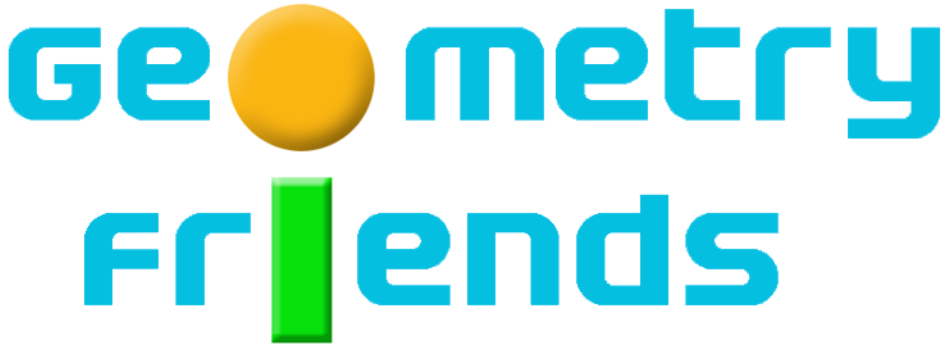
20.12.2015



Report of Term Project

BLG435E

Artificial Intelligence

2014/2015 Fall

Instructor: Sanem Sarıel Talay
Team Members: Muhammed Furkan Akyüz / 040100025 Hüseyin Erdoğan / 040100054

1 Introduction

In this project, we were worked to design a cooperative pair of AI agents for Geometry Friends puzzle game. Geometry Friends is a two player coorperative computer game. Players have to gather a set of diamonds in a given level in the least amount of time for completing level. Gravity and friction are also involved to game for simulated physics.

Geometry Friends game is a cooperative puzzle game developed by the GAIPS INESC-ID laboratory, using Microsoft's XNA framework. Game has two agent; circle and square and each agent has its own characteristics and play-style. In coorperative levels, most diamonds can only be obtained through the cooperationof both characters.

Square agent can change its shape to a horizontal or vertical rectangle without changing its total size. Square agent also can move left or right. There are black, yellow and green platforms in game and square agent can move on black and yellow platforms. Green platform obstructs movement of square agent.

Circle agent can change its radius and changing radius also effects mass of circle. Compared to square, circle also can jump. This agent can move on black and green platforms. Yellow obstructs movement of circle agent.

We installed Visual Studio 2013, .NET Framework 4, XNA Framework Redistributable 4.0 and XNA Game Studio 4.0. We tested and runned program in Visual Studio 2013 on Windows 8 and Visual Studio 2010 on Windows 8.1.

2 Analysis of Agents

2.1 Information About Agents

PEAS of Agents

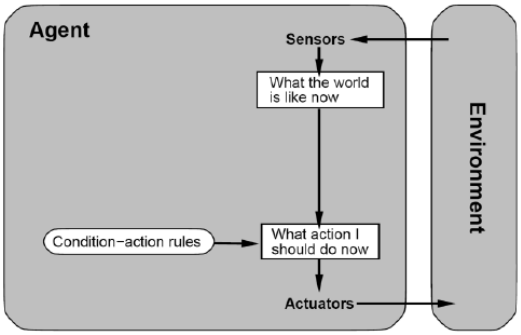
- Observable : Fully
- Deterministic : Stochastic
- Episodic : Sequential
- Static : Static
- Discrete : Continuous
- Agents : Multi

Type of Agents

In this project we design both agents as Simple Reflex Agents. In more complex levels we have to change the type of agent to Utility-based or Learning Agents.

We can show mechanism of agents as follows:

Simple Reflex Agents



2.2 Circle Agent

Actions:

- GROW
- JUMP
- ROLL_LEFT
- ROLL_RIGHT
- NO_ACTION

States:

- State C1 = If speed is higher then 80 AND difference between circele agent's x position and square agent's x position is between 100 and 150 pixel => NO_ACTION
- State C2 = If Square agent is under the diamond AND circle agent is under the square agent AND difference between circele agent's x position and square agent's x position is between 50 and -50 AND difference between circele agent's x position and square agent's x position is not between 25 and -25 => GROW
- State C3 = If Square agent is under the diamond AND circle agent is under the square agent AND difference between circele agent's x position and square agent's x position is between 50 and -50 AND difference between circele agent's x position and square agent's x position is between 25 and -25 => JUMP
- State C4 = If Square agent is under the diamond AND circle agent is under the square agent AND difference between circele agent's x position and square agent's x position is not between 100 and -100 AND circele agent's x position is lower than square agent's x position minus 200 AND circele agent's speed is more than 10 => ROLL_RIGHT

- State C5 = If Square agent is under the diamond AND circle agent is under the square agent AND difference between circele agent's x position and square agent's x position is not between 100 and -100 AND circele agent's x position is higher than square agent's x position plus 200 => ROLL_LEFT

- State C6 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is between 100 and -100 AND speed of circle agent is lower than 0 AND circle agent at left side of square agent => ROLL_RIGHT

- State C7 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is between 100 and -100 AND speed of circle agent is higher than 0 AND circle agent at right side of square agent => ROLL_LEFT

- State C8 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is between 100 and -100 AND square agent height is higher than 150 => JUMP

- State C9 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is not between 100 and -100 AND difference between circele agent's x position and square agent's x position is between 400 and -400 AND difference between circele agent's x position and square agent's x position is not between 250 and -250 AND speed of circle agent is higher than 80 => JUMP

- State C10 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is not between 100 and -100 AND circle agent's x position is lower then square agent's x position minus 200 AND circle agents speed is higher then 10 => ROLL_RIGHT

- State C11 = If Square agent is under the diamond AND circle agent is not under the square agent AND difference between circele agent's x position and square agent's x position is not between 100 and -100 AND circle agent's x position is higher then square agent's x position plus 200 AND circle agents speed is higher then 10 => ROLL_LEFT

- State C12 = If diamond is collected move to next diamond.

Initial State:

- If square agent standing under diamond and circle agent is at left side of square agent, initial state is C6.

- If square agent standing under diamond and circle agent is at right side of square agent, initial state is C7.

Goal State:

- Goal state of this agent is C12.

2.3 Square Agent

Actions:

- MORPH_DOWN
- MORPH_UP
- MOVE_LEFT
- MOVE_RIGHT
- NO_ACTION

States:

- State S1 = If speed is higher then 170 => NO_ACTION // This state is used for fasten up the speed of square
- State S2 = If square agent is at 20 pixel left of diamond AND encounters to a platform => MORPH_UP
- State S3 = If square agent is at 20 pixel left of diamond AND does not encounters to a platform => MOVE_RIGHT
- State S4 = If square agent is at 20 pixel right of diamond AND encounters to a platform => MORPH_UP
- State S5 = If square agent is at 20 pixel right of diamond AND does not encounters to a platform => MOVE_LEFT
- State S6 = If circle agent is at top of square agent => MORPH_UP
- State S7 = If square agent is not in any of this states => MORPH_DOWN
- State S8 = If diamond is collected move to next diamond.

Initial State:

- If first diamond is at right of square agent, initial state is S2.
- If first diamond is at left of square agent, initial state is S3.

Goal State:

- Goal state of this agent is S8.

3 Explenation of Codes

3.1 BallAgent.cs

In this topic, we will explain codes in BallAgent.cs file. Explanations of common codes and data structures are given from [the wiki web page of the competition](). On the other hand, there are some functions which we added to the file. We explained the data structures in the explanation of data structures topic.

We add a functions to this file. It determines motion plan of ball agent. The function gets two variables which are x and y coordinates of the current collebitle. Detailed explanation of code:

```
        private void go_to_coordinate(float x, float y)
        {
//If there is a yellow platform which is shorter than 100 units, the circle grows.
            if (control == false && circlePlatformsInfo[0] != 0 && circlePlatformsInfo[2] < 100)
            {
                SetAction(Moves.GROW);
                control = true;
            }
        //velocity control. if it is more than 80, no action.
            else if (Math.Abs(circleInfo[2]) > 80)
            {
                if (Math.Abs(circleInfo[0] - squareInfo[0]) < 100 || Math.Abs(circleInfo[0] -
squareInfo[0]) > 150)
                {
                    System.Diagnostics.Debug.Write("Hız Kontrolü" + "\n");
                    SetAction(Moves.NO_ACTION);
                }
            }

            //Kare elmasla aynı hizadaysa ve yuvarlak karenin altındaysa
            else if (Math.Abs(squareInfo[0] - x) < 100 && (circleInfo[1] - squareInfo[1]) > 100)
            {
                System.Diagnostics.Debug.Write("Kare Yuvarlakla aynı hizada degil" + "\n");
                //Yuvarlak kareyle aynı hizadaysa
                if (Math.Abs(circleInfo[0] - squareInfo[0]) < 50)
                {
                    if (Math.Abs(circleInfo[0] - squareInfo[0]) > 25)
                    {
                        System.Diagnostics.Debug.Write("grow dongusu \n");
                        SetAction(Moves.GROW);
                    }
                    else
                        SetAction(Moves.JUMP);
                }
                //Yuvarlak kareyle aynı hizada degilse
                else if (Math.Abs(circleInfo[0] - squareInfo[0]) > 100)
                {
                    if (circleInfo[0] < squareInfo[0] - 200 && Math.Abs(circleInfo[2]) > 10)
                    {
                        SetAction(Moves.ROLL_RIGHT);
                        if (Math.Abs(circleInfo[2]) > 100)
                        {
                            SetAction(Moves.NO_ACTION);
                        }
                    }
                    else if (circleInfo[0] > squareInfo[0] + 200)
```

```
                {
                    SetAction(Moves.ROLL_LEFT);
                    if (Math.Abs(circleInfo[2]) > 100)
                    {
                        SetAction(Moves.NO_ACTION);
                    }
                }
            }
        }

        //Kare elmasla aynı hizadaysa ve yuvarlak karenin altında degilse
        else if (Math.Abs(squareInfo[0] - x) < 100)
        {
            System.Diagnostics.Debug.Write("Kare Yuvarlakla aynı hizada " + "\n");
            //Yuvarlak kareyle aynı hizadaysa
            if (Math.Abs(circleInfo[0] - squareInfo[0]) < 100)
            {
                //karenin ortasının solunda ve sola dogru gidiyor ise
                if (circleInfo[2] < 0 && circleInfo[0] < squareInfo[0])
                    SetAction(Moves.ROLL_RIGHT);
                //karenin ortasının sagında ve saga dogru gidiyor ise
                else if (circleInfo[2] > 0 && circleInfo[0] > squareInfo[0])
                    SetAction(Moves.ROLL_LEFT);
                //yuvarlak karenin üstünde ve kare büyümüşse
                else if (squareInfo[4] > 150)
                    SetAction(Moves.JUMP);
                //Hiçbiriyse bişey yapma
                else
                    SetAction(Moves.NO_ACTION);
            }
            //Yuvarlak kareyle aynı hizada degilse
            else if (Math.Abs(circleInfo[0] - squareInfo[0]) > 100)
            {
                //Yuvarlak kareye yaklaştıysa ve hızı 15 ten büyükse
                if (Math.Abs(circleInfo[0] - squareInfo[0]) < 400
                   && Math.Abs(circleInfo[0] - squareInfo[0]) > 250 && Math.Abs(circleInfo[2]) >
15)
                    SetAction(Moves.JUMP);
                else if (circleInfo[0] < squareInfo[0] - 200 && Math.Abs(circleInfo[2]) > 10)
                    SetAction(Moves.ROLL_RIGHT);
                else if (circleInfo[0] > squareInfo[0] + 200 && Math.Abs(circleInfo[2]) > 10)
                    SetAction(Moves.ROLL_LEFT);
                else
                    SetAction(Moves.JUMP);
            }
        }
    }
```

3.2 SquareAgent.cs

In this topic, we will explain codes in SquareAgent.cs file. Explanations of common codes are given from [the wiki web page of the competition](). On the other hand, there are some functions which we added to the file. We explained the data structures in te explanation of data structures topic.

We add two functions to this file. One of them chooses a collebitle as goal. It calculates distances between collebitles and the square agent. It gets a variable which is number of collebitles. The collebitle which collebitle's distance is smaller than others is chosen as current collebitle. Detailed explanation of the code:

```
    private void find_current_collebitle(int collebits_number)
    {
        current_colleb = 0;
        for (int i = 0; i < collebits_number*2-2; i++)
        {
//In this if structure, it calculates the distances as (X1-Xcurrent)^2+(Y1-Ycurrent)^2
```

```
it calculates both distances and compare them. If i.(Ith) collebitle is closer than current collebitle,
current collebitle is changed as i. collebitle.
                if(Math.Sqrt(squareInfo[0]-collectiblesInfo[collebits_number])+Math.Sqrt(squareInfo[1]-
collectiblesInfo[collebits_number+1])<(Math.Sqrt(squareInfo[0]-
collectiblesInfo[i])+Math.Sqrt(squareInfo[1]-collectiblesInfo[i+1])))
                    current_colleb = i;
                i++;
            }
        }
```

Another function that we wrote, determines motion plan of square agent. The function gets two variables which are x and y coordinates of the current collebitle. Detailed explanation of code:

```
        private void go_to_coordinate(float x, float y)
        {
            //current action is morphing down
            SetAction(Moves.MORPH_DOWN);
             //in this if structure, if linear x velocity of square agent is more than 170, it does not
do any actions in order to limitiate the velocity
            if (Math.Abs(squareInfo[2]) > 170)
            {
                SetAction(Moves.NO_ACTION);
                lastAction = 0;
            }
//if there is a circlePlatform which is higher than 100 units, the agent morphs down.
            else if (circlePlatformsInfo[0] > 100 )
            {
        //and if distance between two agents less than 30 units, the square agent morphs down.
                if (Math.Abs(squareInfo[0] - circleInfo[0]) > 30)
                {
                    SetAction(Moves.MORPH_DOWN);
                }

            }
//if square agent is at the left side of the current collebitle,
            else if (squareInfo[0] < x - 20)
            {
        //and if the velocity of the agent is less than 5 units and it's last action is moving right or
left, it morphs up
                if (Math.Abs(squareInfo[2]) < 5 && (lastAction==6 || lastAction==5))
                {
                    SetAction(Moves.MORPH_UP);
                    lastAction = 7;
                }
        //if it is at the left side of the current collebitle, it moves right
                else
                {
                    SetAction(Moves.MOVE_RIGHT);
                    lastAction = 6;
                }
            }
//if square agent is at the right side of the current collebitle,

            else if (squareInfo[0] > x + 20)
            {
        //and if the velocity of the agent is less than 5 units and it's last action is moving right or
left, it morphs up
                if (Math.Abs(squareInfo[2]) < 5 && (lastAction==6 || lastAction==5))
                {
                    SetAction(Moves.MORPH_UP);
                    lastAction = 7;
                }
        //if it is at the left side of the current collebitle, it moves right
                else
                {
                    SetAction(Moves.MOVE_LEFT);
                    lastAction = 5;
                }
            }
//if the circle is on the square or up the square, and their centers are very close to each other,
square agent mophes up.
            else if (Math.Abs(squareInfo[0] - circleInfo[0]) < 20)
            {
                SetAction(Moves.MORPH_UP);
                lastAction = 7;
```

```
        }
    }
```

4 Explenation of Data Structures

The information in this topic is given from the wiki website of the compatition.

The Sensor Array Structure

The sensor array consists of 7 array structures, however it will shorten once the game starts (as some of the information is static and is not required to be passed again through the sensors).

Users should take note of two key methods in our Agent Interfaces:
the **Setup()** and **UpdateSensors()** methods.

4.1 Setup Method Parameters

The setup method is only called once, during the start-up phase of the game and will pass through its parameters all the sensor information.

4.1.1 int[] nI - Number Information Array

This array will contain the number of obstacles, the number of character specific platforms (Circle & Rectangle Platforms) and the total number of purple diamonds within the level.

Index - Information:

- 0 - Number of Obstacles
- 1 - Number of Square Platforms
- 2 - Number of Circle Platforms
- 3 - Number of Collectibles

4.1.2 float[] sI - Square Information Array

This array contains the current information on the rectangle agent, such as position (X and Y), velocity (X and Y) and its current height.

Index - Information:

- 0 - Square X Position
- 1 - Square Y Position
- 2 - Square X Velocity
- 3 - Square Y Velocity
- 4 - Square Height

4.1.3 float[] cI - Circle Information Array

This array contains the current information on the circle agent, such as position (X and Y), velocity (X and Y) and its current radius.

Index - Information:

- 0 - Circle X Position
- 1 - Circle Y Position
- 2 - Circle X Velocity
- 3 - Circle Y Velocity
- 4 - Circle Radius

4.1.4 float[] oI - Obstacle Information Array

This array contains all the information about the obstacles (default obstacles, not character specific obstacles) in the level, such as the center coordinates of the platform (X and Y) and the platform's height and width.

4.1.5 float[] sPI - Rectangle (Square) Platform Information Array

This array contains all the information about Rectangle specific platforms in the level, such as the center coordinates of the platform (X and Y) and the platform's height and width.

4.1.6 float[] cPI - Circle Platform Information Array

This array contains all the information about the Circle specific platforms in the level, such as the center coordinates of the platform (X and Y) and the platform's height and width.

4.1.7 float[] colI - Collectibles Information Array

This array contains the coordinates (center X and Y positions) of all the collectibles (purple diamonds) in the level.

4.2 UpdateSensors Method Parameters

The UpdateSensors method is called every few clicks and will pass through its parameters all the sensor information that is not static such as the collectibles number, the rectangle and circle agent positions and the collectibles position information.

4.2.1 int nC - Number of Collectibles

The current number of collectibles existent within the level.

4.2.2 float[] sI - Rectangle (Square) Information Array

This array contains the current information on the rectangle agent, such as position (X and Y), velocity (X and Y) and its current height.

Index - Information:

- 0 - Square X Position
- 1 - Square Y Position
- 2 - Square X Velocity
- 3 - Square Y Velocity
- 4 - Square Height

4.2.3 float[] cI - Circle Information Array

This array contains the current information on the circle agent, such as position (X and Y), velocity (X and Y) and its current radius.

Index - Information:

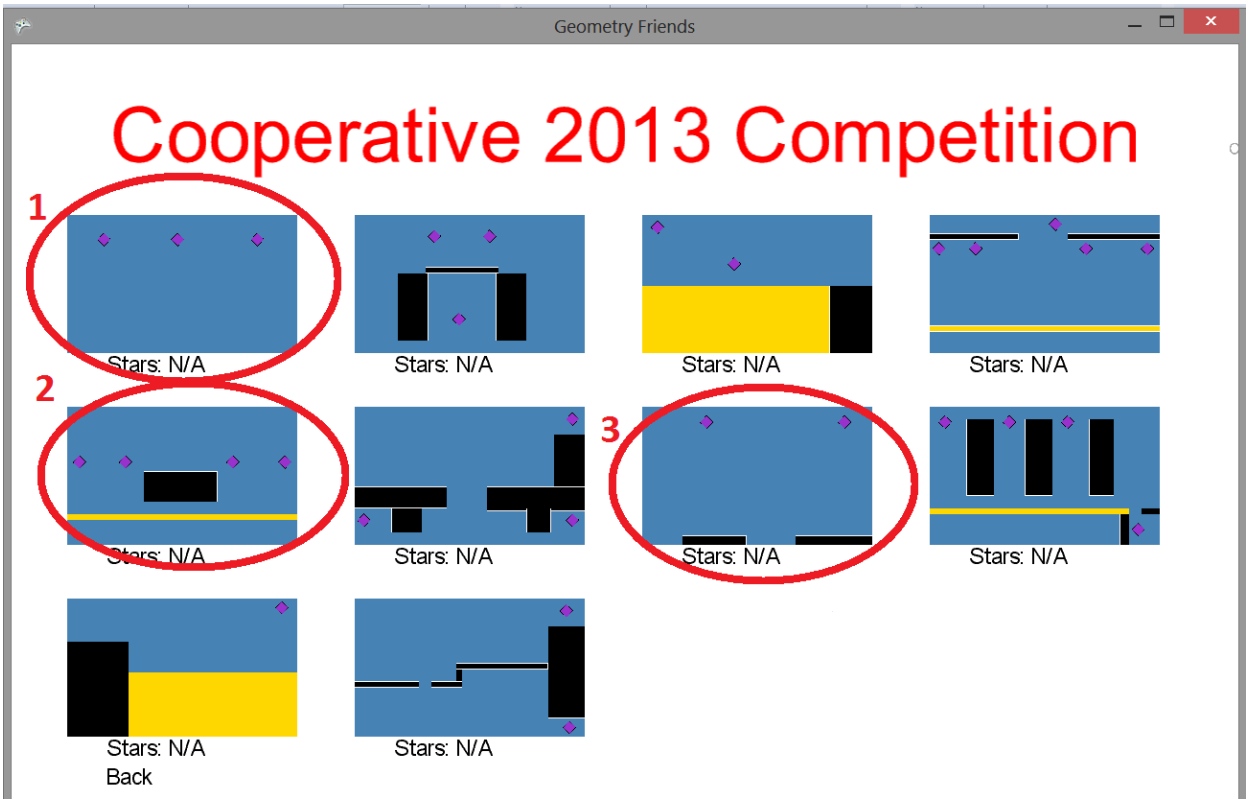
- 0 - Circle X Position
- 1 - Circle Y Position
- 2 - Circle X Velocity
- 3 - Circle Y Velocity
- 4 - Circle Radius

4.2.4 float[] coll - Collectibles Information Array

This array contains the coordinates (center X and Y positions) of all the collectibles (purple diamonds) in the level.

5 Completed Levels in Game

Levels that are in red circle are completed. Numbers that are right side of red circles are id's of levels that we completed. Our agents completes this levels most of the time between 70 and 120 second.



6 Group Member's Responsibilities

6.1 Muhammed Furkan Akyüz

- Circle Agent
- Square Agent (Most Part)
- Investigating features of Agents
- Learning Mechanism of Game

6.2 Hüseyin Erdoğan

- Circle Agent (Most Part)
- Square Agent
- Investigating features of Agents
- Learning Mechanism of Game
- Preparing Report