Fuzzy Logic – Unity Worksheet

1. Download **Fuzzy Logic - Unity** from MyLearningSpace.

1. Open the Project in Unity. Alternatively, open the ‘SampleScene.unity’ scene file in **FuzzyLogic/Assets/Scenes**

1. Run the program, you should be able to click on the screen to move the box.

# Making a Fuzzy Inference System

Making a fuzzy inference system requires that we setup some membership functions, rules and defuzzification method. We are going to make sure that when we move the box, the system will move it back to the centre of the screen.

## Input Variable

1. Open the script attached to the FuzzyBox in the scene. You can do this by selecting the box in the editor and scrolling down in the inspector on the right hand side until you see the component called ‘Fuzzy Box (Script)’. Double click this to open it in Visual Studio.
   1. Alternatively, you can find the script in **Assets->Scripts->FuzzyBox**

1. The script has three functions inside it, **Start()**, **FixedUpdate()**, and **Update()**. You can ignore the **Update()** function for the time being. The Start() function is called once when the script starts at the beginning of the program, FixedUpdate is called at a specific interval (unlike update which is called as often as possible) so it’s very good for physics calculations.

At the top of the script add the following:

using FLS;

using FLS.Rules;

using FLS.MembershipFunctions;

This allows us to use the Library included with the project.

1. Create three private variables at the top of the FuzzyBox class. One IFuzzyEngine called ‘engine’, and two LinguisticVariable objects called ‘distance’ and ‘direction’.
   1. The IFuzzyEngine is what handles most of the FuzzyLogic, but we need to supply it with membership functions for the input and output as well as rules
   2. LinguisticVariable determine the input and output functions

1. In the Start() function add the following code:

distance = new LinguisticVariable("distance");

var right = distance.MembershipFunctions.AddTrapezoid("right", -50, -50, -5, -1);

var none = distance.MembershipFunctions.AddTrapezoid("none", -5, -0.5, 0.5, 5);

var left = distance.MembershipFunctions.AddTrapezoid("left", 1, 5, 50, 50);

* 1. You can plug this data into the **SCILAB Fuzzy Toolkit** to see what it looks like.

1. Implement the ‘direction’ LinguisticVariable so it is similar to the above code. You can reuse the shape of the values if you want.

1. Create a new instance of the FuzzyEngine:

engine = new FuzzyEngineFactory().Default();

1. Create the rules. Here, we want the system to move the box back to the centre of the screen. To do this, when the box is to the right of the centre, we need it to move left, when it is to the left, we move it right.

var rule1 = Rule.If(distance.Is(right)).Then(direction.Is(d\_left));

var rule2 = Rule.If(distance.Is(left)).Then(direction.Is(d\_right));

var rule3 = Rule.If(distance.Is(none)).Then(direction.Is(d\_none));

1. Finally, we add these rules to the FuzzyEngine:

engine.Rules.Add(rule1, rule2, rule3);

1. Now that the setup is done, we need to get the system to use our FuzzyEngine. In the **FixedUpdate()** function, we need to get the result of the Engine to apply a force that will move the box left or right. I’ve setup most of the code for you. Replace the following code:

double result = 0.0;

With the following:

double result = engine.Defuzzify(new { distance = (double)this.transform.position.x });

**What is happening?**

During the **FixedUpdate()** function, we generate a value which is used to apply a force to the box object. This is calculated by passing in the distance the box is from (0,0,0) and applying its coordinates through the membership function for the LinguisticVariable ‘distance’. The rules of the system are now applied to determine the membership of each of the output values, and the final data is defuzzified using the membership functions of the ‘direction’ variable.

Run the program again and see what happens now.

## Exercises

1. Change the values that define the Membership function of the LinguisticVariable. See if you can make change the behaviour of the box by editing these values.

1. Enhance the existing LinguisticVariables by making them more complex. For example, add ‘far right’ and ‘far left’ to the ‘distance’ LinguisticVariable. Do this for both inputs and the output.

1. Add another LinguisticVariable that describes the speed the box is currently travelling towards 0,0,0. Use this as another input variable and add it to the system.
   * 1. You will want to give the LinguisticVariable different values for its shape.
     2. Remember to update the rules, you will likely need to add more
     3. Make sure you pass the boxes current velocity into the defuzzify function

1. Copy the FuzzyLogicSharp library into last week’s Finite State Machine game. Implement a fuzzy inference system that controls the state change of the enemies in the game. See if you can get their detection range to be based on both the distance and speed the player is moving.

E.g. If the player is far away but moving towards an enemy quickly, then it will enter its CHASE state. (Simulates the agent seeing the player coming)

Or if the player is very close but is accelerating away, revert to the PATROL state (Simulates the agent realising it cannot catch the player)