Multiple Potential Fields in Quake 2 Multiplayer

Hector Villena Cazorla
This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Personal data
Author: Hector Villena Cazorla
Telephone number: +46739938593
E-mail: HectorVCaz@gmail.com

University advisors:
Stefan Johansson
Martin Fredriksson
Department of Software Engineering and Computer Science

Department of Software Engineering and Computer Science
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet: http://www.ipd.bth.se/~sja/
Phone: 0707-347107
0457-385825
Fax: 0457-385831
Email: stefan.johansson@bth.se
martin.fredriksson@bth.se
ABSTRACT

This thesis may be interesting for developers within video game industry, people who design the behaviour of agent systems, or those interested in complex robotic control systems.

What potential fields give to the world of AI is a new way to implement the behaviour of the agents, giving weights to all the elements in the world, so the agent can decide if making some action deserves the cost, the combination of several potential fields is what makes this technique so efficient. When it comes to the Quake world several examples of potential fields are: armor and health items, the weapons, the obstacles (like walls, boxes, etc...), the enemies, etc...

Quake 2 is an extremely well programmed game, so it can be the perfect platform to show the results of this study much better than real robots where the hardware actuators and sensors could cause a lot of problems.

An agent was created applying this technique. Implemented as a client application, the agent used the Quake protocol to connect and communicate with the server. Some tests were run on the model concerning the different functionalities a gamer agent should implement. The model turned out to be a success in a virtual environment like quake 2. Taking care of all the entities in the map and environmental factors, the agent is able to keep his state in balance, move to the most interesting positions and be able to face combat situations.

The implementation is clean and mathematical oriented. Using algorithms as the heart of the model, we make the code lightweight in comparison with old bot models.

**Keywords:** Potential fields, reactive behaviour, Quake 2
# Table of Contents

1 INTRODUCTION .................................................................................................................. 6
  1.1 BACKGROUND ............................................................................................................. 6
  1.2 RESEARCH QUESTIONS ............................................................................................. 7
  1.3 HYPOTHESIS .............................................................................................................. 7
  1.4 RESEARCH METHODOLOGY ..................................................................................... 8
  1.5 OUTLINE .................................................................................................................... 8

2 POTENTIAL FIELDS............................................................................................................. 9
  2.1 WHAT IS THE POTENTIAL FIELDS TECHNIQUE? .................................................... 9
  2.2 HOW DO THEY WORK? ............................................................................................ 10
  2.3 THE ACTION VECTOR ............................................................................................ 11
  2.4 TYPES OF POTENTIAL FIELDS ............................................................................. 13
  2.5 COMBINING POTENTIAL FIELDS .......................................................................... 15
  2.6 THE GRID ................................................................................................................ 16
  2.7 THE LOCAL MINIMA PROBLEM ............................................................................. 19

3 QUAKE 2: A SOLID PROGRAMMED GAME .................................................................... 21
  3.1 WEAPONS ................................................................................................................ 21
  3.2 AMMO ...................................................................................................................... 21
  3.3 ARMOR ..................................................................................................................... 22
  3.4 HEALTH ................................................................................................................... 22
  3.5 OTHERS .................................................................................................................. 22

4 QASE: THE MAGIC API .................................................................................................... 23
  4.1 HOW DOES IT WORK? ............................................................................................ 23
  4.2 QASE STRUCTURE .................................................................................................. 23

5 THE AGENT ..................................................................................................................... 25
  5.1 CREATING THE AGENT ......................................................................................... 25
  5.2 GETTING STARTED WITH THE AI ....................................................................... 25
    5.2.1 The Grid ........................................................................................................... 25
    5.2.2 Calculating a potential field ........................................................................... 27
    5.2.3 Multiple potential fields working together .................................................. 32
  5.3 ENEMY STATE ....................................................................................................... 34
  5.4 THE LOCAL MINIMA PROBLEM ........................................................................... 35

6 EXPERIMENTS .................................................................................................................. 38
  6.1 EXPERIMENTAL SETUP FOR TEST1 .................................................................... 38
  6.2 EXPERIMENTAL RESULTS FOR TEST1 ................................................................. 39
  6.3 EXPERIMENTAL SETUP FOR TEST2 .................................................................... 43
  6.4 EXPERIMENTAL RESULTS FOR TEST2 ................................................................ 44
  6.5 EXPERIMENTAL SETUP FOR TEST3 .................................................................... 47
  6.6 EXPERIMENTAL RESULTS FOR TEST3 ................................................................ 48
  6.7 EXPERIMENTAL SETUP FOR TEST4 .................................................................... 48
  6.8 EXPERIMENTAL RESULTS FOR TEST4 ................................................................ 48

7 DISCUSSION ..................................................................................................................... 50
  7.1 LIMITATIONS ........................................................................................................... 50
  7.2 DISCUSSION OF RESULTS ...................................................................................... 50
  7.3 DISCUSSION OF METHODOLOGY ........................................................................... 52

8 CONCLUSION ................................................................................................................... 53
1 INTRODUCTION

This section will cover the early stages of this study; the background will be presented using some related work in the field, and situating this work in comparison with the ones that have been already done.

Some research questions will be raised, as well as the hypothesis we will attempt to prove. At last, in the research methodology we will attempt to validate the rationale behind the tests and studies made and provide justification of why it is appropriate to prove right all the hypothesis.

1.1 Background

Video game industry has become one of the fastest growing industries on the globe in the last two decades, while before it was an industry oriented to a small sector of the population is nowadays a huge part of the cake in the media entertainment industry [20].

The industry has become adult and the age’s range from players has got wider, the users are more skilled, and so they demand more complex challenges. This is where the AI developers come into play, there is no doubt that if the adult users need more complicated enigmas than the young ones, both of them demand enemies that may show a realistic behaviour.

Long time has passed since PAC-MAN, and what before was an enemy movement based on stored patterns, is nowadays a “dynamic” path finding method. The enemies use the terrain to their advantage and are able to coordinate between themselves for setting a trap to the player (Incredibly effective in Half-life series or F.E.A.R [16]).

Even though lots of studies have been made about it, spatial navigation keeps on being a challenge in real-time dynamic environments because there are many factors to take into account. The agent does not know beforehand the environment where he stands, he has to keep an eye on changes and also develop the task he has been asked to, and on top of all we want him to be quick at solving problems.

In 1985 Ossama Khatib came up with one new concept while he was looking for a real-time obstacle avoidance approach for manipulators and mobile robots. The technique which he called Artificial Potential Fields lies on the movement of a manipulator in a field of forces. The position to be reached is an attractive pole for the end effector and obstacles are repulsive surfaces for the manipulator parts [1].

Later on Arkin [9] update the knowledge by creating another technique using superposition of spatial vector fields in order to generate behaviours in his so called “motor schema” concept.

The technique has proved to be superior to the old pre-fixed ones, where the programmer had to keep in mind every possible situation, his/her bot would face, to program it efficiently [19]. By using this technique we provide the robot with some
simple rules which will provide him with the information needed to make a good decision.

Most of the studies concerning potential fields are related to spatial navigation and obstacle avoidance [10] [14] [15]. The technique is really helpful for the avoidance of simple obstacles even though they are numerous, and if we combine it with an autonomous navigation approach such as the following-wall method the result is even better, being able to surpass high complicated obstacles [8].

But most of the premises of these approaches are only based on the repulsive potential field of the obstacle and an attractive potential in some goal for the agent to take it [7].

Lately some other interesting applications for the potential fields have been found. The use of potential fields in architectures of multi agent systems is giving quite good results defining the way of how the agents act between themselves. Here we have the work done by Andrew Howard, Maja J Matari´c, and Gaurav S Sukhatme where they developed a Mobile Sensor Network Deployment using Potential Fields, in which a bunch of coordinate agents expand to explore an unknown environment, setting a distance between each of them (repulsive potential placed on each robot), for covering more space of the environment, communicating with themselves and building the map with the data gathered [11].

Also some studies concerning the combination between potential fields and neural networks are turning out in a better performance of the agent in dynamic environments as well as a reduction of the computational charge for the computer, making the agent to learn from the moment it starts moving on the new environment [13].

As far as the author has researched no further studies have been made on how different potentials work together, this study will take care of that, explaining the advantages of using an approach of this kind for an environment like the virtual world of a game.

1.2 Research questions
Here are the questions this thesis will give answers to:

- What is the result of having multiple potential fields working together?
- How do they work together?
- Which steps were necessary to make the fields work together in balance?
- What are the strengths and weaknesses of this approach in comparison with other techniques?
- Can an agent make the kind of performance that is needed in a first person shooter game through the use of multiple potential fields?

1.3 Hypothesis
These are the hypothesis this study will investigate:
• Multiple potential fields is an efficient technique to manage an agent in a real-time virtual environment, concerning the time limits it is subject to (each 100 milliseconds the Quake package has to be sent with the actions the agent has chosen).

• By using multiple potential fields working together we achieve the development of an agent who takes care of different aspects of his state, having all his properties in balance.

• With this technique the agent considers groups of goals for the decision-making, moving accordingly to the sum of the forces expelled by them, making him to situate in the best position to reach all the items.

• The rules we have to define to program an agent will be much less, since it will be implemented using algorithms, and not in a kind of static condition-response programming. The code will take up less space than the traditional ones.

1.4 Research methodology
A deep literature survey through books, thesis, forums and tutorials concerning Potential Fields has been done; also a Quake 2 study to know how to make the agent an efficient combatant and survivor. An application has been developed to test the statements of this thesis; this way the agent is executed as a client application and connects to the Quake 2 server. In order to study the hypothesis, 4 different tests have been designed and run on the application, getting the results which will be later analyzed.

1.5 Outline
The document is divided in 7 parts: The theory concepts about the potential field approach are explained in the first section, then we move to Quake 2 world to know a little bit more about the environment in which the agent will work, a QASE chapter has been also written, to briefly explain how it influenced the implementation of the agent.

Hereafter comes the detailed creation process of the application; in which the experiments of section 6 will be run and finally analyzed in the Discussion chapter. At the end in the conclusion section a summary will be made talking about the results of this study.
2 **POTENTIAL FIELDS**

This section will study the concepts of artificial potential field technique, as well as its strengths and weaknesses. The knowledge displayed here won’t concern always the way the potential fields have been used in the implementation of the agent (this will be shown in the agent section), but it works for providing the reader information about how this technique works in theory.

2.1 **What is the Potential Fields technique?**

Potential fields is a *behaviour-based* technique of the *always-on* type, which means that no trigger causes the behaviour of the agent, rather than that the agent is looking constantly at the environment for changes.

Each potential field is designed by the programmer to represent a behaviour assigned a specific function or task, and only by the combination of all of these potential fields the agent achieves a complex behaviour [3]. The more potential fields we add to the equation, the more problems our agent will know how to solve and indeed the more intelligent it will seem to be.

![Figure 2.1 Schema of a behaviour-based technique](image)

**Figure 2.1 Schema of a behaviour-based technique**

Figure 2.1 shows a *sensory input* (infrared, optic, contact sensors, etc...) , this input can be i.e. the sight of a can (in a imaginary situation in where we have a robot who collects cans), then as the *releaser* is in the mode always-on, the behaviour will be ready to execute some action in response.

What happens inside the *behaviour* stage is what Arbib called the *schema theory*, represented by the boxes of *perceptual schema* and *motor schema*. In schema theory a behaviour consists of a perceptual schema that directs and organizes perception and a motor schema that takes the output of the perceptual schema and produces a pattern of motor activity [5].
In robotics the *perceptual schema* is a robot-centric coordinate space which depends on the orientation of the bot and the location of the goal (can). In other words we would create a vector field with the robot at the origin and the direction pointing to the objective, indicating also the magnitude of forces and velocities. The *motor schema* represents the action to execute, here is the order for the robot to move and catch the can when appropriate.

But, what is a potential field? Let us imagine the agent as an iron particle travelling through magnetic fields generated by objects, depending on the magnetic field the particle will be attracted to one or another object (goals), or maybe, at the contrary, it will be pushed away (obstacles). A potential field is the force each of these objects applies to agent entities. These forces are obviously imaginary; the magnetic explanation is only an analogy of what really happens in the program.

### 2.2 How do they work?

Using potential fields we assign a high value to the position where the goal is. This value has to be high as well in the adjacent positions to the goal, the farther we go from the goal the smaller the value gets, so what the agent has to do is basically to step on the positions with a higher value than the position he is at the moment.

A way to imagine this as a magnetic field is giving a look at the action vectors in a map with one unique attractive potential. Figure 2.2 represents exactly that situation, and we can see how every vector points to the objective from all positions on the map, seeming to draw the magnetic lines of a magnetic field.

![Figure 2.2 Action vectors in an attractive potential field [3]](image-url)
Every potential field has a radius of action; out of this radius there are no forces which attract the agent to the goal (exactly like a magnet). But there are some cases in which the limitation of the potential fields using a radius is not appropriate, i.e. our agent will be implemented taking care of all the objectives no matter where they are (of course the distance to them will make the potential to decrease) since it helps the agent to situate all the time in the best position for having all items the most accessible possible.

Coming back again to the theory explanation, while the agent is inside the radius of action from any goal he develops a behaviour, each behaviour of the agent creates an output spatial vector called action vector which represents the speed and orientation of the agent.

### 2.3 The action vector

One way to think of a potential field is to think of it as a mapping from one vector (agent) into another (goal) [3], the resulting vector is the so called action vector and it is in fact the most important component in the potential fields approach, since it is what defines the movement of the agent. It is composed of a magnitude and a direction:

- The magnitude is the length of the vector.
- And the direction is just the direction of the vector in the map.

The magnitude of the vector depends on the amount of force that the potential field is exerting on the agent; this amount varies according to the agent’s position in or out the radius of the potential field. If it is in, two values are the ones who determine the magnitude, firstly the distance between the agent and the goal, and secondly the importance of that object for the agent.

The direction will depend on the type of potential field the agent is facing, these vectors represents forces and it depends if the object is a goal desired by the agent which will make the vector points towards the goal, or it is an object the agent has to avoid instead, which will make the vector to point to the furthest position from the obstacle.

1. Let’s imagine an attraction potential field, and let’s place the goal in the coordinates \((X_G, Y_G)\), now we place the agent in the coordinates \((X_A, Y_A)\).

   To determine the magnitude of the vector we just have to calculate the distance between both points in the 2-Dimensional space:

   \[
   d = \sqrt{(X_G - X_A)^2 + (Y_G - Y_A)^2}
   \]

   And now the direction, which we will find through the calculation of the angle between the agent and the goal, we use the arctangent function to get the angle:

   \[
   \theta = \tan^{-1}\left(\frac{Y_G - Y_A}{X_G - X_A}\right) \quad [3]
   \]
Now depending on the situation of the agent, the distance to and the radius of the goal, as well as the radius of the force coming from the goal, we can calculate which will be the final action vector which will push the agent towards the goal:

\[
\begin{align*}
\text{If } d&< r & \Delta G_X &= \Delta G_Y &= 0 \\
\text{If } r &\leq d &\leq s + r & \Delta G_X &= \alpha (d - r) \cos (\theta) & \Delta G_Y &= \alpha (d - r) \sin (\theta) \\
\text{If } d &> s + r & \Delta G_X &= \alpha s \cos (\theta) & \Delta G_Y &= \alpha s \sin (\theta)
\end{align*}
\]  

Let’s consider “s” as the sphere of influence of the goal, “r” as the actual radius of the goal, “d” the distance between the agent and the goal, “\(\theta\)” the angle of the resulting vector from uniting the goal and agent points, while \(\alpha\) is the constant value which represents the importance of the goal for the agent.

Now we have created the action vector with the components \((\Delta G_X, \Delta G_Y)\) which will depend on the variables we named before, if the agent is inside the radius of the proper goal then no forces will be applied on the agent, since it is assumed that he has got the goal and therefore the components of the action vector will be 0.

If the agent stands outside of the radius of the goal but inside of the force radius then it will be attracted to the goal according to the distance to it, by this is meant that the magnitude of the vector will be proportional to the actual distance between agent and goal.

If the agent stands out from both radiuses, we will give the vector the biggest magnitude we can, making his priority to reach the outside radius and to go in.

2. Let’s take a look at a repulsion potential field, the procedure for calculating the direction and magnitude is the same we saw in the last example.

The difference comes now, when we have to form the action vector:

\[
\begin{align*}
\text{If } d&< r & \Delta O_X &= \Delta O_Y &= \infty \\
\text{If } r &\leq d &\leq s + r & \Delta O_X &= \alpha (s + r - d) \cos (\theta) & \Delta O_Y &= \alpha (s + r - d) \sin (\theta) \\
\text{If } d &> s + r & \Delta O_X &= \Delta O_Y &= 0
\end{align*}
\]

If we are inside the radius of the proper obstacle then the repelling force is infinite, the agent cannot be there by any means.

If the agent is out from the obstacle but inside its radius of action then the force (vector’s magnitude) is still proportional to the distance between the agent and the obstacle, notice that \(\alpha\) is the variable we use for setting the importance that the obstacle has over the agent (to make it harder or softer).

At last we have the situation when the agent is out from the radius of action of the object, in this case the agent achieved the objective (which was to get out from the repelling action of the obstacle), and thereof the action vector is 0.
But these are not the only directions an agent can take when it is experiencing the force of a potential field, there are many types more.

2.4 Types of potential fields

Potential fields can be created according to the needs of the programmer, for a project like the one this thesis is about there are many alternative potential fields, like i.e. one related to the firing of the gun, or the movement of an agent inside a water lake. Anyway there are some basic potential fields which are really helpful for tasks like following a wall, escaping from an enemy, or looking for health when it is necessary:

Uniform

It is the most simple of all the types of potential fields, it orders the agent to go in a specific direction but not to seek for a goal, it could be used for ordering the agent to run away, follow a wall or return to the battlefield (if the agent would have the chance to go out from the map, that is not our case since the environment is virtual and all the limits are well defined).

Perpendicular

This behaviour is also a simple one, and it might be useful to avoid walls as we can see on the picture or to run away from a territory.
Attraction

This potential field attracts the agents to it, it is perfect for the Seek-Goal behaviour, and as we will see in Quake 2 environment it will be very helpful, since there are lots of items (weapons, power-ups, health) which will have this kind of potential field.

![Figure 2.5 Graphic of the line forces in an attraction potential field](image)

Repulsion

Just the opposite of the previous potential field, this one push away the agents from it, it is very good for avoiding an obstacle, like an enemy, or some terrain cover with lava (which is dangerous for the agent). Combining these last two potential fields we could program a very efficient bot for navigation in dynamic and real environments.

![Figure 2.6 Graphic of the line forces in a repulsive potential field](image)

Other ones

Of course there are several potential fields more, used fewer times than the previous ones but still quite useful depending on the situations. If the robot gets stuck in a “local minima” a random potential field which pushes him to another position can make the agent to go out from his blockade.
2.5 Combining Potential Fields

According to Arbib’s theory some type of behaviours are possible to combine, the way to make it is through their action vectors. Each behaviour has its own vector, so if two behaviours are using the same sensors and motors we can add their vector fields to produce a single one and make the robot moves towards the best position for getting both objectives.

![Diagram of Arbib's schema theory]

The advantages are clear in a situation like the avoiding of an obstacle to get a goal behind it; as it was showed in Section 2.4 the goal and the obstacle have their own potential field, one positive (the goal) and one negative (the obstacle), and the combination of both vectors is what will make our agent successful to accomplish the objective and surpass the obstacle.

We have seen in the 2.3 Section how to get the action vector of an attractive potential field and a repulsive one, and it did not seem too complicated since only one force is acting upon the agent. Let us put both potentials in the same scenario, now we have two objects exerting forces over all the agents on the map, but how do we combine them? According to Arbib we only need to add the vectors, since the sensors (in this case the agent knows the location of the objects in the map) and the motor schemas (walk action) are the same:
• We have the action vector of the attractive potential field: \((\Delta_{GX}, \Delta_{GY})\)
• And the action vector corresponding the repulsive one: \((\Delta_{OX}, \Delta_{OY})\)

Well, and this will be the action vector the agent will have to follow if he wants to reach the goal by avoiding the obstacle:

\[
\Delta X = \Delta_{GX} + \Delta_{OX} \\
\Delta Y = \Delta_{GY} + \Delta_{OY}
\]

Now we have a two behaviour agent which is able to reach goals and avoid obstacles if necessary. But as we will further see, this kind of combination does not work between all the potential fields we will have in the environment, once again let us remember it only works with potentials the agent detect with the same sensors and which motor schemas are the same.

![Figure 2.8 Repulsive and attractive potential fields working together.][3]

This explanation using vectors is quite clarifying in Figure 2.8, but when it is applied to an environment like a videogame world with lots of entities and obstacles we need to use some mechanism to gather all the potential fields in each position of the map, that is the main purpose of the grid.

2.6 The Grid

In the subject of this thesis we are working in a virtual combat situation [17], so it is logical that the bot will have to open fire over an enemy, take health, armour or special items, use elevators, swim, etc... This is not a normal environment and it demands lots of potential fields, and lots of behaviours in response.
This amount of potential fields is not easy to manage together, so we need some way to organize them, making clear to the agent which movement he should do. We divide the map into a grid; each square in the grid is a position, and each position will have a value representing how close that position from the objective is. The size of the squares depends on the environment, the game and our purposes.

There are two main ways of creating the grid:

1- ) A grid which covers the whole map, calculating in each square the value of all the potential fields, the usual way to calculate it is through this formula:

\[ \text{PF}(x,y) = \frac{c}{d^2} \]

Where \( c \) is the strength of the charge (the actual importance that the item has for the agent, i.e. a single bullet will have less strength of charge than a box of bullets), and \( d \) is the distance from the position to the object.

We will calculate all the potential fields that are affecting that position, and once they are calculated, we proceed to add them:

\[ \text{PF}_\text{tot}(x,y) = \sum \text{PF}(x,y) \]

This way may be interesting for those games which do not need to have a quick answer or games with static environments where we can precalculate a lot. It is not the case of Quake 2 since it is a client/server application, which means the agent will be exchanging information with the server frequently (each 100 milliseconds) and we
cannot grant that if every time a movement is made we have to recalculate the whole grid.

2- ) The second way is to create the grid on the positions around the agent, that way the computer does not need to recalculate every position on the grid all the time. It may be more difficult in the sense that the agent has to take care of the obstacles all the time, checking if the goals are behind a wall and therefore calculate the best route to avoid them. With the other technique this was not necessary since the positions adjacent to the walls had very low values which made the agent avoid those positions, anyway as we pointed out before the whole grid is a computational charge we can not assume in this kind of game.

We can take a look at how the agent local grid is on Figure 5.1 in Section 5..

We must know how the potential fields will affect the behaviour of the agent, or more specifically the values on the positions, i.e. in the Quake world the agent can find one position very interesting to take because it is closer to a big health box, but let us imagine an enemy is coming towards him in that direction, and at this moment his health level is low, what previously was an interesting position is now an undesirable one, and the positions behind him will get a higher value for him to run away from the enemy. In Quake 2 the environment changes constantly due to the action of the agent and the other entities playing the game, so a recalculation of the grid values needs to be made each time we receive a package from the server.

Sometimes it is possible that the agent can not move because there are not better positions around him than the one he is standing on at the moment. This can happen i.e. when two opposite potential fields are exerting the same force over the agent, making his action vector equal to 0, or if a goal is in the same parallel of the agent and between them there is a symmetrical wall which the agent will not know where to surround. There are still methods to avoid the local minima.
2.7 The local minima problem

When an agent get hung up in an obstacle because the value of the positions lead it there we say it got hung up in a “local minima”, which means that the agent cannot access a position with a higher level than the previous one, and it cannot go back in his steps because these positions have a smaller value than the one he is standing on. That happens also when opposite but equal forces affect the agent making him to freeze without taking a better square.

Here we have an example of local minima:

![Figure 2.10 Graphic representation of the local minima problem with one charged object](image1)

In this situation we have only the potential field of the goal, which means that the agent is being attracted to it, as we do not have other forces which can affect the agent, he is going to walk straight to the objective. As we can observe in Figure 2.10 the agent will collide with the wall, and he will try to reach the next position, which is impossible without avoiding the corner within he is.

As we explained before, we could solve this by applying a negative potential field to the obstacle, which will push away the agent from the walls, making the agent to surround their limits, but what if the repulsive forces have the same magnitude and the agent is standing in the middle of them, we would still be in the same problem.

![Figure 2.11 Graphic representation of the local minima problem with 2 potential fields](image2)
But there are some ways to get out of this problem, here are different kind of potential fields which could be useful for it:

Random potential field
It is a potential field which makes the agent to move to different random positions; it could be useful if it manages to get the agent out of the positions in the local minima.

Follow wall method
When we know where the agent came from, and the conditions a position needs to satisfy to have a local minima, we could apply a constant force to escape from the position until we have straight line of vision with the goal, one escape force is in example the follow wall behaviour, the agent will walk parallel the wall surrounding it [4] [8].

Create a tree
Another option is to create a tree, representing the movements the agent can make from the position he is when he is stuck in the local minima. These optional movements will be the leafs of the tree, and each one of them will have the next positions the agent could take from that point, and so on. It is not allowed to have two leafs equal (representing the same position). [18]

The idea is that from each leaf of the tree (possible positions of the agent) we check if the agent has a direct visual of the goal or by contrary the wall is still in the way. This way we would fill the local minima by analyzing before all the possible options of the agent. Finally we would apply a best-first search algorithm, when one of the positions returns a visual of the object, for getting the shortest path to the objective, avoiding therefore the obstacle and the local minima.

Inside the Section 5 there is a chapter devoted to the local minima avoidance problem where is explained what did we exactly implement for solving the problem.
3 QUAKE 2: A SOLID PROGRAMMED GAME

Quake II was released on the 6th of December of 1997, a year later than the previous Quake. It is a first person shooter game designed and programmed by John Carmack and id software.

As a FPS game the objective is clear, to kill all the enemies the player will find over the 10 compounds the history is formed of. The player will have 10 weapons at his service, as well as several power-up items, the maps are totally 3-Dimensional with complex parts as elevators, lava and water lakes, doors, ladders, etc...

The game was a great advance for its time, developing some sort of AI (now the enemies didn’t stand frozen when the player started to fire at them, they try to avoid the rockets while firing at him) and having a very solid 3D engine, which has been used in lots of titles later, there we have Half-life, Sin, Soldier of Fortune or Daikatana.

Quake 2 is also a server-client application; since it has a multiplayer game incorporated which allow people to play the game together through the internet or using a normal LAN. A socket is created between the server and the clients, the only info needed is the IP of the server and the port it is using (usually the 27910). Quake 2 multiplayer game has its own protocol which runs over TCP/IP; nowadays both the protocol and the game source code are public, which makes the programming of bots and mods possible.

Here next comes the list of entities which will generate a potential field that will affect the agent:

3.1 Weapons

The agent will seek the most effective weapon all the time he will be in the environment, at the moment of implementation the agent has been programmed with a priority in the order of weapons he should use, according to the effectiveness and power of each one. In Table 5.4 we have a list of the existing weapons in the game as well as their strength of charge; the values have been given according to the importance of each weapon for the agent (Weapons table available in appendix A).

3.2 Ammo

Items which give the agent the ammunition necessary for each weapon, and if the weapons have different weights (importance) for the agent, the ammo items have it as well, for the agent it depends on various factors:

1. The importance of the weapon for the agent and therefore its ammo.
2. The amount of ammo the agent has already in his inventory.
4. If the agent is carrying that weapon in his hands.

As the factors are so localized for each item, it was impossible to apply this calculation to the AmmoPotential once all the potentials of the ammo items are added, so these values were taking into account at the moment of the calculation of the strength of charge. Table 5.5 shows the formulas used for the calculation of this value.
3.3 Armor

The amount of damage an agent can take depends on two things: the health and the armor level. While the health reaches its top at 100, the armor can rise till 200, and that really makes a difference in a combat situation since the agent can take heavy damage and survive his enemy. There are 3 main armor items, each of them gives a different level of armor to the agent, and logically the more points they give the more priority they have for the agent. We can take a look at them in Table 5.3.

3.4 Health

The third important aspect in the game is the health level. So far we had weapons to fire the enemy, armor to protect ourselves and now at last the health level, it goes from 0-200 but its natural balance is 100. This means that if the agent manages to gather any health over 100 this level will automatically decrease slowly till 100 again. The importance for the agent to get these items depends on the actual health’s and armor’s level of the agent. We have 4 health items in the game, but only three will be considered as healing items, the mega health will be considered as “other items” due to its rareness and way of working with the health points (it gives 100 points to the agent, not matter what the health level is, this makes it independent of the way the environmental factors affect the Healing Potential). Table 5.2 and Table 5.6 (for the mega health item) show the strength of charge assigned to these items.

3.5 Others

There are also other items in the game, but the agent doesn’t take care of them all. Anyway as the 3 main aspects of the game are covered with the previous items, these ones will have a constant importance for the agent, which will make the agent go for them without taking care of many factors, only the health level will make the item to decrease its importance if it is low, so we have to be careful with the value we give to these items. Table 5.6 takes care of the so called other items.

Some tables have been elaborated grouping all the items in the game according to their type, thanks to these tables now we know which items or weapons will be preferred for the agent, or which ones he will not care about. The objective of the thesis is to find out if the hypotheses hold developing an efficient agent using the potential fields technique, so it is not necessary that the agent takes care of all the variables in the quake world. But once the AI is developed is very scalable so it is quite easy to keep on adding more potential fields to the code, only taking care of how they act and react with the other potential fields. The tables showing the Quake 2 items, with the data the programmer has taken into account to assign the strength of field values, are placed in the Appendix A.

The development of this thesis has been possible thanks to the work of Bernard Gorman, Martin Fredriksson and Mark Humphrys who in 2005 released an application programming interface called QASE which really simplifies the work to program the client part in a multiplayer game like Quake 2. [2]
4 QASE: THE MAGIC API

QASE (Quake 2 Agent Simulation Environment) is an API designed to unify the procedures in the creation of game agents for different multiplayer games, programmed in java, it is a powerful tool and object oriented; and with which a bit of research quite easy to use. 3 versions have been developed so far, increasing the efficiency and adding more methods. What makes QASE so general and so good is that it does not take into consideration the code of the games; it uses the protocol between server and client instead.

Thanks to this tool I have been able of concentrate my work on the design of the AI without taking care of other subjects as the communication packets, the representation of the 3-Dimensional maps in programming structures and many more things.

4.1 How does it work?

QASE has been designed for creating the packages the server is waiting for, at the moment this version of QASE works with Quake 2 protocol, but is so scalable that with a few changes would be able to work with Quake 3 or many other titles which use the same engine as these two.

Each 100 milliseconds the server sends a package to all the clients with the game info (objects, players, movements, actions, etc…). The clients process the info and build a package to send back to the server. The only thing the server doesn’t send is the structure of the map, which every agent must have in their Quake 2 folders.

Between the client and the server there is a class called \textit{proxy}, which the client sees as the server and the server sees as the client. The class has the objective of receive and send packages. When a package is received it is sent to the thinking part of the client (the AI we will design), the program will analyze the info contained in the package, decide which action the agent will do in response and send it back to the \textit{proxy} class which in turn build a package and send it to the server.

4.2 QASE structure

The structure is divided into 16 packages, and with a total of 119 classes, that are supplying AI techniques, controlling protocol subjects, giving the user geometrical functions to deal with the vectors, getting and setting player’s info or the entities on the game, etc...QASE structure can be found in the Figure B.1 located in the Appendix B.

We will briefly see the most important packages QASE provide for their users, since most of classes work at an internal level which the programmer will not touch.

QASE supplies some AI packages for the programmer to use if necessary; in this case I haven’t found a use for them, since I considered this technique was better to build from zero.

The bot package counts with an interface, and several classes which makes easy and intuitive the creation of a bot of our own, i.e. if I want to order the bot to fire at some
moment, I only have to call the function action.setAttack(True), the same occurs with
the rest of the actions, like crouch, walk, jump, most of them are controlled by flags. It
has one class called Basic Bot which is the one my agent directly inherits from, and it
takes care of calling undercover to all the necessary classes to provide us the methods
we will need to create a functional agent.

The com package is mostly used by the API in a internal way, which means the
programmer doesn’t have to take care of touching it.

The file package is the responsible for reading and managing the files concerning
Quake 2, like the BSP files (maps), the DK2 files (in game demos) and the PAK files
(also related to maps). It is not usually used for the casual programmer; there are other
classes which call to the methods contained in this package.

The info package is also another one which is use in a very internal way, it contains
several classes inside but only one will be of interest to us, the User class, with options
for the player we are creating. [6]

The state package is one of the most useful in the API as we will have to be all the
time navigating through all its classes, for making the agent move, for checking the
position of the agent, its orientation, etc… The example I said before about the order of
firing is a function contained in the Action class inside the state package.

The tools package is the one dealing with vectors, it contains the classes’ vector3
and vector2 which are really helpful for managing the 3dimensional coordinates of the
agent, containing operations such as subtract, add, make the scalar product, etc…
5 THE AGENT

Potential fields have been applied before to robotics, since it is a suitable method for real-time dynamic environments. But from the personal experience of the author no such a thing has been done with first person shooter games.

The reason may be in the easiness of programming an agent using waypoints. A waypoint is a mark than a programmer can place into specific locations of a map, that way the agent knows his location and the points he can reach from there.

The AI of the agent has been built from 0, based on the theory concept of how the potential fields work, and applied to the movement patterns of the agent. In next sections we will see the whole process, from the creation of the agent to the way he applies the algorithms to calculate where to go:

5.1 Creating the agent

As mentioned in section 4, the programming of the agent has been made using the QASE library. In the API there is one called Bot with an interface called the same way. One of the classes in this package is the abstract BasicBot class which implements the Bot interface. In this class most of methods the agent will use are implemented, anyway there are more methods in other classes which we can reach from within the BasicBot class.

First of all, we have the new Java Application package with its own Main procedure where the object OwnBot is created. OwnBot is an object of the MyBot class; MyBot class is a subclass of Basic Bot.

The constructor of the MyBot class needs two arguments at least, the name and the skin of the agent (OwnBot=new MyBot("Homer","male");), after we have the object OwnBot right created it is moment for it to connect to the server: OwnBot.connect("localhost",27910);

When the constructor of the MyBot class is called the runAI method starts executing in a parallel thread, waiting for the proxy class to process the info sent by the server and send it to the runAI method. As soon as it receives the info, it generates an answer and sends it back.

There are also another threads executing, checking if the agent is dead or not (for if it needs a respawn), or the one in charge of the communication between the client and the server, receiving, sending and solving the situation if any package is lost.

5.2 Getting started with the AI

5.2.1 The Grid

First of all, let us decide what kind of grid this situation needs, as there are certain positions in the map very far away from the agent, it is no point in calculating the value
of those, since the agent could not reach there in 100 milliseconds (which is the time before the server sends another package to the client and when the program has to recalculate the values of the positions close to the agent).

The map in Quake 2 is not a chess board (board delimited by fixed squares), the system is based in 3-Dimensional coordinates, which means there are as many positions as points exist in the space of the map. Decided to create the grid, the longest distance the agent could make in the 100 milliseconds was calculated; running at maximum speed the agent made an average of 28 points each interval.

A grid is created and divided into 8 positions around the agent, each one identified by one number:
- Up = 0.
- Diagonal up-right = 1.
- Right = 2.
- Diagonal right-down = 3.
- Down = 4.
- Diagonal down-left = 5.
- Left = 6.
- Diagonal left-up = 7.

This is how the grid will finally look:

![Figure 5.1 Grid created around the agent, 8 possible positions to move.](image)

The magnitude of the movement vector of the agent is 28 as maximum, and that vector is pretty simple to represent over 1 axis, but when the agent must take the diagonal directions this vector is made of two components, which magnitude must be equal to 28, I calculated the exactly length of the cathetus for getting a hypotenuse of 28 points, but the result was not exact (19.7989), so I decided to round up the result and make the components 20 each one.
Now every time the RunAI method is called, it will calculate the value of each position around the agent, the process of calculating the value is easy:

5.2.2 Calculating a potential field

To calculate the force that a potential field is exerting on a position we need to know two things: first the strength of charge of that object and second the distance between the object and the selected position.

The strength of charge is a value which represents the importance that this object has for the agent, first of things is to choose the range we want for this value, I chose from 0-100, being 100 an item really desirable for the agent. Of course there are a few items that get out of the scale, but these items are so weird to find in the game that if the agent sees them he should go and collect them right away, among this items we have the quad damage, the invulnerability, or the mega health.

If you take a look at the implementation (specifically the calculateStrength method) you will see how the agent calculate this value, first, the formula depends on which item we are considering:

• **Health**: The method returns a constant value according to the type of health item it is, there are three types:

<table>
<thead>
<tr>
<th>Health</th>
<th>Strength of charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stim Pack</td>
<td>20</td>
</tr>
<tr>
<td>First Aid</td>
<td>30</td>
</tr>
<tr>
<td>Med Kit</td>
<td>70</td>
</tr>
</tbody>
</table>

*Table 5.2 Strength of charge of healing items*

• **Armor**: The method returns a constant value according to the type of armor item it is, there are four types:

<table>
<thead>
<tr>
<th>Armor</th>
<th>Strength of charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flak Jacket</td>
<td>50</td>
</tr>
<tr>
<td>Combat Suit</td>
<td>70</td>
</tr>
<tr>
<td>Body Armour</td>
<td>80</td>
</tr>
<tr>
<td>Armour shard</td>
<td>10</td>
</tr>
</tbody>
</table>

*Table 5.3 Strength of charge of armor items*

• **Weapons**: The weapons have a constant value according to the relevance they have for the agent (table A.1), the method checks if the agent has already the weapon in the inventory, if so it returns 0, so the agent wont be interested on taking that weapon anymore.
<table>
<thead>
<tr>
<th>Weapon</th>
<th>Strength of charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blaster</td>
<td>0</td>
</tr>
<tr>
<td>Shotgun</td>
<td>40</td>
</tr>
<tr>
<td>Super-shotgun</td>
<td>55</td>
</tr>
<tr>
<td>Machine-gun</td>
<td>60</td>
</tr>
<tr>
<td>Chain gun</td>
<td>50</td>
</tr>
<tr>
<td>Hand grenade</td>
<td>0</td>
</tr>
<tr>
<td>Grenade launcher</td>
<td>45</td>
</tr>
<tr>
<td>Rocket launcher</td>
<td>75</td>
</tr>
<tr>
<td>Hyper Blaster</td>
<td>65</td>
</tr>
<tr>
<td>Rail gun</td>
<td>90</td>
</tr>
<tr>
<td>BFG-9000</td>
<td>70</td>
</tr>
</tbody>
</table>

Table 5.4 Strength of charge of weapon items

- **Ammo:** The strength of charge of the ammo items depend on three variables, the relevance they have for the agent (table A.2), if the agent wears at the moment the weapon this ammo is for, and the actual level of ammo.

  If the agent is handling the weapon for the actual ammo item we increase the value of the ammo, also the lower the amount of this ammo is the bigger the value gets and finally if the type of ammo corresponds to a powerful weapon the value will get even higher.

<table>
<thead>
<tr>
<th>Ammo</th>
<th>Strength of charge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ammo weapon/s</td>
</tr>
<tr>
<td>Shells</td>
<td>8*(100-ammo)/10</td>
</tr>
<tr>
<td>Cells</td>
<td>6*(200-ammo)/10</td>
</tr>
<tr>
<td>Bullets</td>
<td>4*(200-ammo)/10</td>
</tr>
<tr>
<td>Grenades</td>
<td>5*(50-ammo)/4</td>
</tr>
<tr>
<td>Slugs</td>
<td>3*(50-ammo)/2</td>
</tr>
<tr>
<td>Rockets</td>
<td>4*(50-ammo)/2</td>
</tr>
</tbody>
</table>

Table 5.5 Strength of charge of ammo items depending on the weapon the agent wields.

There are reasons for have chosen these values, first of all we have to balance the strength resulting from the calculation with only the current ammo level (without the weapon factor). Each weapon has its own limit of ammunition, in some of them the limit is high (200 in the guns and blasters) while in others is low (50 in the railgun, grenade and rocket launchers), this force us to be careful with the result, and that is the reason why the formulas have different divisors.
Also the importance of the weapon, this ammunition is for, influences on the equation, let’s take a look i.e. to the Cells and the Bullets. While both of them have their limit in 200, one (cells) is divided by 4 and the other (Bullets) by 5. That is because the Hyperblaster and the BFG (which use cells) are more important for the agent than the Chain and Machine guns (which use bullets).

Other times, other factors have influenced the calculation of the strength, like for example, let us take a look at the Slugs and Rockets. Both belong to the most powerful weapons in the game, the railgun and the rocket launcher, both have the same limit, but their formula differs, why?

The answer is the weapon’s firing frequency, while the rocket launcher shoots almost two rockets per second, the railgun shoots 1 time each.

About the calculation when the agent wears the weapon, the item is for, let’s notice that is just the same formula but multiplied by one constant, only to give that item more weight. This difference about if the agent carries or not the weapon the item is for, is to avoid the change of weapon in the middle of a combat situation.

Note: All these factors have been chosen based on the tastes and measurements of the programmer.

- **Others**: The rest of items are power-ups, usually powerful items that the agent finds in rare occasions, and without limit of pickups. That’s why they have a constant strength of field, usually high (QuadDamage, Invincibility, Adrenaline, etc...).

<table>
<thead>
<tr>
<th>Others</th>
<th>Strength of charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mega health</td>
<td>400</td>
</tr>
<tr>
<td>Quad damage</td>
<td>500</td>
</tr>
<tr>
<td>Invulnerability</td>
<td>500</td>
</tr>
<tr>
<td>Adrenaline</td>
<td>70</td>
</tr>
</tbody>
</table>

*Table 5.6 Strength of charge of other items*

So once we have the strength and the distance from the agent to the item, we are able to calculate the potential field.

\[
\text{Potential} = 1000 \times \frac{\text{Strength of charge}}{\text{distance} + 1}
\]

We check what type of item it is and once we calculate the potential we add it to the corresponding Potential field variable:
Now, by the time we have finished calculating each of these variables for a certain position, comes the time to modify them according to the environmental issues (all formulas are shown in pseudo code):

```java
if EntityType = healing {
    HealingPotential=HealingPotential+Potential;
} else if EntityType = armor {
    ArmorPotential=ArmorPotential+Potential;
} else if EntityType = ammo {
    AmmoPotential=AmmoPotential+Potential;
} else if (EntityType = enemy){
    aweap=3; // I increase the importance of weapons and armor while in combat. The agent will look for weapons while fighting.
    EnemyPotential=EnemyPotential + 5*(getHealth()-30)*800/entDist+(getArmor()-100)*800/entDist;
} else {
    OtherPotential=OtherPotential+Potential;
}
```

Figure 5.7 Pseudo-code showing how the Types of potential are added.

As you can see on the formula, the health level of the agent is 4 times more important than the armor level, this will make the agent to interest on a health item even if his armor level is very high. Notify that the reason why I multiply the getArmor()/100 part with the getHealth division is only for if the agent is at some point full of health, that way we would get a HealingPotential=0 and not a negative one if the agent has some armor.

Very similar to the Health formula, is the ArmorPotential. It deals with the same variables, this time the armor limit is 200, and the less the health level is the smaller the result of this formula gets, this means that if the agent is low of life the taking of an armour item is not that interesting, a health item would generate a higher strength of field, still if the armor item is close the agent would go for it.
Figure 5.9 ArmorPotential field being modified by the environmental factors of health and armor levels.

Here the strength of field would reach its highest when the agent has no level of armour and is totally healthy. On the contrary it would reach the lowest value when the agent is almost full of armor and the health level is low.

About the AmmoPotential and the OtherPotential, they are affected by the Health of the agent directly, getting their highest value when the agent is full of life:

$$\text{AmmoPotential} = \text{AmmoPotential} \times \left( \frac{\text{getHealth}()}{100} \right)$$

Figure 5.10 AmmoPotential field being modified by the environmental factor of agent’s health.

$$\text{Other Potential} = \text{Other Potential} \times \left( \frac{\text{getHealth}()}{80} \right)$$

Figure 5.11 OtherPotential field being modified by the environmental factor of agent’s health.

The enemyPotential depends on the health and armor of the agent, when those levels reach a certain limit the enemyPotential variable changes its value, becoming repulsive or attractive:

$$\text{EnemyPotential} = \text{EnemyPotential} + \left( \frac{4000 \times (\text{getHealth}() - 30)}{\text{distance}} \right) + \left( \frac{800 \times (\text{getArmor}() - 100)}{\text{distance}} \right)$$

Figure 5.12 EnemyPotential field being modified by the environmental factors of agent’s health and armor.

Repulsive if the health is under 30 and the Armor is under 100 or attractive if the agent is not in such a critical situation.

As we can see, the EnemyPotential will have a really high value for the agent to go for him as his first priority.
And finally we have the last Potential field variable, the WallPotential, which keeps the agent out of the walls when there are items behind them.

\[ \text{WallPotential} = \text{WallPotential} + \text{WallDistance} \]

Once we have all the potentials modified according to the environmental values, we add them all.

\[ \text{SumPotential} = \text{HealingPotential} + \text{ArmorPotential} + \text{AmmoPotential} + \text{EnemyPotential} + \text{WallPotential} + \text{OtherPotential} \]

We repeat the process for each position around the agent, we compare the results and we order the agent to move there.

### 5.2.3 Multiple potential fields working together

Once got the strength of charge, we only need another variable to calculate the potential field; the package Quake 2 sends each 100 milliseconds contains all the info about all the entities in the map, so we know the position of each object.

Get the distance between the agent and the object is as easy as calculating the magnitude of the action vector, which we previously calculated by subtracting the coordinates of both points.

We agree then that the agent will choose one entity or another according to the importance of the item and the distance to it. As we are working with multiple potential fields the agent will be sensitive to groups of potentials, i.e. if there is one really important item for the agent in one corner of the map and there are lots of items in the opposite corner with a medium relevance the agent will probably lead towards the group of items, since they are adding their attractive force. It would depend of course of the distance and the strength of charge, if the lonely item would have a bigger potential than the sum of all potentials of the group, the agent will run to it.

Let’s make a little test:

**State:** Agent with 60 points of health and 60 points of armor.
**Items:** One Med_Kit item (it gives 25 health points) and one body_combat armor (it gives 100 points of armor).
**Position:** Just in the symmetric middle of both. 200 points of distance to each one.
**Result expected:** The agent will choose to take the armor item, even if the health level is not in 100, the body_combat gives the agent 100 points of armor which is more interesting for a combat situation.
Potential field is calculated first:

As we are exactly in the middle of both items, there will be only two interesting positions (the agent will calculate the 8 around him anyway), the one closer to one item, and the one closer to the other item; we first calculate the position closer to the armor item. We will call the position closer to the Body Combat position 1 and the position closer to the Med Kit position 2 (these names have been chosen for the reader to see the formulas and the variables clearer, the actual directions are 3 to the body armor and 7 to the health large).

\[
\text{Potential}=1000*\text{Strength of charge}/(\text{distance}+1)
\]

\[\text{PF}_1\text{ of Healing }= 1000*70/(228+1)=305.67\]

\[\text{PF}_1\text{ of Armor}= 1000*80/(172+1)=462.42\]

The agent moves 28 points each time interval, which means that from the closer object will be 172 points away and 228 from the furthest.

Now we calculate the weight of each potential according to the environmental factors, let’s go first with the healing potential:

\[
\text{HealingPotential}= 305.67* \left( \frac{100 - 60}{25} - \frac{60}{100} \times \frac{100 - 60}{100 - 60 +1} \right) = 305.67* \left( \frac{40}{25} - \frac{60}{100} \times \frac{40}{41} \right) = 305.67* \left( 1.6 - 0.585 \right) = 305.67* 1.015 = 310.25
\]
ArmorPotential = 462.42 * \left( \frac{200 - 60}{100} - \frac{100 - 60}{100} \times \frac{200 - 60}{200 - 60 + 1} \right) = 462.42 * \left( \frac{140}{100} - \frac{40}{100} \times \frac{140}{141} \right) = 462.42 * \left( 1.4 - 0.396 \right) = 462.42 * 1.004 = 464.26

\text{PF}_{\text{TOTAL1}} = 310.25 + 464.26 = 774.51

Now we calculate the potential fields from the second position (closer to the health item):

PF$_2$ of Healing = $1000 \times \frac{70}{173} = 404.62$

PF$_2$ of Armor = $1000 \times \frac{80}{229} = 349.34$

We apply once again the environmental factors to the weights (as the agent state is the same we have the same environmental values):

HealingPotential = $404.62 \times 1.015 = 410.68$

ArmorPotential = $349.34 \times 1.004 = 350.73$

\text{PF}_{\text{TOTAL2}} = 410.68 + 350.73 = 761.41

The agent will move to the first position, which is closer to the body_combat item, but by a minimal advantage, and in this case I find it appropriate since it is a very balanced situation, would we get the health item first for having almost all the health up or would we get more armor even though we already have some. Finally the factors point to the armor, because it gives lots of points of armor and the environmental situation is really balanced for the armor and the health.

Let us remember also that if the armor reaches its top at 200, and the agent will be interested on armor items until he reaches it; the health level always has more weight in the decision making.

5.3 Enemy state

The enemies are together with the walls, the two repulsive forces of the game. In the case of the wall it is always repulsive, but with the enemies it depends, it is the only potential field which changes type in game time.

Some circumstances have to occur for the enemy to be a repulsive potential field:

- The health of the agent is less than 30 points and it does not have an armor over 100.
If any of these conditions are not fulfilled then the enemy has an attractive potential field around him, which will make the agent go towards him while firing. We can see how the EnemyPotential changes according to the formula in Figure 5.12, when the result becomes negative the potential is repulsive, otherwise it is attractive (positive).

When the agent has an enemy in line of vision the game state changes completely and another layer arises, the main objective of the agent is to kill the enemy now, we do that by placing a huge attractive potential field on the enemy, much higher than the rest of them. Here we have two behaviors depending on if the enemy potential field is attractive or repulsive:

- **Enemy potential field is repulsive**
  The agent will run away from the enemy and will be looking for the item he needs to be in advantage, for that matter it is given a higher strength of charge to the essential items (variable aweap in Figure 5.7).

- **Enemy potential field is attractive**
  The agent will run towards the enemy firing all the way, once there he will keep on firing to the enemy while running around him (which will make the agent more difficult to kill).

Once the enemy is killed the combat layer goes down and the game returns to a normal state.

Note: The agent is programmed to always wield the best gun available, the best gun criteria is according to the table A.1 (Appendix A).

### 5.4 The local minima problem

There are some problems that the potential fields approach cannot solve. When an agent has to avoid a simple obstacle the solution is easy, the obstacle is exerting a repulsive force, the goal exerts an attractive one and by adding the components of the action vectors the agent will find the way to surround the obstacle. But with complex obstacles, like rooms or walls the result is not the same. Let us put it this way, the agent is blind, the only thing he knows is the position of the entities in the map and if they are behind a wall.

**Figure 5.14** Four force lines that cause, by counteracting themselves, the agent to get stuck in local minima.
The agent cannot recognize structures, only points in the space, and we give a repulsive force to the point where the agent know a obstacle is, which will make the agent to not get closer to the point, but that is not enough. If we take a look at Figure 5.14, we can see how the line forces would affect the agent.

What happens is that all the forces are counteracting with each other, which will make the agent to get stuck and be frozen; this is one of the main issues when applying the potential fields approach. As we saw in Section 2.7 there are some techniques to get rid of local minima.

Random field technique or the creation of a tree with all the possible movements will be discarded. The random field would maybe get the agent out of the local minima but it is not sure that we will not come to the same position getting stuck again, it will help in other kind of local minima but not in this situation.

The creation of a tree is just not feasible, the server needs the answer of the client very quickly, and 100 milliseconds are not enough time to create the tree, insert all the leaves and make a best-first search.

Complex avoiding obstacles mechanisms have not been developed for the achievement of this study, since it is not the subject what it is about. The walls have a repulsive force but according to the distance they have to the goal.

The only tool the agent can use for exploring the scenario is the method getObstacleDistance, which we basically could define as a sonar tool for the agent. As we have said before the agent is blind when he enters the map, the only things the server tells him are the position of the entities (items and enemies) and information about them.

The only way the agent has for finding an obstacle is to throw lines in a certain direction and with a certain magnitude, if the line hits a wall in its way it returns the distance to the obstacle.

We may see it clear in Figure 5.15, but the agent only sees points in the space where there is an obstacle, he does not know the shape or length of the wall, he simply knows where a certain position is occupied by an obstacle and to get away from it, since it generates a repulsive potential field.

For simply obstacle avoidance I developed a geometric theory which allows the agent to get rid of some simple constructions. The main idea is that if the agent is chasing an item behind a wall he will always choose the positions which project the line with the longest distance from the wall to the Goal, that way the agent will surround the wall.
Here another problem showed up; just when the agent was at the edge of the obstacle he did not know where to move since two positions had the best value for him and these positions were in opposite directions, what made the agent get stuck in a kind of local minima, we can have a deeper look in the graphic below:

The line, that the agent throws to check where the obstacle is, returns the same value in two different points, since these two points have the same distance to the obstacle, that made the agent go crazy and trap him in a local minima. I solved this problem applying some inertia to the agent, making him to go straight after he decide to change the direction, he only advances one position more than he was supposed to in that direction. After applying this, the agent didn’t get stuck and was able to avoid the obstacle, going straight to the goal.

Of course these are simply methods for letting the study center on the multiple potential fields working together. For having more information regarding the real time obstacle avoidance we can take a look at the work of J.Borenstein and Y.Koren [8] which applying grids, potential fields and follow wall methods developed a good system able to explore an unknown and dynamic scenario.
6 EXPERIMENTS

Here the methodology presented in the Section 1.4 will be applied to try to find support for the hypothesis. The results will be shown in further sections and finally discussed in Section 7.

To support the theories proposed in this study, we need to see results showing the decision-making of the agent. We will achieve that by explaining all the process, from the moment the agent makes the decision of taking one direction (calculating each type of potential, modify them according the environmental factors, add them for each direction and compare them to see which is the biggest), till he gets the item.

6.1 Experimental setup for test1

We will take a look first at the way the agent keeps his basic balance (health and armor):

First of all, we design the scenario and the variables playing on it:
Scenario with 4 items: Health large, Body armor, Ammo cells and Weapon Railgun. Distributed with an original distance of:

<table>
<thead>
<tr>
<th>Item</th>
<th>Strength of charge</th>
<th>Initial distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Med Kit (Health Item large)</td>
<td>70</td>
<td>337.82</td>
</tr>
<tr>
<td>Body armor (Armor large)</td>
<td>80</td>
<td>358.50</td>
</tr>
<tr>
<td>Ammo Cells</td>
<td>(200-ammo)/4 = 200/4 = 50</td>
<td>344.07</td>
</tr>
<tr>
<td>Weapon Railgun</td>
<td>90</td>
<td>378.22</td>
</tr>
</tbody>
</table>

Table 6.1 List of items in the experiment, with their values of strength of field and distance to the agent.

Environment:

Figure 6.2 Graphic representing the experiments environment.
6.2 Experimental results for test1

Now we will proceed to calculate the Potential of each item on each position of the grid:

<table>
<thead>
<tr>
<th>Potentials</th>
<th>Pos 0</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 4</th>
<th>Pos 5</th>
<th>Pos 6</th>
<th>Pos 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body A</td>
<td>222.52</td>
<td>234.97</td>
<td>249.51</td>
<td>255.63</td>
<td>247.96</td>
<td>232.41</td>
<td>220.56</td>
<td>216.57</td>
</tr>
<tr>
<td>Cells</td>
<td>144.89</td>
<td>149.02</td>
<td>145.66</td>
<td>137.70</td>
<td>130.84</td>
<td>127.55</td>
<td>129.80</td>
<td>136.38</td>
</tr>
<tr>
<td>Railgun</td>
<td>237.32</td>
<td>225.58</td>
<td>215.34</td>
<td>211.55</td>
<td>215.53</td>
<td>226.50</td>
<td>238.55</td>
<td>243.98</td>
</tr>
<tr>
<td>Total</td>
<td>811.32</td>
<td>811.84</td>
<td>817.5</td>
<td>824.28</td>
<td>828.13</td>
<td>828.25</td>
<td>823.3</td>
<td>816.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Potentials</th>
<th>Pos 0</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 4</th>
<th>Pos 5</th>
<th>Pos 6</th>
<th>Pos 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body A</td>
<td>222.52</td>
<td>234.97</td>
<td>249.51</td>
<td>255.63</td>
<td>247.96</td>
<td>232.41</td>
<td>220.56</td>
<td>216.57</td>
</tr>
<tr>
<td>Cells</td>
<td>144.89</td>
<td>149.02</td>
<td>145.66</td>
<td>137.70</td>
<td>130.84</td>
<td>127.55</td>
<td>129.80</td>
<td>136.38</td>
</tr>
<tr>
<td>Railgun</td>
<td>237.32</td>
<td>225.58</td>
<td>215.34</td>
<td>211.55</td>
<td>215.53</td>
<td>226.50</td>
<td>238.55</td>
<td>243.98</td>
</tr>
<tr>
<td>Total</td>
<td>811.32</td>
<td>811.84</td>
<td>817.5</td>
<td>824.28</td>
<td>828.13</td>
<td>828.25</td>
<td>823.3</td>
<td>816.33</td>
</tr>
</tbody>
</table>

Table 6.3 List of potential values in each position of the grid, calculating finally the total potential and therefore the position to go.

This is the potential each item has for the agent in each position; let us remember that the way of calculating these potentials is:

Potential=1000*Strength of charge/(distance+1)

So, as the strength of the field is a constant we can affirm that the potentials vary according to the distance from the agent to the item (except for the ammo items which strength of field vary according the weapon that the agent carries and the amount of ammo). Without taking any environmental factors into account, the agent chooses to go to position 5 on the grid, towards the health item, and two reasons contribute to this decision:

1- ) The Med Kit is the closest item to the initial position of the agent, and it has a high strength of charge.

2- ) The body armor is also attracting the agent to the lower half of the map.

Note: Notice the small difference between the position 4 in which the agent would be walking to the middle of both items and the position 5 which he finally takes.

If the body armor item would not be in the environment, the agent would have gone towards the weapon, even if it is the furthest object from him.
Applying environmental factors to the potentials:

According to the method explained in Section 5.2.2, we are going to calculate the Potential types that we find in this experiment: Healing Potential (Med Kit), Armor Potential (Body Armor), AmmoPotential (Cells) and OtherPotential (Weapons) and modify them according to the environmental factors (agent with 100 health level and 0 armor level):

<table>
<thead>
<tr>
<th>Type P</th>
<th>Pos 0</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 4</th>
<th>Pos 5</th>
<th>Pos 6</th>
<th>Pos 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healing</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Armor</td>
<td>445.05</td>
<td>469.95</td>
<td>499.02</td>
<td>511.26</td>
<td>495.93</td>
<td>464.83</td>
<td>441.12</td>
<td>433.14</td>
</tr>
<tr>
<td>Ammo</td>
<td>144.89</td>
<td>149.02</td>
<td>145.66</td>
<td>137.70</td>
<td>130.84</td>
<td>127.55</td>
<td>129.80</td>
<td>136.38</td>
</tr>
<tr>
<td>Other</td>
<td>237.32</td>
<td>225.58</td>
<td>215.34</td>
<td>211.55</td>
<td>215.53</td>
<td>226.50</td>
<td>238.55</td>
<td>243.98</td>
</tr>
<tr>
<td>Total</td>
<td>827.27</td>
<td>844.56</td>
<td>860.03</td>
<td>860.53</td>
<td>842.31</td>
<td>818.89</td>
<td>809.48</td>
<td>813.52</td>
</tr>
</tbody>
</table>

Table 6.4 List of potential values in each position of the grid calculated taking care of the health and armor levels of the agent, calculating finally the position to go.

Finally the agent chooses to move to position 3, which is exactly the direction where the armor item is, the most valued for him considering the environmental factors (health, armor, weapon he carries and ammo of the weapon).

Let us study in a more general way how do these factors affect the different potentials. We will start considering only the Health and the Armor of the agent, and how do them vary one specific potential of each type, I have chosen the Potential fields of the position 3:

<table>
<thead>
<tr>
<th>Type P</th>
<th>100H 0A</th>
<th>100H 100A</th>
<th>100H 200A</th>
<th>50H 0A</th>
<th>20H 160A</th>
<th>60H 60A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Healing</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>438.81</td>
<td>658.21</td>
<td>219.40</td>
</tr>
<tr>
<td>Armor</td>
<td>511.26</td>
<td>255.63</td>
<td>0.0</td>
<td>383.44</td>
<td>0.0</td>
<td>255.63</td>
</tr>
<tr>
<td>Ammo</td>
<td>137.70</td>
<td>137.70</td>
<td>137.70</td>
<td>68.85</td>
<td>27.54</td>
<td>82.62</td>
</tr>
<tr>
<td>Other</td>
<td>211.55</td>
<td>211.55</td>
<td>211.55</td>
<td>132.22</td>
<td>52.88</td>
<td>158.66</td>
</tr>
<tr>
<td>FDecision</td>
<td>Pos 3</td>
<td>Pos 2</td>
<td>Pos 0</td>
<td>Pos 4</td>
<td>Pos 5</td>
<td>Pos 4</td>
</tr>
</tbody>
</table>

Table 6.5 List displaying the modified potential of each item in position 3 according to the environmental factors of health and armor, showing also the first movement the agent will make.

The Fdecision field refers to the first movement the agent will do if we run the simulations with those values of Health and Armor. The health and armor values have been chosen to illustrate the flexibility of the model. The reason why these have been selected is that they represent the main important combinations between health and armor.
Now, let’s study a little the reason for the movement decision of the agent, as well as the first item he finally picked up:

- The first column of the Table 6.5 does not need any comment, since it is the simulation we run first in Table 6.4, and we saw where the agent went and why.

- In the 2\textsuperscript{nd} case the agent finally took the ammo (which is reached by walking to Position 1), but he started walking towards the position 2, the reason is that both potentials (armor and ammo) from the right side of the map were more powerful together than the weapon potential in the left. Once the agent is placed in the right he made the decision of taking the ammo, since his armor level was quite high.

- 3\textsuperscript{rd} case made the agent to take the weapon, and we saw again how at the beginning he was affected by two potentials which made him go towards the direction in the middle of both items, at some point the weapon potential is stronger than the ammo’s and the agent goes to take the railgun. Notice that both potentials health and armor are 0, since the agent has reached the limit in both factors.
• In the 4th the agent went forward to the Med Kit. The health always weights more than the armor, so even if the difference between the top level of armor and his actual level is bigger than the health’s the agent decides to go first for the MedKit. Once again let us notice than the agent does not go straight to the Med Kit from the beginning, the forces of the body armor and the health work together and makes him to go straight down.

• In the 5th case the agent goes straight to the medkit pushed by his actual low health level (only 20); the fact that the armor level is so high also contributes to that decision. The ArmorPotential is 0 since the health level of the agent is very low and the armor level is huge. As soon as the agent recovers health points, the potential of the armor raises.

• In the 6th attempt he takes the health as well but by a very small difference compared to the body armor. Actually the only reason why it takes the Health is because it is closer to the initial position of the agent. He walks to position 4 since both potentials affect him, but at some point one of them will exert a bigger force over him; at that moment, the values are:
  775.63 to go to Position 5 (towards the Med Kit).
  771.85 to go to Position 3 (towards the Body Armor).

The values of the rest positions at that moment are from 760 to lower, so we can appreciate how minimal the difference is.
6.3 Experimental setup for test2

The objective of this study is to notice how the AmmoPotential varies according environmental factors, like the weapon the player handles at the moment or the current amount of ammo the agent has of its type.

Scenario:

Entities: The 6 types of ammo items: Slugs, Shells, Bullets, Rockets, Grenades and Cells.
Disposition: The 6 types of ammo items have been placed in a hexagon’s disposition, for them to be at the same distance from the initial position of the agent.
Environmental factors: Since the nature of the Ammo Strength of charge calculation is special, the environmental factors affect the Strength of charge straight.

<table>
<thead>
<tr>
<th>Item</th>
<th>Strength of charge</th>
<th>Initial distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ammo Cells</td>
<td>(200-ammo)/4 = 50</td>
<td>343.34</td>
</tr>
<tr>
<td>Ammo Bullets</td>
<td>(200-ammo)/5 = 40</td>
<td>346.85</td>
</tr>
<tr>
<td>Ammo Grenades</td>
<td>2*(50-ammo)/3= 33</td>
<td>347.67</td>
</tr>
<tr>
<td>Ammo Slugs</td>
<td>5*(50-ammo)/4= 62</td>
<td>346.13</td>
</tr>
<tr>
<td>Ammo Shells</td>
<td>(100-ammo)/2 = 50</td>
<td>344.14</td>
</tr>
<tr>
<td>Ammo Rockets</td>
<td>(50-ammo) = 50</td>
<td>344.32</td>
</tr>
</tbody>
</table>

Table 6.7 List of items placed in the experiment, with their values strength of field value and distance to the agent.

Map:

![Figure 6.8 Graphic representing the experiment’s environment](image)
6.4 Experimental results for test2

We will first calculate the AmmoPotential, since it is the only type of potential in this scenario. Let us remember that the only environmental factor which affects the AmmoPotential at this level is the health of the agent (Figure 5.10), and since the health of the agent is 100 in this test, the potential will be just the addition of the potential of each item:

<table>
<thead>
<tr>
<th>Potentials</th>
<th>Pos 0</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 4</th>
<th>Pos 5</th>
<th>Pos 6</th>
<th>Pos 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>145.20</td>
<td>145.89</td>
<td>139.80</td>
<td>131.86</td>
<td>126.83</td>
<td>126.09</td>
<td>130.51</td>
<td>138.18</td>
</tr>
<tr>
<td>Bullets</td>
<td>114.99</td>
<td>120.11</td>
<td>128.41</td>
<td>134.99</td>
<td>134.34</td>
<td>126.90</td>
<td>118.88</td>
<td>114.30</td>
</tr>
<tr>
<td>Grenades</td>
<td>94.64</td>
<td>94.30</td>
<td>98.08</td>
<td>104.69</td>
<td>110.47</td>
<td>111.37</td>
<td>105.94</td>
<td>99.09</td>
</tr>
<tr>
<td>Slugs</td>
<td>178.60</td>
<td>169.60</td>
<td>166.21</td>
<td>169.60</td>
<td>178.60</td>
<td>190.40</td>
<td>195.58</td>
<td>190.40</td>
</tr>
<tr>
<td>Shells</td>
<td>144.86</td>
<td>153.55</td>
<td>157.72</td>
<td>153.55</td>
<td>144.86</td>
<td>136.78</td>
<td>134.04</td>
<td>136.78</td>
</tr>
<tr>
<td>Rockets</td>
<td>144.79</td>
<td>138.18</td>
<td>130.51</td>
<td>126.09</td>
<td>126.55</td>
<td>131.86</td>
<td>139.80</td>
<td>145.89</td>
</tr>
<tr>
<td>Total</td>
<td>823.10</td>
<td>821.65</td>
<td>820.77</td>
<td>820.82</td>
<td>821.69</td>
<td>823.42</td>
<td>824.78</td>
<td>824.66</td>
</tr>
<tr>
<td>AmmoP</td>
<td>823.10</td>
<td>821.65</td>
<td>820.77</td>
<td>820.82</td>
<td>821.69</td>
<td>823.42</td>
<td>824.78</td>
<td>824.66</td>
</tr>
</tbody>
</table>

Table 6.9 List of potential values in each position of the grid, calculating finally the total potential and therefore the position to go.

The agent decides to go and take position 6, towards the slugs item. It was the predictable decision, since all items were almost at the same distance to the agent, so what really inclined the balance was the strength of the field, which the slugs item had the highest. As we see there is a little difference between the AmmoPotential in each position, it is logical since there were no environmental factors which acted on the agent, this is the simplest situation that can be set for the variation of the AmmoPotential.

Applying environmental factors to the Strength of charge (the weapon):

Now we are going to choose one position of the grid and calculate the potential of each item according to the weapon the agent handles. Let’s use the position 6 as the sample, the state of the agent is the same: 100 Health, no armor and no ammo of any kind.

<table>
<thead>
<tr>
<th>Potentials</th>
<th>Hyperb</th>
<th>Glauncher</th>
<th>Rlauncher</th>
<th>Railgun</th>
<th>Shotguns</th>
<th>Chaingun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cells</td>
<td>313.23</td>
<td>130.51</td>
<td>130.51</td>
<td>130.51</td>
<td>130.51</td>
<td>130.51</td>
</tr>
<tr>
<td>Bullets</td>
<td>118.88</td>
<td>118.88</td>
<td>118.88</td>
<td>118.88</td>
<td>118.88</td>
<td>237.77</td>
</tr>
<tr>
<td>Grenades</td>
<td>105.94</td>
<td>199.04</td>
<td>105.94</td>
<td>105.94</td>
<td>105.94</td>
<td>105.94</td>
</tr>
<tr>
<td>Slugs</td>
<td>195.58</td>
<td>195.58</td>
<td>195.58</td>
<td>236.59</td>
<td>195.58</td>
<td>195.58</td>
</tr>
<tr>
<td>Shells</td>
<td>134.04</td>
<td>134.04</td>
<td>134.04</td>
<td>214.47</td>
<td>134.04</td>
<td>134.04</td>
</tr>
<tr>
<td>Rockets</td>
<td>139.80</td>
<td>139.80</td>
<td>279.61</td>
<td>139.80</td>
<td>139.80</td>
<td>139.80</td>
</tr>
<tr>
<td>FDecision</td>
<td>Pos 0</td>
<td>Pos 5</td>
<td>Pos 7</td>
<td>Pos 6</td>
<td>Pos 2</td>
<td>Pos 4</td>
</tr>
</tbody>
</table>

Table 6.10 List of potential values in each position of the grid calculated taking care of the weapon the agent wields, calculating finally the position to go.
The agent took the ammo item necessary for the weapon he was carrying in every test, even if the first move he made was not to the closest position to the item.

- Here we have examples like the first one, where the agent first moved to 0, which is caused by the high potentials of the slugs (the most important ammo item by default and situated in direction 6) and the Rockets (it is the ammo for the 2\textsuperscript{nd} most important weapon of the game) counteracting the high attracting force of the cells.

- In the second case the agent was wearing the grenade launcher and went straight to position 5 (pointing to the grenades). The explanation is quite easy, the grenades were exerting a high potential over the agent, also the slugs, and finally the bullets (situated in direction 3). The bullets and the slugs were counteracting with each other favouring also the selection of position 5.

- The agent headed straight to pick up the rockets, same reason that we saw in the grenades potential.

- The slugs item is already the strongest one, so if on top of that we have the railgun, the potential field is huge, what made the agent take the position 6.

- The shells were picked up straight as well, since the other potentials were balanced together, and the slugs potential were placed in the same parallel than the shells, which made no variation in the direction of the action vector.

- In the final case the agent took the position 4 for directing towards the bullets item, this was caused by the potentials (direction 6) of slugs, rockets (direction 7) and grenades (direction 5) counteracting the potentials in the right.

**Applying environmental factors to the Strength of charge (the ammo):**

The next and last table will display how the amount of ammo affects the strength of the potentials. We will have 6 situations where the agent wears the appropriate weapon for each ammo item, and several ammo quantities for each of these situations.

The results shown will be the item potential, the first direction the agent will take, and the item he picked the first. The position 6 is the one chosen to run the experiment.

First column represents the same state the agent had in the previous table in the position 6, since there is no ammo level affecting the potentials:
<table>
<thead>
<tr>
<th>PotentialBest</th>
<th>0 Ammo</th>
<th>25 Ammo</th>
<th>50 Ammo</th>
<th>100 Ammo</th>
<th>200 Ammo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cells</strong></td>
<td>313.23</td>
<td>274.08</td>
<td>234.92</td>
<td>156.61</td>
<td>0.0</td>
</tr>
<tr>
<td>Position 0</td>
<td>Position 0</td>
<td>Position 0</td>
<td>Position 7</td>
<td>Position 5</td>
<td></td>
</tr>
<tr>
<td>Cells</td>
<td>Cells</td>
<td>Cells</td>
<td>Rockets</td>
<td>Slugs</td>
<td></td>
</tr>
<tr>
<td><strong>Bullets</strong></td>
<td>237.77</td>
<td>208.05</td>
<td>178.33</td>
<td>118.88</td>
<td>0.0</td>
</tr>
<tr>
<td>Position 4</td>
<td>Position 4</td>
<td>Position 4</td>
<td>Position 6</td>
<td>Position 7</td>
<td></td>
</tr>
<tr>
<td>Bullets</td>
<td>Grenades</td>
<td>Grenades</td>
<td>Slugs</td>
<td>Rockets</td>
<td></td>
</tr>
<tr>
<td><strong>Grenades</strong></td>
<td>199.04</td>
<td>99.52</td>
<td>0.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Position 5</td>
<td>Position 7</td>
<td>Position 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grenades</td>
<td>Rockets</td>
<td>Rockets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Slugs</strong></td>
<td>236.59</td>
<td>116.71</td>
<td>0.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Position 6</td>
<td>Position 2</td>
<td>Position 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slugs</td>
<td>Shells</td>
<td>Shells</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Shells</strong></td>
<td>214.47</td>
<td>160.85</td>
<td>107.23</td>
<td>0.0</td>
<td>N/A</td>
</tr>
<tr>
<td>Position 2</td>
<td>Position 1</td>
<td>Position 6</td>
<td>Position 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shells</td>
<td>Cells</td>
<td>Slugs</td>
<td>Slugs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Rockets</strong></td>
<td>279.61</td>
<td>145.89</td>
<td>0.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Position 7</td>
<td>Position 6</td>
<td>Position 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rockets</td>
<td>Slugs</td>
<td>Grenades</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.11 List displaying the modified potential of each item in position 6 according to the environmental factors like the weapon the agent wields and the amount of ammo, it shows also the first movement the agent will make and the first item picked up.

We can observe how the potential decreases according the increase of agent’s ammo level in Figure 6.12. That may make the agent to change his decision about the first direction to take, leading sometimes to the pickup of another entity if the potential has diminished sufficiently.

![Figure 6.12 Graphic showing how the potentials of each ammo item decrease according to the table 6.11.](image)
Each weapon has its own max ammo capacity, when the agent reaches it, the potential becomes 0 and all the items of that class are not interesting anymore for the agent (until he consumes some of this ammo).

The Rockets case is a good example of how the environmental factors influence the potential of the ammo items; in all situations of this row the agent wields the rocket launcher. The amount of ammo he has in possession at the beginning is 0 so the potential value is high. Now let us calculate the potential for when the agent has 25 rockets; we see how the value of the potential decreases, letting the Slugs and Grenade potentials influence more on the agent and leading him to take the position 6 (which is in the middle of these three forces).

Finally when we reach the limit of rockets in the inventory, the potential falls to 0; now we have 5 forces acting around the agent. As the environment has lost one strong potential in the upper part, the agent will be more attracted from the potentials in the lower-east side, which will make him eventually move there, but as the slugs potential is so strong the agent finally takes the position 4, instead of the position 3 which is the one in the middle of all the potentials of the map.

### 6.5 Experimental setup for test3

The first hypothesis proposed in this study was the efficiency of this technique in real-time server-client arquitectures, stating that the implementation would not take lots of resources to work and therefore the agent would not have lag responding the server.

For proving this statement, several tests have been made, from maps with a small number of entities to big maps with lots of them.

The time samples have been taken measuring the whole runAI method, and all the methods referred from there, which is what has really been implemented.

<table>
<thead>
<tr>
<th>Maps</th>
<th>Nº entities</th>
<th>Nº samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test2.bsp</td>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>Test5.bsp</td>
<td>6</td>
<td>120</td>
</tr>
<tr>
<td>Test1.bsp</td>
<td>28</td>
<td>311</td>
</tr>
<tr>
<td>Test12.bsp</td>
<td>41</td>
<td>183</td>
</tr>
<tr>
<td>Tsm_deadsimple.bsp</td>
<td>75</td>
<td>117</td>
</tr>
</tbody>
</table>

*Table 6.13 List of maps tested and the number of entities within them.*

The tests have been run in a Pentium M Centrino 2.0 GHz with 1 Gb of RAM DDR2 and an ATI MOBILITY RADEON X700. The server was created as a no dedicated server, which means that the Quake 2 server and client are created both at once, and then our agent can connect.

Note: Both parts the agent and the server have been executed in the same machine.
6.6 Experimental results for test3

After running the test we have noticed how much the time samples differ if the entities are visible or not. If the entities are not visible, the agent just calculate the distance from the wall to the item (as we saw in the agent section), but in change if the entities are visible the agent makes much more calculations, with the consecutive charge for the processor.

Here are the results for the test we run:

<table>
<thead>
<tr>
<th>Maps</th>
<th>Size</th>
<th>Nº entities</th>
<th>Nº samples</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test2.bsp</td>
<td>798x798</td>
<td>1</td>
<td>49</td>
<td>1.70</td>
</tr>
<tr>
<td>Test5.bsp</td>
<td>798x798</td>
<td>6</td>
<td>120</td>
<td>3.82</td>
</tr>
<tr>
<td>Test1.bsp</td>
<td>798x798</td>
<td>28</td>
<td>311</td>
<td>5.84</td>
</tr>
<tr>
<td>Test12.bsp</td>
<td>798x798</td>
<td>41</td>
<td>183</td>
<td>6.08</td>
</tr>
<tr>
<td>Tsm_deadsimple.bsp</td>
<td>2142x1950</td>
<td>75</td>
<td>117</td>
<td>39.88</td>
</tr>
</tbody>
</table>

Table 6.14 List of maps tested, including their size, number of entities within, nº samples taken and the average time the application has taken for respond to the server.

As we can see in Table 6.14, the complexity of the model grows according to the size of the map instead to the number of items in it. We have not reached the top in these tests, but the robot has been able to play fine in all the environments the programmer has tried.

For the agent all the entities count at every moment, no matter the distance to them, so this opens one possibility of improving the algorithm. The improvement would be to assign a radius of action to the agent, taking care only of the entities inside of it. If there would not be entities in, then the radius could grow dynamically until new ones are reached, and decrease when lots of entities are around the agent.

6.7 Experimental setup for test4

In this last experiment we are going to check data related to the code, like the size of the final application or the number of packages and lines it took to program it.

The purpose is to prove that the agent can face most of problems in the Quake 2 world being programmed with a light code.

In the experiment the implementation of the agent has been compared to 3 well-known bots in Quake 2, regarding their source code, and binaries: EraserBot, CRBot and Q2Bot Core.

6.8 Experimental results for test4

The agent of this study has been implemented using the API called QASE, which means that lots of methods were already implemented, and the programmer just had to centre his work on the AI method. This has been very helpful to focus the study on the
programming of a good AI, but it also forced the programmer to adapt to the existing methods and the way the API is implemented.

Two of the bots of this comparison are server-bots, which means that the program has to be run from the server instead of connecting to the game like a client bot. Of course this has disadvantages like that they slow down the speed of the server, also that they can compromise the stability of the server if they have any kind of failure.

The other two are the Q2 Bot Core and MyBot, which are programmed from the client part, and that makes the model more reliable and more efficient (fast).

Next we are going to see the data of each bot concerning to its source code and binary files:

<table>
<thead>
<tr>
<th>Names</th>
<th>Version</th>
<th>Source code size</th>
<th>Binary files</th>
</tr>
</thead>
<tbody>
<tr>
<td>EraserBot</td>
<td>v1.2</td>
<td>128 Kb</td>
<td>1.52 Mb</td>
</tr>
<tr>
<td>CRBot</td>
<td>v1.14</td>
<td>160 Kb</td>
<td>600 Kb</td>
</tr>
<tr>
<td>Q2Bot Core</td>
<td>v1.0</td>
<td>173 Kb</td>
<td>275 Kb</td>
</tr>
<tr>
<td>MyBot</td>
<td>Beta 1.0</td>
<td>27 Kb + 163 Kb of QASE</td>
<td>177 Kb</td>
</tr>
</tbody>
</table>

Table 6.15 List of bots observed, including the size of source and binary files.

While all of bots have a similar size, the MyBot code is much smaller, it is true that is because it is using an API which has already implemented most of important functions; but if we take a look at the binary files we can appreciate that the MyBot agent takes much less space than the other ones.

The size of the MyBot project is also quite small, it takes up around 900 code lines, to be able to face combat situations, keep a balance getting the items and with his priorities, select the best weapon at every moment, and not be influenced by the environment he is moving through (since everything in the agent is dynamic, no waypoints).
7 DISCUSSION

Artificial potential fields is one AI technique used for goal-oriented navigation methods in real robots, and it was proved effective to solve obstacle-avoidance as well as path-finding in dynamic environments [8], [10], [11]. The purpose of this thesis was to apply this technique to a real-time virtual environment like Quake 2 multiplayer game, and with the peculiarity of using several goals instead of one single.

The good point about this technique is that is very scalable, it can be applied to games without taking care of the structure the world has or the items, of course changing the methods called from within the AI method, but still the heart of the matter is the same and the agent will act the same way.

4 hypotheses were stated in this study, and so several experiments were run to check their veracity, in next section I will discuss about the results of these studies.

7.1 Limitations

There have been only a couple of limitations when it came to the programming of the agent; the first one was the impossibility of accessing to the map structure. That forced the programmer to implement a basic navigation system, which made things, like recognise a ladder or a box in the map, more difficult. Even if we are using the potential fields which is a dynamic-navigation technique, there were some times when recognising a structure like an elevator, ladder, doors, etc… could have been really helpful.

The second and last limitation was the way Quake 2 manages the vectors of movement and firing. Quake 2 mix those vectors making the isolation really difficult, which caused that moving in one direction and firing into another was not possible, still as the agent follows his enemies when they are in straight line of vision, the direction of the gun is well set. The isolation of these vectors is possible, and it will be included as future work.

7.2 Discussion of results

As explained in the previous section, 4 tests were run on the agent. First two were related to the proper potential field algorithm and the actual heart of the thesis, proving how the agent tends to balance taking into account the environmental factors and the entities in the world.

The first test was run to prove that the environmental factors like health or armor levels of the agent changes the priorities he has for the decision taking. First we calculated the potentials without taking into account environmental factors in Table 6.3 and later we developed another table (Table 6.5) proposing different combination of environmental factors and showing the actions that the agent would take.

The result of the experiment was a success, we proved that when the agent has a lack of health or armor he goes straight to take items of this kind, and as long as he gets back into a normal state, the rest of the items are interesting for him again. This really
makes the agent to take care of himself, making him a good survivor. We also saw in Table 6.5 that the HealingPotential has a higher weight for the agent than the ArmorPotential (the agent can play with no level of armor but not without any health).

The second experiment belongs basically to the same type of experiment as the first one. One new environment with only ammo items was created and several combinations of weapons and ammo levels were applied to the agent to see how his decisions changed.

The results were also what we expected, and the agent made a good performance, keeping his ammo levels balanced, and always taking care of the ammo for the weapon he was handling at the moment, this way he would not get out of ammo in a combat simulation with the consecutive change of weapon and being defenceless in front of his opponents.

These first two tests also proved how the agent takes care of group of items, instead of choosing the best goal in the world and go to take it. It is true that this kind of method makes the agent not to use the shortest path to take the goal, but in a multiple entity world like Quake 2 I considered the group-behaviour more efficient since a delayed decision increase the freedom to choose. And also in this way the agent is always in an advantage position, since it is in the proximities of several interesting items. He will take less time to take all the items than a single-object oriented agent and that can really make the difference between winning a combat or not.

The code was lighter than in other bots as we could saw in Table 6.15, and that makes the application to not take up too much space in the machine. But what was more interesting is that the agent can perform in a similar way to the other agents being programmed in much less code lines, not being the last one in a deathmatch play according to some preliminary studies performed. Actually we are talking about bots which have been reprogrammed several times and are not in the first version, so there are aspects in which they are better at the moment (like the path finding). Anyway it was a great result that the agent could face these bots.

At last we have test the computer charge for the processor, and therefore the time it takes to calculate the action of the agent, as the values in the environment and the size of the map grow. As we said in Section 6.5, the client and the server were run in the same machine, which made the results to be higher than they would probably be in a normal computer running the client part only.

In the experimental results of this test an improvement is proposed, concerning radius of action for the agent to not take care of all the items at the same time, but in radius, opening it and closing it dynamically according to the situation.

Note: The programmer did not found any map with more than 100 entities.
7.3 Discussion of methodology

The methodology followed in the experiments was quite appropriate for showing the veracity of the statements. The results were the expected, and the tests were varied, considering many factors in the developing of them.

Still there was one test more it would have been interesting to perform, and that can be added in the future, the interaction of the bot with the other ones in a deathmatch game (all against all). This way we could see also the performance of the agent in comparison with the best bots designed for Quake 2. The reason why this test was not included is that the agent has been programmed for showing how the artificial potential fields work together, and in some situations has not been programmed in a combat orientated way. We have the example to this in the fact that the enemies change their type of potential according to the life and armor of the agent. If the agent has a health level under 30 and does not have more armor than 100, the potential of the enemy is repulsive, making the agent to run away from him and look for health and armor items; in this time the agent does not fire to the enemy.

That kind of behaviour was created for making clearer how the artificial potential fields technique works, because in a real combat situation the last thing we have to do is to turn our back on the enemy.

The improvement of the bot is planned for future versions, researching in two fields, the path finding method and combat rules mainly.
8 CONCLUSION

The use of artificial potential fields in a real-time server client architecture has been proved to work well. The agent places himself always in the best positions of the map according to the objects he consider of interest and to his actual state, which made of him an entity who takes care of himself and tends to reach the perfect state (in which he has all weapons, all ammo, full level of armor and health and all the enemies killed). We got the results to this study in Sections 6.2 and 6.4.

By using simple algorithms we are able to program an agent performing all the functionalities of a player of Quake 2, of course it can be improved, but the great thing about this technique it is that once we understand it, is very easy to keep on adding potentials and functionalities to the behaviour (great scalability).

A bot developing the basic actions in Quake 2 really takes a little to be programmed if we use the potential fields, since we do not have to think in all the possible situations he will face, the performance of the bot depends on the obstacle and items he finds, no matter if we situate suddenly a new wall in the environment, the bot interacts with the environment in real time. This was supported by the test in Section 6.8 when we compared the size of the project with other bots already programmed.

The use of this technique together with another path finding method shows a great performance [8], and if we would apply it to a virtual environment such as Quake it would result in a agent who would be able to explore the whole world (no matter the complexity) and also being a good fighter (with the multiple potential fields).

In brief, the artificial potential fields is a technique that should really be considered in the industry of video-game as well as for robotic applications (here it is more known). The great flexibility is its weapon; with this technique many scenarios can be faced, from virtual environments like first person shooter games, to real robots developing services to humans, i.e. vacuum cleaner robots distributing themselves to clean different areas of a building, or exploring robots in unreachable environments for humans.
9 FUTURE WORK

An approach using multiple potential fields have been proposed for agent performance in Quake 2. The method is able to manage an agent through an environment he does not know, while taking care of himself by collecting entities in the world, and reacting to other agent’s attacks.

Still there are some points which would be very interesting to investigate and incorporate to the agent for a better performance:

- Further study on real-time obstacle avoidance. Although the agent works well with obstacles and several potential fields, he is not able to face very complicated environments where corridors and rooms contain the goals he is looking for. It would be necessary to implement an escaping method for when the agent cannot reach a goal. As some studies have shown [8] [10], the implementation of a follow-wall method works perfect in combination with the potential fields for facing this kind of situations. While the agent has not surround the obstacle he does not abandon the follow wall behavior, once there the potential fields acts again, and if he get stuck again then the follow-wall method shows up again.

- Also, some path-finding methods would be interesting for the performance of the agent on environments with floors at different height levels. This way the agent could be sensible to structures like ladders or elevators, and use them if the goal is unreachable from his level. This aspect is has a lot of research behind, since the agent should first explore the map, knowing how to recognize ladders, boxes, elevators, etc..., then memorize their situation, and the position he could reach through using them. Maybe some potential fields method could be used, i.e. placing a really big positive charge in the beginning of these structures when the goal is on another level. Actually this is one thing I haven’t seen in other bots through my study except if they are programmed in bot route support (waypoints).

- Coordination between bunch of agents using multiple potential fields could improve the way the bots explore the world [11] and also the organization of combat tactics (surrounding the enemy, blocking the exits, etc...). Here we have already the work done in Quake 3 by Marco Mamei and Franco Zambonelli [12], where they coordinate a set of agents for combat situations.

- Combat tactics using potential fields would be also interesting; applying a repulsive potential field to the bullets fired by the enemy the agent would be able to avoid the shots. Also the technique can be improved by applying the repulsive potential field when the enemy’s gun angles are pointing to the agent.

- Lastly there is another interesting research to make, regarding potential fields and first person shooter games: The creation of two different layers of
potential fields, one for the movement, and one for the firing of the agent. Researching into this method, the agent would be able to get the items he needs while firing at the enemy, and also as each layer has its own potentials, the forces are isolated, and there would be many more situations which the agent could face. One example would be a multiple enemy combat situation, firing to each enemy according their position, weapons, etc... while the agent can be in the meanwhile looking for the best gun to face that situation.
10 ACKNOWLEDGEMENTS

I wish to thank to all the people who has been supporting me through all the stages on the making of this thesis, people who was there on the good times but also on the difficult ones. Thanks to:

- Stefan, due to his good direction applying the technique to the agent and the hard work in the last weeks on the document.
- Martin, for helping me when I was having problems with the QASE API.
- My parents for supporting me when I wasn’t seeing this coming to an end.
- My girlfriend for being always there, even giving me a hand with the grammar of the document.
- All my friends, family and people who believed and supported me.
11 BIBLIOGRAPHY


[6] Bernard Gorman, Martin Fredriksson “The QASE specification 2.0” Department of interaction and system design, School of engineering, Blekinge institute of technology, Ronneby, Sweden. 2005


# APPENDIX A: QUAKE 2 ITEMS

## Weapons

<table>
<thead>
<tr>
<th>Weapons</th>
<th>Name</th>
<th>Description</th>
<th>Damage</th>
<th>Max Ammo Capacity</th>
<th>Importance for my agent(1-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blaster</td>
<td>It is the basic weapon, it never gets empty. It is slow and causes low damage.</td>
<td>15</td>
<td>∞</td>
<td>Not applicable. This weapon is always on the inventory of the agent.</td>
</tr>
<tr>
<td></td>
<td>Shotgun</td>
<td>Good weapon in the short distance, it reduces its efficiency in long distances. It uses shells.</td>
<td>4 per pellet, 12 pellets per shell</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Super-shotgun</td>
<td>The advanced version of the shotgun. It is deadly in the short distance and still bad in the long ones. Consumes the double of shells than the shotgun.</td>
<td>6 per pellet, 20 pellets per shell</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Machine-gun</td>
<td>Perfect weapon to attack a bunch of close enemies. Its backward movement makes the weapons</td>
<td>8 per bullet</td>
<td>200</td>
<td>6</td>
</tr>
<tr>
<td>Weapon</td>
<td>Description</td>
<td>Individual Damage</td>
<td>Blast Radius Damage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chain gun</td>
<td>Deadly weapon, fires the most amount of bullets per second but it also takes a while to initiate the fire. Not recommended for fast combat.</td>
<td>8 per bullet</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand grenade</td>
<td>Useful for melee situations. Big damage but very slow.</td>
<td></td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grenade launcher</td>
<td>This weapon launches grenades further and faster. The shot is parabolic.</td>
<td>Individual damage:120</td>
<td>Blast radius damage: 160 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rocket launcher</td>
<td>It uses rockets. Very useful against enemies in a med-far distance. Not use in close distances.</td>
<td>Individual damage: 100 + Random(0-20)</td>
<td>Blast radius damage: 120 points</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyper Blaster</td>
<td>This blaster fires way faster than the first one. It is very harmful which makes</td>
<td>15</td>
<td>200</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
this weapon a desirable one for the agent. It consumes cells. Rail gun
It is the sniper weapon of the game. It can kill one enemy with a single shot. It uses cells.

<table>
<thead>
<tr>
<th>Weapon</th>
<th>Description</th>
<th>Amount</th>
<th>Relevancy for the agent(1-6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rail gun</td>
<td>It is the sniper weapon of the game. It can kill one enemy with a single shot. It uses cells.</td>
<td>100</td>
<td>10 I gave this weapon the first position due to easiness the bot can aim with.</td>
</tr>
<tr>
<td>BFG-9000</td>
<td>The most dangerous weapon of the game. One single shot can kill several enemies at once. The shot is expansive, like a wave. Individual damage: 100/200 Blast radius damage: 500 points</td>
<td>200</td>
<td>8 I didn’t choose this weapon as the best, due to the amount of cells it consumes. It is very difficult to find in the map and is quite slow to fire in a fast combat situation.</td>
</tr>
</tbody>
</table>

Table A.1 Weapons in Quake 2

Ammo

<table>
<thead>
<tr>
<th>Ammunition</th>
<th>Name</th>
<th>Description</th>
<th>Amount</th>
<th>Relevancy for the agent(1-6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shells</td>
<td>Shells</td>
<td>Ammunition cartridges necessary for the shotgun and super shotgun.</td>
<td>It has 10 shells.</td>
<td>2</td>
</tr>
<tr>
<td>Cells</td>
<td>The energy cells are used by: the hyper blaster and the BFG-9000.</td>
<td>It has 50 cells.</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------</td>
<td>------------------</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>Bullets</td>
<td>Used by the Machine gun and the Chain gun.</td>
<td>It has 50 bullets inside.</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Grenades</td>
<td>They are hand grenades, to throw with the hand or use with the grenade launcher.</td>
<td>It has 5 grenades inside.</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Slugs</td>
<td>Used by the rail gun.</td>
<td>10 uranium slugs.</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Rockets</td>
<td>The rocket launcher uses this ammunition.</td>
<td>Each rack has 5 rockets.</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

*Table A.2 Ammo items in Quake 2*

**Armor**

<table>
<thead>
<tr>
<th>Armor</th>
<th>Name</th>
<th>Description</th>
<th>Protection(1-200)</th>
<th>Priority(1-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flak Jacket</td>
<td>It absorbs 30% of the damage done to the player.</td>
<td>Starts with 25 but picking up some armour shards it can reach 50.</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### Health

<table>
<thead>
<tr>
<th>Health</th>
<th>Name</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>First Aid</td>
<td>It gives 10 additional points to the health of the agent for each pick up.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Med Kit</td>
<td>It gives 25 additional points to the health of the agent for each pick up.</td>
<td>3</td>
</tr>
</tbody>
</table>

Table A.3 Armor items in Quake 2

**Combat Suit**
- It absorbs 60% of the damage done to the player.
- Starts with 50 but picking up some armour shards it can reach 100.

**Body Armour**
- It absorbs 80% of the damage done to the player.
- Starts with 100 but picking up some armour shards it can reach 200.

**Armour shard**
- It is not armor by itself, although it adds 2 armor points to the agent.
- It gives 2 points to the armour level of the agent. There is no limit for the agent to take items like this one.
- Not applicable. As it is always adding 2 points to the armor level of the agent, I gave this item a constant value, which will make the agent take it when passing by.
**Table A.4 Health items in Quake 2**

### Others

<table>
<thead>
<tr>
<th>Others</th>
<th>Name</th>
<th>Description</th>
<th>Are they relevant items for the agent (yes/ no)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Energy Armour</td>
<td>If the player is hit frontally he gets a 66% protection, otherwise a 33%. Each hit the armour absorbs one energy cell is consumed.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Silencer</td>
<td>Silences all the weapons when fired.</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Bandoleer</td>
<td>Increases the carrying capacity of the player. 50 more on every ammunition over the limit, except in the rail gun that increases in 25.</td>
<td>No</td>
</tr>
<tr>
<td>Image</td>
<td>Name</td>
<td>Description</td>
<td>Available</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>![Image](Heavy Pack)</td>
<td>Heavy Pack</td>
<td>Increases the carrying capacity of the player. 100 more on every ammunition over the limit, except in the rail gun that increases in 50.</td>
<td>No</td>
</tr>
<tr>
<td>![Image](Underwater breather)</td>
<td>Underwater breather</td>
<td>The player can be under the water for 30 seconds without drowning.</td>
<td>No</td>
</tr>
<tr>
<td><img src="Enviro-suit" alt="Image" /></td>
<td>Enviro-suit</td>
<td>The player can swim in hazardous liquids without getting injured.</td>
<td>No</td>
</tr>
<tr>
<td>![Image](Quad damage)</td>
<td>Quad damage</td>
<td>Increases the damage of the weapon the player carries in 4 times. It lasts for 30 seconds.</td>
<td>Yes</td>
</tr>
<tr>
<td><img src="Invulnerability" alt="Image" /></td>
<td>Invulnerability</td>
<td>Makes the player invulnerable for 30 seconds. But the armour still deteriorates with the hits.</td>
<td>Yes</td>
</tr>
<tr>
<td><img src="Adrenaline" alt="Image" /></td>
<td>Adrenaline</td>
<td>It raises the health of the agent till 100 progressively.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table A.5 Power-up items in Quake 2*
APPENDIX B: QUASE STRUCTURE

Fig B.1 QUASE structure