**SUPERIOR UNIVERSITY LAHORE**

|  |
| --- |
| Superior Logo |

**Faculty of Computer Science & IT**

**Game Programming**

**Project**

**Project Team**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Student Name** | **Student ID** | **Program** | **Contact Number** | **Email Address** |
| Asadullah Shahid | BCSM-F17-097 | BSCS |  | bcsm-f17-097@superior.edu.pk |
| Mustafa Manzoor | BCSM-F17-091 | BSCS |  | bcsm-f17-091@superior.edu.pk |
| Adeel | BCSM-F17-074 | BSCS |  | bcsm-f17-074@superior.edu.pk |

**Flappy Bird AI Final Project**

Flappy Bird is a 2013 mobile game, developed by Vietnam-based developer Dong Nguyen and published by GEARS Studios, a small independent game developer also based in Vietnam. The game is a side-scroller where the player controls a bird, attempting to fly between rows of green pipes without hitting them. The objective is to direct a flying bird, named Faby, who moves continuously to the right, between sets of Mario-like pipes. If the player touches the pipes, they lose. Faby briefly flaps upward each time that the player taps the screen; if the screen is not tapped, Faby falls because of gravity; each pair of pipes that he navigates between earns the player a single point.

**THE GOAL:**

Using learning algorithms we want to allow AI player to do a training set with different definitions of states (more details about those differences will discuss later), and learn the flappy bird world. After the training set we want the player to successfully continue playing as long as we want without any hit. We can examine the learning success using different parameters as number of training runs, learning rate, discount factor etc.

**THE APPROACH:**

We chose to use learning against search methods because of Flappy Bird is a continuous and dynamic game, the environment is not predefined, and we wanted to create agent that can take a first phase of learning and then play the game fluently. Q-learning algorithm is the most fit to our problem because the algorithm builds a policy and then can manage the states it will counter during the game. In order to make effective learning we needed to defined the state space of the problem. We found that there are 3 elements that we want to include in the state space: ❖ Vertical distance from the lower pipe. ❖ Horizontal distance from the lower pipe. ❖ Bird's velocity. We assumed that those 3 elements are the most important to describe a current state of the bird.

**Different approaches to state definitions Definition:**

GR – grid resolution; parameter we used to quantize the game grid. We used GR values of 1,2 and 4. We discuss more about it in the results part. Naïve state space The various elements in the state vector have the following ranges:

❖ Vertical: ~ 0 – 500

❖ Horizontal: ~ 0 – 300

❖ Velocity ~ -9 – 9

(units are in pixels and pixels per frame for the velocity) It is easy to see that the size of the state space is above 10 which means that using all the 6 states data will take too much time to learn, so we decided to use different approaches of the space quantization.

**Optimized state spaces :**

In order to decrease number of states we decided to use 4 different approaches:

1. Distance only – using only vertical and horizontal distances.

2. Horizontal relativity (before or between the pipes) and vertical distance.

3. Similar to approach 2 but with bird velocity.

4. Distance and an boolean indicator for positive/ non-positive velocity. All of the distances were tested with different GR values, and in order to decrease the size of the state space even more we defined the max and min values distances from pipes. Outside of this range we count it as the upper\lower bound of distance. Different approaches to state definitions Definition: GR – grid resolution; parameter we used to quantize the game grid. We used GR values of 1,2 and 4. We discuss more about it in the results part. **Naïve state space**

The various elements in the state vector have the following ranges:

❖ Vertical: ~ 0 – 500

❖ Horizontal: ~ 0 – 300

❖ Velocity ~ -9 – 9

(units are in pixels and pixels per frame for the velocity) It is easy to see that the size of the state space is above 10 which means that using all the 6 states data will take too much time to learn, so we decided to use different approaches of the space quantization.

**Optimized state spaces :**

In order to decrease number of states we decided to use 4 different approaches:

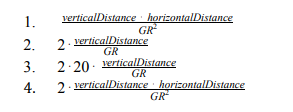
1. Distance only – using only vertical and horizontal distances.

2. Horizontal relativity (before or between the pipes) and vertical distance.

3. Similar to approach 2 but with bird velocity.

4. Distance and an boolean indicator for positive/ non-positive velocity. All of the distances were tested with different GR values, and in order to decrease the size of the state space even more we defined the max and min values distances from pipes. Outside of this range we count it as the upper\lower bound of distance.

After all those reductions, we got new size of state space for each approach (in correspondence to the approaches above):



**RESULTS :**

The number of options and parameter combinations to run the games is unlimited. Hence we ran a script that switched between different parameters and state spaces, every 3 minutes. We chose to present 3 runs for each state space - to show the relation between the parameters and performance. The graphs show the probability of the bird to crash into a pipe.

