**A Project Report**

**On**

**Pacman Game**

**Bachelor of Technology**

**in Computer Science Engineering and Information Technology**

**by**

**Abhishek Chauhan (15csu007)**

**Abhishek Girdhar (15csu008)**

**Akhil Sharma (15csu017)**

****

**Department of CSE & IT**

**The NorthCap University**

**Gurugram**

**TABLE OF CONTENTS**

1. **Abstract………………………………………………..………...………..........................................3**
2. **Introduction........................................................................................................................................4**
3. **ProjectSetup.......................................................................................................................................6**
4. **The problem........................................................................................................................................7**

**4.1. Competition .................................................................................................................................7**

**4.2. Goal ..............................................................................................................................................7**

**4. Possible approach................................................................................................................................8**

**5.1. Learning method........................................................................................................................8**

**5.2. Symbolic approach.....................................................................................................................8**

**5. Result....................................................................................................................................................9**

**6. Conclusion.........................................................................................................................................10**

**7.References………………………………………………………………………………...................11**

**Abstract**

Artificial intelligence has been an important aspect of computer science to make a machine think like an human has been a diificult task for the developers but therse has been a rapid increase in the work of Ai in the past few years many games and machines have been developed which have Ai and can defeat a human mind too .This report discusses the Pacman game in Unity 3d. In this game there is a pacman who tries to protect itself from 4 ghosts Blinky, Clyde, Inky, Pinky. The report focus on how Unity can be a usefull tool to implement AI in games and make the game more attractive. The pacman game is the perfect example of Ai as the four ghost do team work to eat the pacman in a Maze. The game has many feature like lives for the pacman and the teleportation system through which it can teleport through the corners in the maze.

**1. Introduction**

In the early days of computer science, artificial intelligence was purely in the academic domain and even as computer games came on the scene in the 1970s and 80s, game AI was only an afterthought and could be very primitive indeed [Haahr 2010]. As computer technology has become more powerful and games more prolific however, much more complicated and effective AI techniques have crossed into the game domain [Haahr 2010]. It is with the above in mind, that this report discusses and implements an AI technique for a video game. Bizarrely, the game being used is one from the early days of computer games : Ms Pac-Man.

1. **The Problem**

* **The competition**

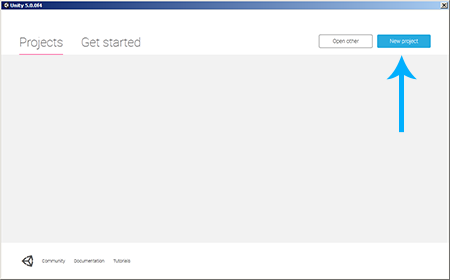
The rules for the league can be broken up into two categories, the competition rules which are those restricting the AI controller implementation and any failure to observe them will result in disqualification and the game rules which are enforced by the competition framework and determine the behaviour of the game world for example how points are awarded, levels progressed etc. The competition rules are : —AI controllers must finish initialisation within 5 seconds. —AI controllers are restricted to a 512MB memory footprint. —AI controllers must reside on a single thread. —Files may only be read from or written to if they are in the controller’s directory, are accessed only by the provided IO class and do not exceed 10MB. —Levels last for 3000 ticks of 40ms with the game advancing to the next level when time runs out. In such an event, the score that would have been awarded from the remaining pills is halved and given to the controller. —Each game can consist of a maximum of 16 levels. —Ghosts cannot reverse. The game rules are : —Ms Pac-Man begins the game with three lives which are deducted whenever she is caught by a ghost. Additional lives can be gained through the collection of 10,000 points and if all lives are depleted the game ends. —The game contains four mazes which are traversed in order until the game is complete or over. These mazes differ in terms of layout and pill placement. —Pills give 10 points, power pills 50 points and ghosts (if edible) give initially 200 points with this amount doubling for each additional ghost. —Ghosts become edible (and reverse direction) whenever Ms PacMan consumes a power pill. The time the ghost remains edible decreases with each level and if another power pill is eaten during this period, the score multiplier is reset. —If Ms Pac-Man loses a life, the ghosts are reset and she spawns in her initial position. —Once all pills are consumed or the time limit is up, the game progresses to the next level.

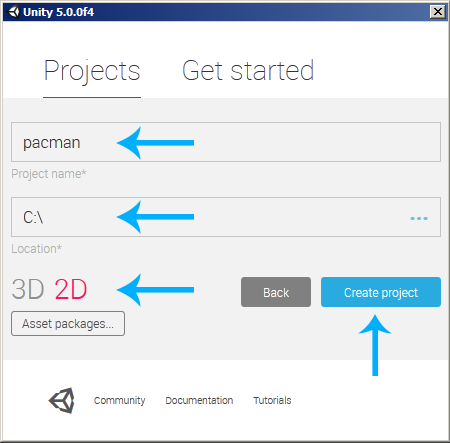
It will allow doctors to use the information provided by the patients in their profile. This information when combined with the current symptoms will help the doctors to advice an effective treatment with ease.

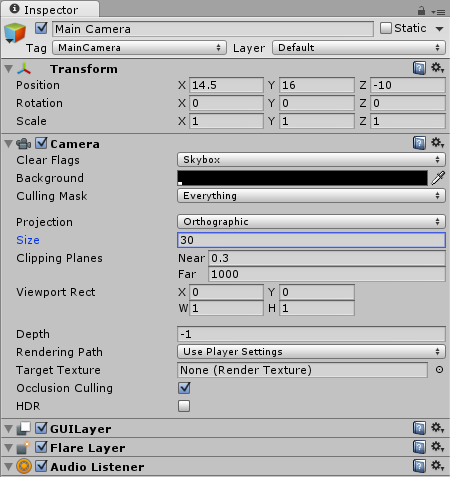
* **Goal**

This report implements just the Ms Pac-Man controller and not the ghosts controller. This is because traditionally the Pac-Man character is the one controlled by the player and as such it feels more natural to try and maximise its performance rather than hinder it.

## **2. Project Setup**

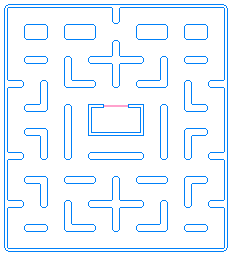
Let's get to it. We will start Unity and select New Project:  


We will name it pacman, select any location like C:\, select 2D and click Create Project:  


We will select the Main Camera in the Hierarchy and then set the Background Color to black. We will also adjust the Size and the Position like shown in the following image:  


## The Maze

### The Maze Sprite

Let's create the Pac-Man typical maze. We will draw one that is inspired by the original one, but not completely the same:  


**3. Possible approaches**

* **Learning method**

The first methods considered were those utilising some form of learning such as supervised learning or reinforcement learning techniques. Supervised techniques were considered where the AI would be trained against other opposing AIs using some form of annotated training data but this approach was dropped as there was no way of knowing which AIs our controller would face . As the behaviour of the ghosts could not be relied upon, the AI would not operate under the inductive learning assumption and as such techniques like supervised learning would not be ideal . Unsupervised reinforcement learning methods such as dynamic programming and temporal difference methods were then considered. These methods showed particular promise for the following reasons : —Training data could be obtained automatically from direct interaction with the game. —A “clearly defined adversary” is not necessary. —They work well in environments such as Ms Pac-Man’s where the search space could be quite large. It was for these above reasons that reinforcement learning methods were the first to be seriously considered. Temporal difference learning was given particular attention as it does not require a model of the environment [Luz 2012]. Temporal difference learning instead of storing a large database of game states and their appropriate responses, controls the agent through a neural network . The neural network is trained by exposing the agent or controller to a number of games and adjusting the weights of the network units to approximate the desired output . When the game is being played using this network, the state is passed into it and the network outputs an approximate action . This system allows a very large amount of game states to be encoded into a considerably smaller memory footprint. The performance of temporal difference methods has also been shown to be rather good, for example when used to implement a backgammon AI it approached the ability of some of the world’s best human players and shows significant potential for surpassing them . However, temporal difference learning or any other learning method was not used in the final implementation for a number of reasons. Reinforcement learning was introduced relatively late in the course and there was no introductory lab material for it so when it came to implementing it for this project there were many great difficulties. Difficulties included how best to represent the state to the learning method and how to determine values, rewards etc. Finally, it could not be determined definitively if enough training could be performed or if the quality of the training would be good enough to bring the controller’s performance above simpler methods. It was also not certain if many of the issues detailed in Tesauro [1992] could be resolved. Essentially, more tried and tested methods were used in the end and these will be detailed in the next section.

* **Symbolic approach**

After learning methods were abandoned, symbolic methods were then considered and in particular reactive agents. Reactive agents are relatively simple in comparison with the learning approaches discussed earlier. Reactive agents are stateless and operate as a hierarchical structure of condition → action rules meaning they react with a certain action for each state without considering previous or later states . The main appeal of this style of architecture over more advanced methods is that it is easier to understand, implement and test as the rules can be quite intuitive and the effects of changes can be seen immediately. As will become clear in section 4, a particular type of reactive agent known as a subsumption architecture was used. A subsumption architecture is produced by determining what problem we wish to solve with the agent, decomposing that problem into a set of tasks and implementing each individually as a separate layer . These independent layers provide a specific piece of functionality by themselves such as path finding, enemy evasion etc and by combining them together we can get a relatively advanced agent . The main benefits of this system are that specific parts of the agent’s behaviour can be implemented and tested independent of another part and new behaviours can be added without any major modification . Subsumption architectures also work well in systems with multiple and perhaps conflicting goals like Ms Pac-Man’s, for example Ms Pac-Man’s goal to avoid yet hunt ghosts. It is for the above reasons that a subsumption architecture reactive agent based on A.Brooks was ultimately used for this project’s final implementation

**4. Result**

The performance of the AI controller was evaluated across a range of parameter values with the average score for 100 trials being used as the performance metric. The two parameters for the AI controller hunt distance and evade distance were input across a range of 0 to 95 in increments of 5. Ideally, a larger range with a smaller increment step size should have been used but the time required to do so would have been prohibitive and even the relatively modest plots produced for this project took several minutes to output.

Using the results of the test discussed above, the graph in figure 1 was produced. The graph shows that the highest average score for the given range was around 5431 with the parameter values of 5 for evade distance and 75 for hunt distance. The full table of results can be seen in figure 3. To verify that 75 or thereabouts was the ideal hunt distance, another plot of results was produced but this time with a greater hunt distance range (0 to 195) which can be seen in figure 2. Figure 2 does indeed confirm that a distance of around 75 is ideal and that no great score difference can be observed with any higher distance value (most likely as we are reaching the width or height of the playing field).

**5. Conclusion**

In conclusion, this report has discussed the implementation of a simple AI controller for Ms Pac-Man from concept to evaluation. This report has discussed some of the varied approaches to creating a game AI from reinforcement learning to reactive agents in Unity 3d. It has detailed the construction of a purely reactive agent based upon the subsumption architecture from abstract architecture to final implementation and has assessed its performance across a range of parameter values. It has been shown that for a reactive Ms Pac-Man agent that risk taking is desirable and that hunting is one of the best ways to increase the average score. Finally some of the shortcomings of such an approach such as the inability to have long term goals or learn from mistakes have been highlighted and the alternative of finite state machines has been noted.

.