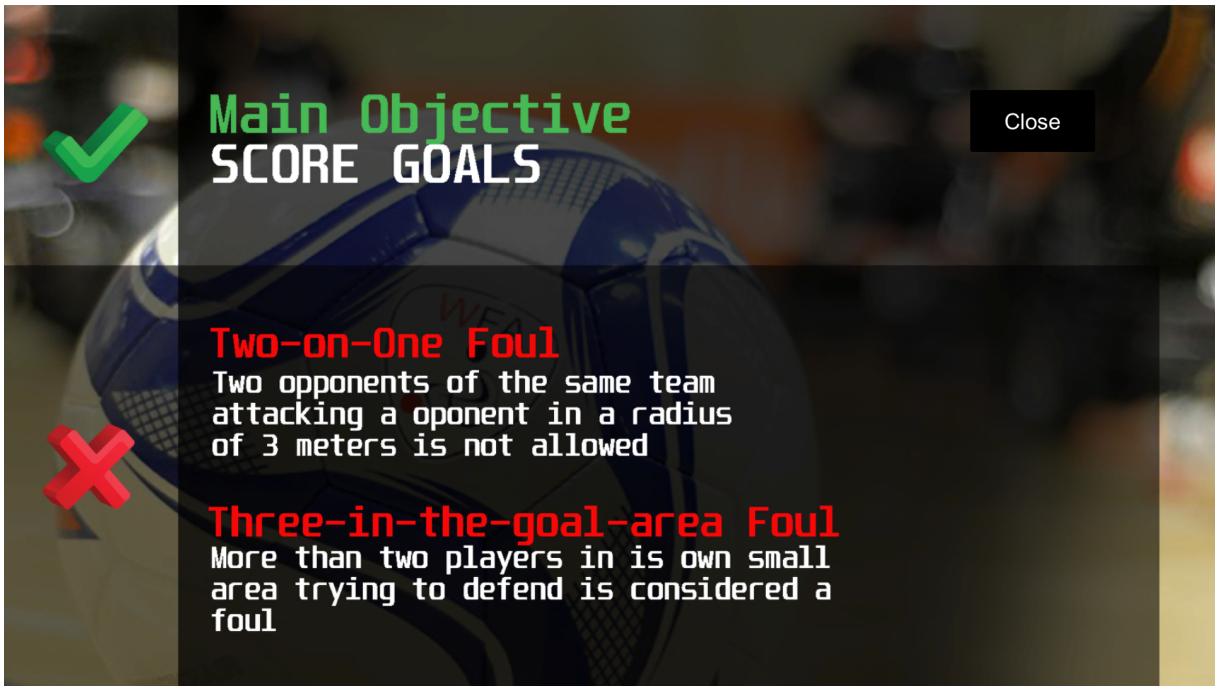


Figure A.3: Main Menu: Rules - User Interface

Figure A.4: Beggining of the game

Figure A.5: Pause Menu - User Interface

Figure A.6: Dribble the ball

Figure A.7: Teammate back wheelchair perspective

Figure A.8: Opponent from wheelchair perspective

Figure A.9: Penalty Situation

A.2 IntellPowerSoccer Questionnaire

IntellPowerSoccer Questionnaire

Venho por este meio convidá-lo/a a participar num estudo de investigação proveniente do projeto Intellwheels 2.0 e da dissertação do aluno Francisco José Sousa Silva. Este estudo tem por base avaliar a experiencia do utilizador ao jogar o IntellPowerSoccer.

I hereby invite you to participate in a research study from the Intellwheels 2.0 project and the dissertation of student Francisco José Sousa Silva. This study is based on evaluating the user experience when playing IntellPowerSoccer.

*Obrigatório

TERMO DE CONSENTIMENTO INFORMADO

Venho por este meio convidá-lo/a a participar num estudo de investigação proveniente da dissertação do aluno Francisco José Sousa Silva, *Wheelchair Serious Game: A Powerchair football video-game with machine learning*. Este estudo tem por base avaliar a experiência do utilizador ao jogar o IntellPowerSoccer e a sua avaliação será feita via eletrónica com base num questionário.

IntellPowerSoccer é um jogo digital de futebol de cadeiras de rodas motorizadas (Powerchair Football) construído em Unity 3D. Faz recurso a técnicas de Machine Learning no treino de agentes inteligentes que são utilizados como adversários ou companheiros de equipa no decorrer do jogo. Este projeto tem como principal objetivo ser um jogo divertido, imersivo e, se possível, ajudar utilizadores reais a aprender a controlar a cadeira de roda motorizada e a modalidade desportiva.

Ao participar neste estudo, será assegurado o direito de, a qualquer momento, desistir de participar no estudo. Serão também salvaguardados o anonimato e a confidencialidade do participante (não haverá identificação nominal do titular, sendo aposto um código).

Garante-se a confidencialidade e uso exclusivo dos dados recolhidos para o presente estudo, em condições de anonimato (não registo de dados de identificação), cumprindo os requisitos da Comissão Nacional de Proteção de Dados. Garantindo ainda que a identificação dos participantes nunca será tornada pública.

A participação será de caráter voluntário, não havendo quaisquer prejuízos caso não queira participar e/ou escolher descontinuar a sua participação. Não está contemplado qualquer ressarcimento ou remuneração para participação no estudo. Não haverá qualquer custo para o participante.

Será obtido Consentimento Livre e Esclarecido de todos os participantes do estudo. O consentimento informado será obtido antes da inclusão do participante voluntário no estudo. Somente o consentimento de pessoas capazes de compreender e comunicar dúvidas sobre o estudo e/ou procedimentos, após lhes ser provida informação oral e escrita sobre o estudo e/ou procedimentos, será considerado satisfatório para a inclusão no estudo.

Agradeço desde já a sua participação,
Francisco Silva

Concorda com o termo a cima descrito? *

Ao responder "sim, concordo", declara ter lido e compreendido o documento a cima referido, bem como as informações verbais que lhe foram fornecidas pela pessoa que acima assina. Foi-lhe garantida a possibilidade de, em qualquer altura, recusar participar neste estudo sem qualquer tipo de consequências. Desta forma, aceita participar neste estudo e permite a utilização dos dados que de forma voluntária fornece, confiando em que apenas serão utilizados para esta investigação e nas garantias de confidencialidade e anonimato que lhe são dadas. (ENGLISH TRANSLATION: By answering "yes, I agree", you declare that you have read and understood the document mentioned above, as well as the verbal information provided to you by the person who signs above. You were guaranteed the possibility to, at any time, refuse to participate in this study without any consequences. In this way, you accept to participate in this study and allow the use of the data that you voluntarily provide, trusting that it will only be used for this investigation and in the guarantees of confidentiality and anonymity that are given.)

- ◯ Sim, concordo. / Yes, I agree
- ◯ Não, não concordo. / No, I do not agree

Seguinte

IntellPowerSoccer Questionnaire

*Obrigatório

1 - IntellPowerSoccer Questionnaire

IntellPowerSoccer é um jogo digital de futebol de cadeiras de rodas motorizadas (Powerchair Football) construído em Unity 3D e faz recurso a machine learning no treino de agentes inteligentes. Estes agentes são por sua vez utilizados para jogar contra o utilizador real, portanto, no decorrer do jogo, os agentes são vistos como os jogadores em campo "controlados pelo computador".

Este jogo tem como principal objectivo ser divertido, imersivo e, se possível, ajudar utilizadores reais a aprender a controlar a cadeira de roda motorizada e a modalidade desportiva.

Nota: O seguinte questionário deverá ser preenchido DEPOIS do utilizador experimentar o video-jogo correspondente. Recomendamos que o utilizador faça um jogo completo (5+5 = 10min).

Observação: Se está a sentir dificuldade em correr o jogo, experimente reduzir a qualidade dos gráficos.

1.1 - Qual o seu género? *

What's your gender?

- ◯ Masculino / Male
- ◯ Feminino / Female
- ◯ Prefiro não dizer / I Prefer not to say

1.2 - Qual a sua idade? *

What's your age?

A sua resposta

1.3 - Qual a sua nacionalidade? *

What's your nationality?

A sua resposta

1.4 - Sofre de algum tipo de incapacidade motora? *

Do you suffer from any motor disability?

- ◯ Sim / Yes
- ◯ Não / No

1.5 - Com que frequência utiliza cadeira de rodas motorizada? *

How often do you use motorized wheelchairs?

- ◯ Muita frequência / Very Frequently
- ◯ Frequentemente / Frequently
- ◯ Ocasionalmente / Occasionally
- ◯ Raramente / Rarely
- ◯ Nunca / Never

Anterior Seguinte

IntellPowerSoccer Questionnaire

*Obrigatório

2 - Utilizadores de cadeira de rodas

A seguinte secção serve apenas para utilizadores reais de cadeira de rodas motorizadas.

2.1 - Após experimentar o jogo, como classifica a condução da cadeira de rodas motorizada em termos sensacionais quando comparada com a condução de uma cadeira de rodas motorizada real? *

After playing the game, how do you classify, in terms of realism, the game wheelchair driving sensation when compared with a real wheelchair driving sensation?

- ◯ Realista / Realistic
- ◯ Moderadamente Realista / Moderately Realistic
- ◯ Neutro / Neutral
- ◯ Pouco Realista / Not so Realistic
- ◯ Irrealista / Unrealistic

Anterior Seguinte

IntellPowerSoccer Questionnaire

*Obrigatório

3 - Experiência de jogo

Game Experience

3.1 - Com que frequência joga jogos de computador / consola? *

How often do you play videogames?

- ◯ Quase sempre / Almost always
- ◯ Muitas vezes / Often
- ◯ Às vezes / Sometimes
- ◯ Raramente / Rarely
- ◯ Nunca / Never

3.2 - Antes de jogar o jogo, o quão familiarizado estava com a modalidade de futebol de cadeira de rodas (Powerchair Football)? *

Before playing the game, how familiar you were with the powerchair football modality?

- ◯ Extremamente / Extremely
- ◯ Muito / Very
- ◯ Moderadamente / Moderately
- ◯ Pouco / Slightly
- ◯ Nada / Not at all

3.3 - Por favor indique como se sentiu ENQUANTO jogava o jogo. *

Please indicate how you felt WHILE playing the game.

	0 - De maneira nenhuma / Not at all	1 - Levemente / Slightly	2 - Moderadamente / Moderately	3 - Bastante / Fairly	4 - Extremamente / Extremely
Estava interessado na historia do jogo / I was interested in the game's story	○	○	○	○	○
Senti-me bem sucedido / I felt successful	○	○	○	○	○
Senti-me aborrecido / I felt bored	○	○	○	○	○
Achei impressionante / I found it impressive	○	○	○	○	○
Esqueci-me de tudo à minha volta / I forgot everything around me	○	○	○	○	○
Senti-me frustrado / I felt frustrated	○	○	○	○	○
Achei cansativo / I found it tiresome	○	○	○	○	○
Senti-me irritado / I felt irritable	○	○	○	○	○
Senti-me habilidoso / I felt skilful	○	○	○	○	○
Senti-me	○	○	○	○	○

completamente envolvido / I felt completely absorbed					
Senti-me contente / I felt content	○	○	○	○	○
Senti-me desafiado / I felt challenged	○	○	○	○	○
Tive de me esforçar muito / I had to put a lot of effort into it	○	○	○	○	○
Senti-me bem / I felt good	○	○	○	○	○

3.4 - Por favor indique como se sentiu DEPOIS de jogar o jogo. *

Please indicate how you felt AFTER you finished playing the game.

	0 - De maneira nenhuma / Not at all	1 - Levemente / Slightly	2 - Moderadamente / Moderately	3 - Bastante / Fairly	4 - Extremamente / Extremely
Senti-me revitalizado / I felt revived	○	○	○	○	○
Senti-me mal / I felt bad	○	○	○	○	○
Senti difícil voltar à realidade / I found it hard to get back to reality	○	○	○	○	○
Senti-me culpado / I felt guilty	○	○	○	○	○
Senti que foi uma vitoria / It felt like a victory	○	○	○	○	○
Senti um desperdicio de tempo / I found it a waste of time	○	○	○	○	○
Senti-me cheio de energia / I felt energised	○	○	○	○	○
Senti-me satisfeito / I felt satisfied	○	○	○	○	○
Senti-me desorientado / I felt disoriented	○	○	○	○	○
Senti-me	○	○	○	○	○

exausto / I felt exhausted					
Senti que podia ter feito mais / I felt that I could have done more useful things	○	○	○	○	○
Senti-me poderoso / I felt powerful	○	○	○	○	○
Senti-me cansado / I felt weary	○	○	○	○	○
Senti-me arrependido / I felt regret	○	○	○	○	○
Senti-me envergonhado / I felt ashamed	○	○	○	○	○
Senti-me orgulhoso / I felt proud	○	○	○	○	○
Senti que tinha voltado de uma jornada / I had a sense that I had returned from a journey	○	○	○	○	○

Anterior Seguinte

IntellPowerSoccer Questionnaire

*Obrigatório

4 - Avaliação pessoal do Jogo

Personal Game Experience Evaluation

4.1 - Quantos jogos fez? *

How many games did you made?

A sua resposta

4.2 - No ultimo jogo que fez, quantos golos marcou? *

In the last game you made, how many goals did you score?

A sua resposta

4.3 - No ultimo jogo que fez, quantos golos sofreu? *

In the last game you made, How many goals did you suffer?

A sua resposta

4.4 - Qual dos seguintes controladores utilizou? *

Which of the following controls you used?

- ◯ Teclado do computador (teclas W, A, S, D) / Computer keyboards (keys W, A, S, D)
- ◯ Joystick (Comando de Consola, Joystick personalizado...) / Joystick controller (Console Controller, personalized joystick)
- ◯ Ambos / Both
- ◯ Outra:

4.5 - Se respondeu à pergunta anterior com "ambos", qual dos controladores achou mais fácil de utilizar? (Se não escolheu a opção ambos, seleccione "Não escolhi ambos") *

If you answered the previous question with "both", which controller did you find easier to use? (If you did not choose both, select "I did not choose both" in this field.)

- ◯ Joystick (Comando de Consola, Joystick personalizado...) / Joystick controller (Console Controller, personalized joystick)
- ◯ Teclado do computador (teclas W, A, S, D) / Computer keyboard (keys W, A, S, D)
- ◯ Outro / Other
- ◯ Achei que eram idênticos / I thought they were identical
- ◯ Não escolhi "ambos" / I did not choose "both"

4.6 - No menu do jogo existe uma opção que permite activar o movimento da câmara (Toggle Look Around). Este controlador simula o movimento do pescoço de um humano através do rato ou outro joystick e permite o utilizador olhar à sua volta. Se activaste este controlador, como classificas a sua utilidade? *

In the game menu there is an option that allows you to activate the movement of the camera (Toggle Look Around). This controller simulates the movement of a human's neck using a mouse or other joystick and allows the user to look around. If you activated this controller, how do you rate its usefulness?

- ◯ Muito útil / Very Useful
- ◯ Útil / Useful
- ◯ Indiferente / Indifferent
- ◯ Pouco útil / Little useful
- ◯ Nada útil / Not useful at all
- ◯ Não activei porque não quis / I did not activate it because I did not want
- ◯ Não activei porque não reparei / I did not activate it because I did not notice

4.7 - Relativamente à Interface de utilizadores (Menus, indicador do resultado actual, indicador de faltas etc...), como a classifica? *

Regarding the User Interface (Menus, current result indicator, fault indicator etc ...), how do you classify it?

- ◯ Muito intuitiva / Very intuitive
- ◯ Intuitiva / Intuitive
- ◯ Neutra / Neutral
- ◯ Pouco Intuitiva / Little intuitive
- ◯ Nada Intuitiva / Not intuitive at all

4.8 - Numa escala de "muito eficientes" a "nada eficientes", como classifica o comportamento dos agentes no decorrer do jogo? *

On a "very efficient" to "not efficient at all" scale, how do you rate the agents' behavior during the game?

- ◯ Muito eficientes / Very efficient
- ◯ Eficientes / Efficient
- ◯ Neutros / Neutral
- ◯ Pouco eficientes / Little efficient
- ◯ Nada eficientes / Not efficient at all

Observações

If you have an opinion, constructive criticism or observation to make about the game, you can do it in the field below.

A sua resposta

Anterior Submeter