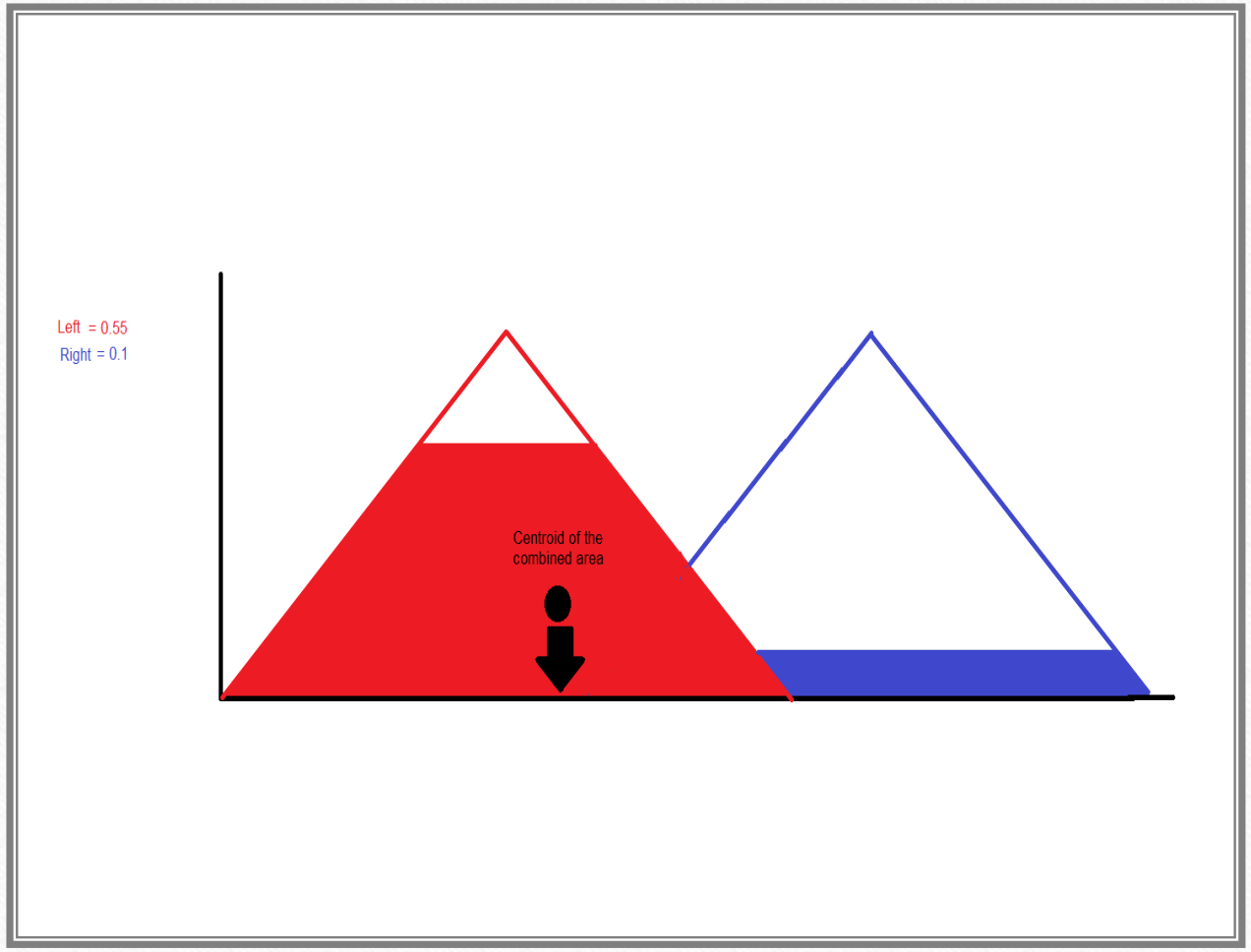
MAT301 Unit-1 Assignment

Cameron Wiggan 2001551

Introduction

- The goal of this application is to move a box object towards a goal object and avoid/move around an obstacle. Randomly repositioning the goal and obstacle objects once the agent has reached the goal and resetting the box object to the centre then letting it repeat for an allotted time. The application will also calculate a score for each session of the application reaching the goal object will add points and colliding with the obstacle will subtract points. Along with this it will count the number of times the box got collided with the obstacle and the average time it took for the box to reach the goal.
- For the AI technique used to accomplish this task Fuzzy logic was decided on.
- Fuzzy Logic was chosen due to its ability to deal in non-absolutes and ambiguous values which would be used to determine how much the object would have to move to reach its destination. With similar logic to avoid the obstacle.



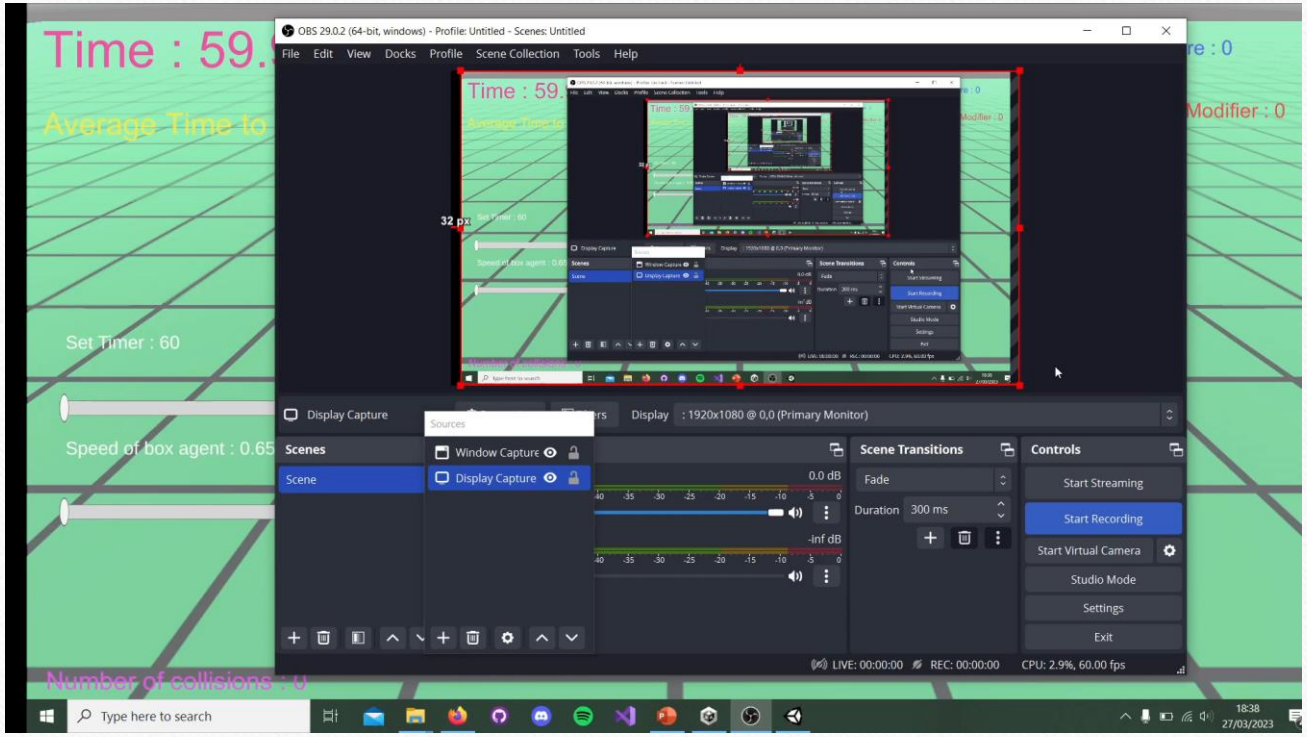
Fuzzy Logic Methodology

- Fuzzy logic is the Idea that most values are not absolute so a car might be going 'Quite fast' or need to turn 'a little to the left' rather than the traditional way computers represent values as being exact such as left or right and fast and slow.
- It achieves this by creating a graph of membership functions to demonstrate values such as left and right. The input value would then be used to determine how high of a percentage both the left and right values have. We then use a defuzzification method (For this example I will use centroid) to calculate the output value of the system.

Implemented Method

- The implementation used in my application used four engines in total to have the agent act appropriately. Two were for the goal object which operated in the x and the z-axis appropriately and the other two were for the obstacle object also in the x and z axis. Two engines were used for both the goal and the obstacle due to fuzzy logic only being able to act in a 1D space at a time due to it being graph based. So multiple engines had to be used to have the box agent move in multiple dimensions.
- These were then separately defuzzified and the results in the same axis were added together and added to the box agent in force in the appropriate axis.
- I decided to use the Fuzzy-logic-sharp library for unity developed by David Grupp (Link to download the library here : https://github.com/davidgrupp/Fuzzy-Logic-Sharp)

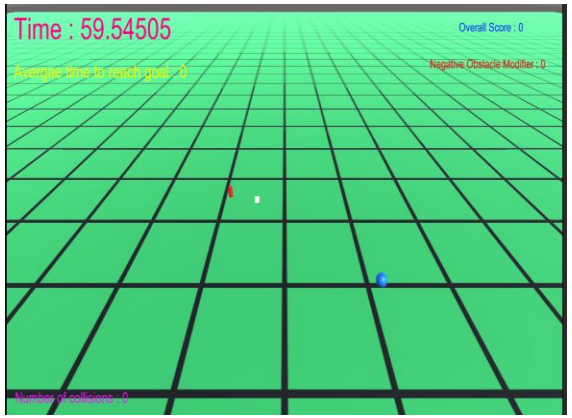
Application Demonstration



Code Demonstration

- As stated before in the Implemented method slide I used multiple engines to create the AI that moves in two dimensions, moving towards a goal and avoiding an obstacle.
- The Input sets for both the Goal engines consisted of the Direction (With their membership functions being the Right direction, Left Direction and none direction) and the Distance (With its member ship functions being the Right Distance, Left Distance and None Distance). However the exact values and positioning of each membership function changes depending on the engine it is attached to as the values that define the shape of the member change depending on the position of the attached object. As the graphs origin at value 0 is instead changed to be that of the object. For example the goal engine in the x-axis member ship functions would be defined by the positional x-value of the Goal object which would then subtract the original values for the logic.

```
//Sets the valuses for the following variables
Goal_x = Goal.transform.position.x;
Goal_z = Goal.transform.position.z;
diffrence_x = 200;
diffrence_z = 200;
avoidance_x = 6.25f;
avoidance_z = 6.25f;
Obstacle_X = obstacle_array[pos_in_arr].transform.position.x;
Obstacle_Z = obstacle_array[pos_in_arr].transform.position.z;

// Here we need to setup the Fuzzy Inference System
//Sets up the linguistic variable
distance_X = new LinguisticVariable("distanceX");
//Sets up the shapes for the fuzzy logic graph



//AddTrapezoid
var right_X = distance_X.MembershipFunctions.AddTrapezoid("right_X", Goal_x - diffrence_x, Goal_x - diffrence_x, Goal_x - 10, Goal_x - 1);
var none_X = distance_X.MembershipFunctions.AddTrapezoid("none_X", Goal_x - 10, Goal_x - 0.5, Goal_x + 0.5, Goal_x + 10);
var left_X = distance_X.MembershipFunctions.AddTrapezoid("left_X", Goal_x + 1, Goal_x + 10, Goal_x + diffrence_x, Goal_x + diffrence_x);
```




```
//Fixed update that is used to update physics of the box object
Unity Message | 0 references
void FixedUpdate()
{
    resultX = 0.0;
    resultZ = 0.0;

    avoid_result_X = 0.0;
    avoid_result_Z = 0.0;

    //Defuzzify the engines to get a representable value
    resultX = engineX.Defuzzify(new { distanceX = (double)this.transform.position.x });
    resultZ = engineZ.Defuzzify(new { distanceZ = (double)this.transform.position.z });

    //TrapezoidCoGDefuzzification;
    //CoGDefuzzification;
    //MoMDefuzzification;

    avoid_result_X = avoidEngineX.Defuzzify(new { Avoidance_distanceX = (double)this.transform
    avoid_result_Z = avoidEngineZ.Defuzzify(new { Avoidance_distanceZ = (double)this.transform

    //Combine the results
    complete_resultX = resultX + avoid_result_X;
    complete_resultZ = resultZ + avoid_result_Z;

    //Apply the complete result in force multiplied by the speed value to the box object
    Rigidbody rigidbody = GetComponent<Rigidbody>();
    rigidbody.AddForce(new Vector3((float)(complete_resultX * speed_), 0f, (float)(complete_re
}
```

Code Demonstration Continued

- The Input set for the two Obstacle engines consisted only of the Distance with the Membership Inputs Right and left (An Input set for None was not considered as at no point while the box agent is within range of the obstacle should it consider not moving away from it).

Code Demonstration Continued

- The rules for both Goal engines were Identical as the only difference was the axis in which they operate So for the application to operate correctly they need to be identical. These rules were:
  - IF the Distance is Right THEN Direction is Left
  - IF the Distance is Left THEN Direction is Right
  - IF the Distance is None THEN Direction is also None

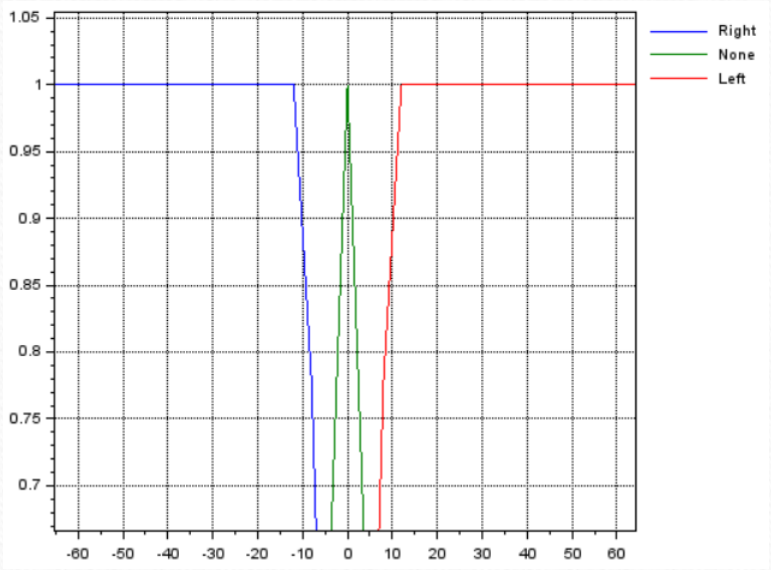
```
//Initialise the given engine
engineX = new FuzzyEngineFactory().Default();

//Sets up the variable rules for the engine
//Will determine the state of one linguistic variable based off the state of another
//So since distance represnts how far from the object the box is
//So if the box sits to the right of the goal it should move in the left direction
var rule1_X = Rule.If(distance_X.Is(right_X)).Then(direction_X.Is(left_direction_X));
var rule2_X = Rule.If(distance_X.Is(left_X)).Then(direction_X.Is(right_direction_X));
var rule3_X = Rule.If(distance_X.Is(none_X)).Then(direction_X.Is(none_direction_X));

//Add the rules to the engine
engineX.Rules.Add(rule1_X, rule2_X, rule3_X);
```

- The rules for both Obstacle engines were Identical as once the only difference was the axis in which they operate. These rules were:
  - IF the Distance is Obstacle Right THEN Direction is Move Left
  - IF the Distance is Obstacle Left THEN Direction is Move Right

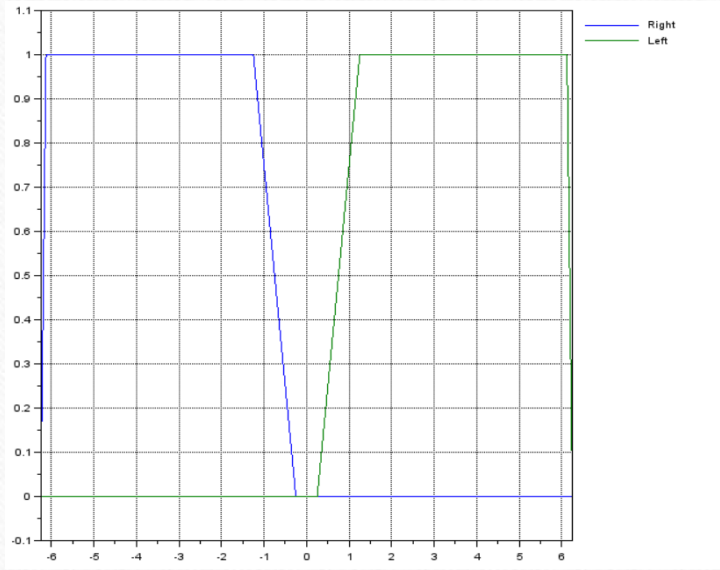
```
avoidEngineX = new FuzzyEngineFactory().Default();

//For the obstacle we want to do the opposite of the goal
//as in we want to move away from it
//so if the box is to the right of the obstacle keep moving to the right
var rule4_X = Rule.If(Avoidance_distance_X.Is(right_avoidance_distance_x)).Then(direction_X.Is(right_direction_X));
var rule5_X = Rule.If(Avoidance_distance_X.Is(left_avoidance_distance_x)).Then(direction_X.Is(left_direction_X));

avoidEngineX.Rules.Add(rule4_X, rule5_X);
```

Logic Graphs

Direction & Distance Logic



Avoidance Logic



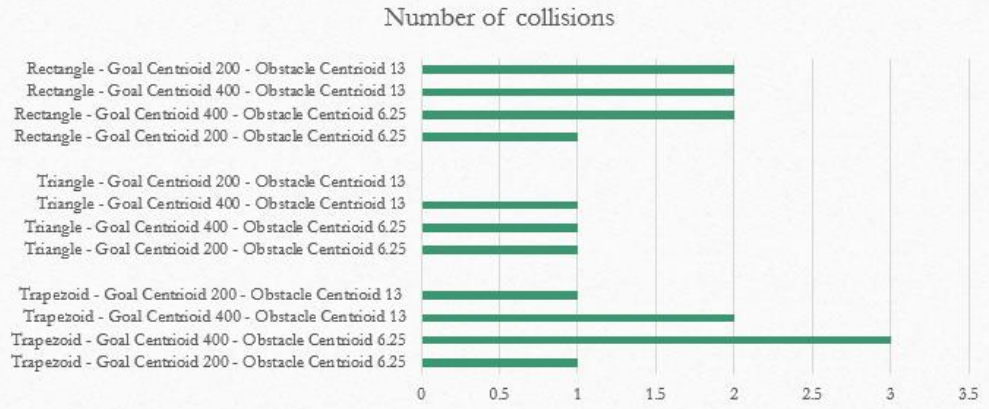
(Created using the software Scilab)

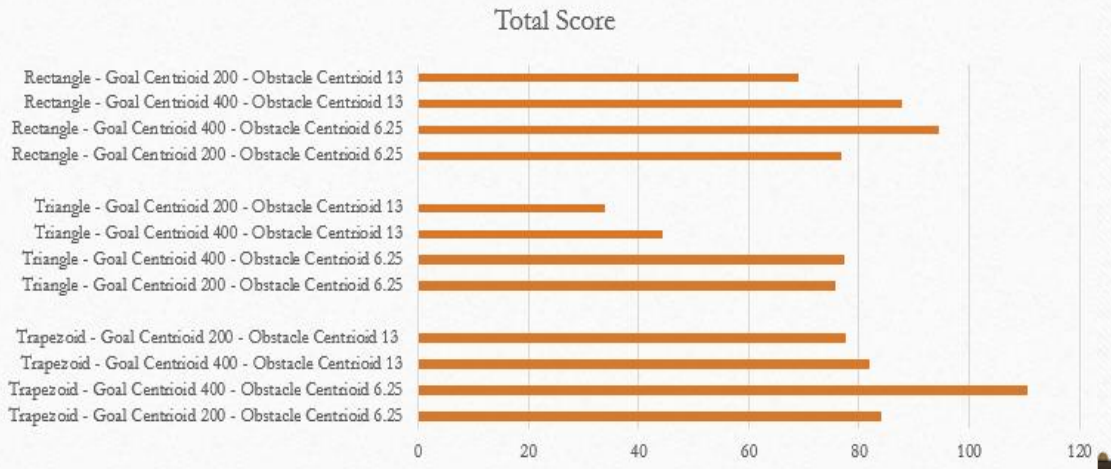
Improvements that could be made

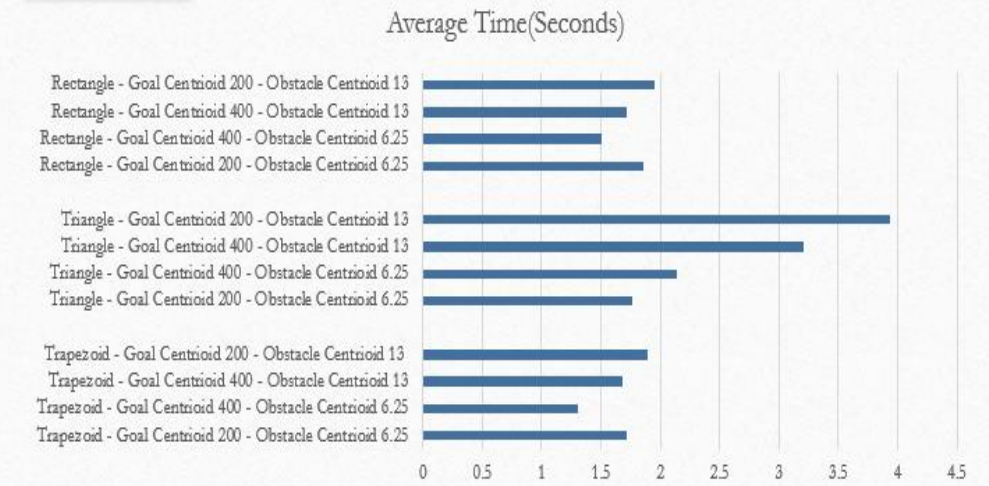
- Some Issues that were discovered during development and Improvements to fix them were :
  - Due to the random repositioning of objects some times the goal would spawn incredibly close the obstacle or the player which could interfere with the results. A way to have mitigated this would have been to normalise the distance of the goal from the origin of the box agent and divide the timing data by that.
  - Due to Fuzzy Logic not having an innate priority system if the goal object and the obstacle object were of a similar distance from the box agent this could cause it to become confused and not move as it becomes unsure of which direction to move.

Results

- For determining the effectiveness of the applications I decided to measure three factors the average time it took for the box object to reach the goal, the total score at the end of the playthrough (Adding two point five to the score when it hits the goal and subtracting one if the box agent hit the obstacle) and the number of times the agent collided with the obstacle per playthrough.
- I experimented with different shapes for the Fuzzy logic along with different sizes of graphs for both the goal logic and the obstacle logic.

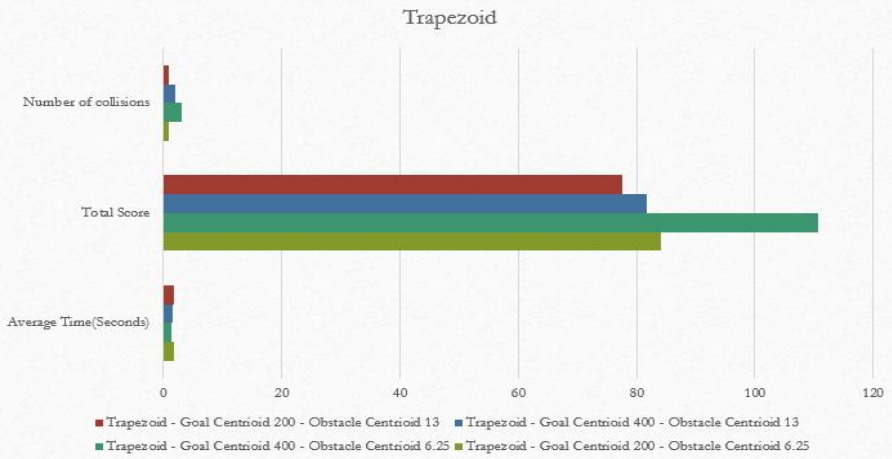


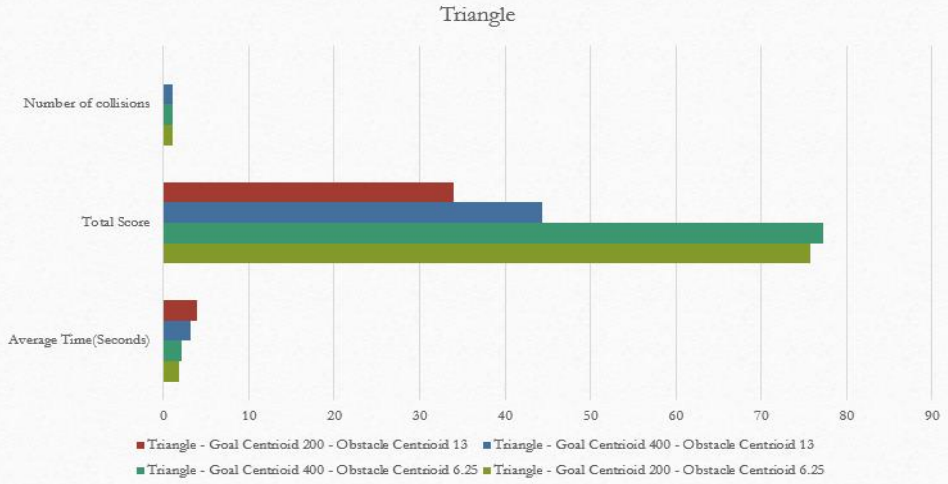


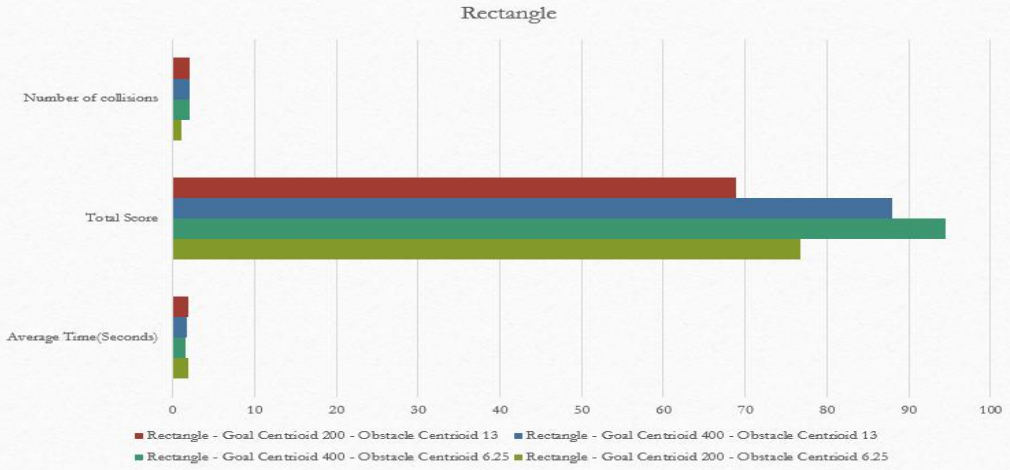


- These graphs represent the average of five runs of the experiment.
- As you can see the highest number of collisions belongs to Trapezoid – Goal Centroid 200(Distance from origin in both positive and negative directions for the fuzzy logic graph shape) – Obstacle Centroid 6.25 with zero collisions. With the triangle shape having a lower consistent value.
- However We can see a clear corelation between the Number of collisions and the Average Time it took for the agent to reach the goal.
- Aswell as the more obvious corelation between the lower the average time it took for the agent to get to the goal the higher the total amount of points collected by the end simply by the fact that the agent could actually collect more points if they spent less time moving towards the goal.

- As you can see here the settings with the highest score but also the lowest average time and lowest number of collisions is Triangle – Goal Centroid 200 – Obstacle 6.25.
- And the worse performing settings were Trapezoid – Goal Centroid 400 – obstacle 13.







Conclusion

- So In conclusion based on the data that I have gathered I believe that a larger radius for the goal logic results in better performance. Lowering the size in which the obstacle logic is active also helps in performance only coming into effect when agent needs to change course meaning it can take a more direct movement towards the goal.
- I have also discovered that using different shapes for the fuzzy logic also results in different paths the agent will move in as the areas in which the box agent knows whether to turn left, right or not at all changes.

References

- Grupp, David (2017), Fuzzy-Logic-Sharp available at : https://github.com/davidgrupp/Fuzzy-Logic-Sharp
- Scilab (1990) Available at : https://www.scilab.org/