ID2209 – Distributed Artificial Intelligence and Intelligent Agents

Final Project

Course Coordinator
Mihhail Matskin

Course Assistants
Mehdi Satei
Óttar Guðmundsson
Shatha Jaradat

Group 5
Mukesh Kottiliyil Muralidharan
Rui Qu

2019/1/12

<Intelligent Salesman>

In the final project, we were tasked with creating new agents in Unity based on a real-life scenario. People go shopping and choose whether to purchase or not, while salesman ought to improve their sales skill to promote deals.

How to run

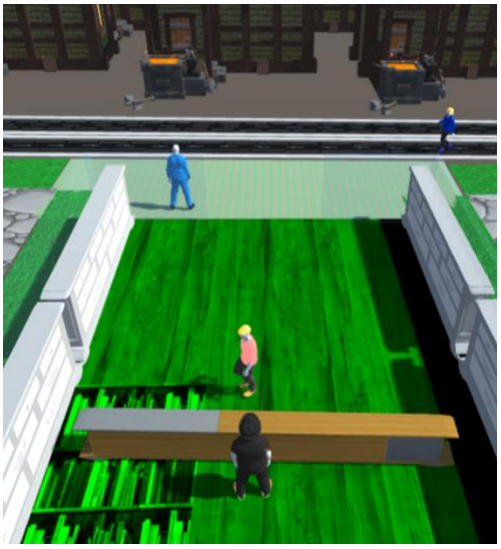
<Include a simple description for future readers who might want to run your code example, including some parameter tweaking if necessary>

After starting the attached application, as is shown in figure 1, press “s” key in the keyboard to initiate selling so that the customers will start moving to the store and process continues. If the customer buys the product, the floor will be highlighted with green color, otherwise with red color.

To restart the application, press the “r” key in the keyboard.

To quit the application, press the “Esc” button.

Figure 1: A screenshot of the solution.



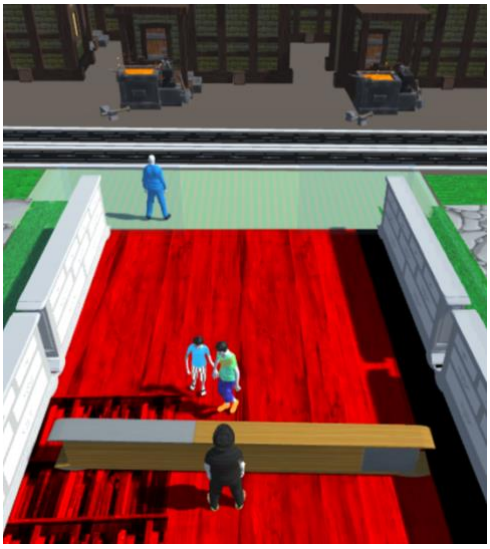


Figure 2: A screenshot of customer with&without purchasing.

Agent Salesman

The Agent inside the store tries to sell mechandise to Agent Customers. The interaction between these two agents is a randomly set array: for each state of Agent Customer, Agent Salesman gives him a feedback if the feedback meets the requirement of Customer, then deal is done, otherwise it‘s failed. As a result of randomness, the salesman without training will success but sometimes fail. What we cosider is to train salesman to make sure he understands which feedback he should provide to the Custmer and improve the overall sales.

Agent customer

The Agents hanging on the street will go inside the store only for purchasing.(Here we don‘t cosider window shopping, which means customers have the intension to purchase stuff in the store, but to purchase or not depends on the communication with salesman.) The maximum of Agent customers purchasing simultaneously in the store is 2. The one getting inside backward will wait in queue behind the former one.

Figure 3: A screenshot of Customer waiting in queue

We also create two hostile agents, which will fight with each other when they meet on the street. BUT when one of them has the intension to purchase, they won‘t fight. In case of this circumstance, an Agent call Security Gurad is created.

Agent Security Gurad

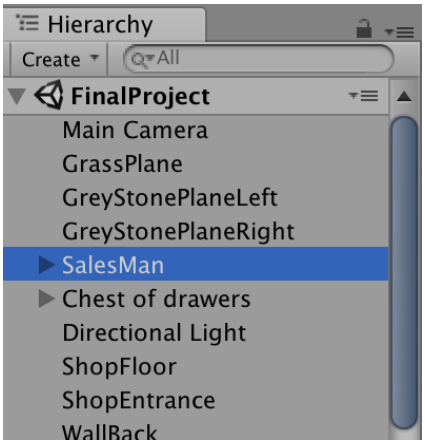
The Agent next to the entrance of store could detect the surroundings. When he finds someone is fighting, he‘ll come to them and stop them to ensure **Peace& Love,** which is aslo our tenet in our Assignment 1-3 .

Figure 3: A screenshot of SecurityGuard stopping two hostile customers

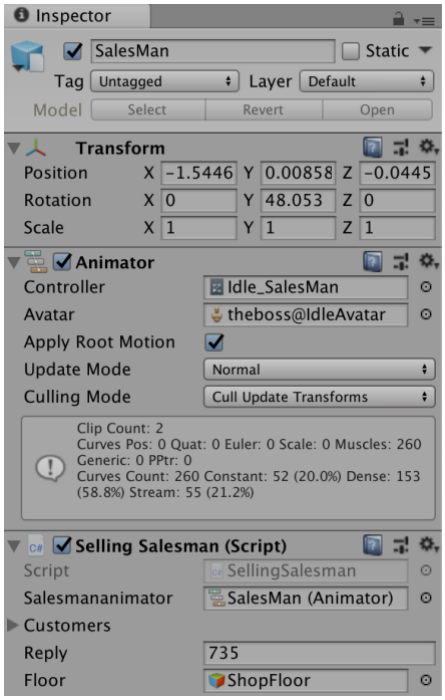
Introduction to Unity beginner level

We choose Unity in this project as it is not only a prevalent game engine but also the industry standard these days, and ML kit uses tensorflow machine learning tool in order to implement Reinforcement Learning.
Go https://unity3d.com/ for download.

The main difference between GAMA and Unity is that monitoring and editing in Unity could be executed simultaneously. Pre-programmed functions can be implemented easily and it gets much more splendid 3D animation effect than that in GAMA.



This Hierarchy above is what we create to make the overall visualised situation shown in figure 1. For each segment, there is a Inspector to mornitor their attributes and realise their functions. Here is an example of salesman. Scripts are writen in C# and will be given in the next part.



Implementation

Code of Agent Customer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class ShoppingRemy : MonoBehaviour {

    public GameObject salesman;
    SellingSalesman salesman_script;
    public NavMeshAgent agent;
    public Animator animator;
    public int ask;
    int question;
    public bool bought;
    public int InputX;
    public bool salesDone;

    // Use this for initialization
    void Start ()
    {
        ask = 0;
```

```
    bought = false;
    salesDone = false;
    // Get the game object animator
    animator = this.gameObject.GetComponent<Animator>();
    // Get the salesman script data.
    salesman_script = salesman.GetComponent<SellingSalesman>();
    // Signaling end of customer boughting interest when true.
    salesDone = false;
}

// Update is called once per frame
void FixedUpdate ()
{
    // Intiate the sales by pressing the "s" key.
    if (Input.GetKey("s"))
    {
        salesDone = false;
        bought = false;
        ask = 0;
        // Get a new question for new sale.
        question = Random.Range(1, 1000);
        // Make the agent moves to the sales man
        agent.SetDestination(new Vector3(0.07f, 0.0f, 1.20f));
        // Change the animation to Talking
        InputX = -1;
    }
    // Check if destination is reached.
    if ((Vector3.Distance(transform.position, new Vector3(0.07f, 0.0f, 1.0f)) <= 0.5f) && !salesDone)
    {
        // Stop agent navigation
        agent.isStopped = true;
        // Set the animator selection input.
        animator.SetFloat("InputX", InputX);
        // Initiate agent communication to Salesman
        CommunicateToSalesman();
    }
}

// Custom function for customer communication.
void CommunicateToSalesman()
```

```
{
    // Ask a question to salesman
    ask = question;
    Debug.Log("Ask" + ask);
    // Check for salesman's relpy. If same as question, sales is salesDone.
    if (salesman_script.reply == ask)
    {
        SalesDone();
    }
    else
    {
        // Call the function to wait.
        StartCoroutine(Waiting());
    }
}

void SalesDone()
{
    bought = true;
    salesDone = true;
    ask = 0;
    // Resume agent navigation
    agent.isStopped = false;
    // Make the agent move to the normal walking road
    agent.SetDestination(new Vector3(-6.5f, 0.0f, 33f));
    // Change the animation to Walking.
    InputX = 0;
    // Set the animator selection input.
    animator.SetFloat("InputX", InputX);
}

void SalesCancelled()
{
    salesDone = true;
    ask = 0;
    // Resume agent navigation
    agent.isStopped = false;
    // Make the agent move to the normal walking road
    agent.SetDestination(new Vector3(-6.5f, 0.0f, 33f));
    // Change the animation to Walking.
```

```
        InputX = 0;
        // Set the animator selection input.
        animator.SetFloat("InputX", InputX);
    }
    IEnumerator Waiting()
    {
        // Wait for some time to get the response from salesman.
        yield return new WaitForSeconds(5.0f);
        SalesCancelled();
    }
}
```

Code of Agent Salesman

```
    // Use this for initialization
    void Start () {
        // Init
        changed = false;
        // Get the animator
        salesmananimator = this.gameObject.GetComponent<Animator>();
        // Get the list of customer objects in the scene
        foreach(Transform child in transform)
        {
            customers.Add(child.gameObject);
        }
        // Get the floor renderer
        floorRenderer = floor.GetComponent<Renderer>();
        floorColor = floorRenderer.material.color;
        // Get the object handles for each customer.
        stefani = customers.Find((obj) => obj.name == "Stefani");
        stefani_script = stefani.GetComponent<ShoppingStefani>();
        remy = customers.Find((obj) => obj.name == "Remy");
        remy_script = remy.GetComponent<ShoppingRemy>();
        shae = customers.Find((obj) => obj.name == "Shae");
        shae_script = shae.GetComponent<ShoppingShae>();
        malcom = customers.Find((obj) => obj.name == "Malcom");
        malcom_script = malcom.GetComponent<ShoppingMalcom>();
```

```
    jasper = customers.Find((obj) => obj.name == "Jasper");
    jasper_script = jasper.GetComponent<ShoppingJasper>();
    regina = customers.Find((obj) => obj.name == "Regina");
    regina_script = regina.GetComponent<ShoppingRegina>();
    liam = customers.Find((obj) => obj.name == "Liam");
    liam_script = liam.GetComponent<ShoppingLiam>();
}

// Update is called once per frame
void FixedUpdate () {
    // Change the animator to talking when the customer asks question.
    if (remy_script.ask != 0)
    {
        inputY = -1;
    }
    else
    {
        inputY = 0;
    }
    // Set the animation selection on the animator.
    salesmananimator.SetFloat("inputY", inputY);
    // Initiate agent communication to customers
    CommunicateToCustomer();
    // Quit unity application when "Esc" is pressed.
    if (Input.GetKey("escape"))
    {
        Application.Quit();
    }
    // Restart unity simulation when "r" is pressed.
    if (Input.GetKey("r"))
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}

// Custom function for salesman communication.
void CommunicateToCustomer ()
{
    if (remy_script.ask != 0)
    {
```

```
        //Listen and reply to Stefani
        reply = Random.Range(1,1000);
        Debug.Log("Reply" +reply);
        changed = false;
    }
    // Change the floor color when the sales is done.
    if (remy_script.salesDone && !changed)
    {
        if (remy_script.bought)
        {
            // Customer bought the product
            StartCoroutine(SalesDoneColor());
        }
        else
        {
            // Customer didnt buy the product.
            StartCoroutine(SalesCancelledColor());
        }
    }
}

// Change the floor color when the sales is successful.
IEnumerator SalesDoneColor()
{
    floorRenderer.material.color = Color.green;
    Debug.Log("SalesDone");
    yield return new WaitForSeconds(0.5f); // Wait for 0.5sec
    floorRenderer.material.color = floorColor;
    changed = true;
}

// Change the floor color when the sales is failed.
IEnumerator SalesCancelledColor()
{
    floorRenderer.material.color = Color.red;
    Debug.Log("Sales Cancelled");
    yield return new WaitForSeconds(0.5f); // Wait for 0.5sec
    floorRenderer.material.color = floorColor;
    changed = true;
}
```

```
}
```

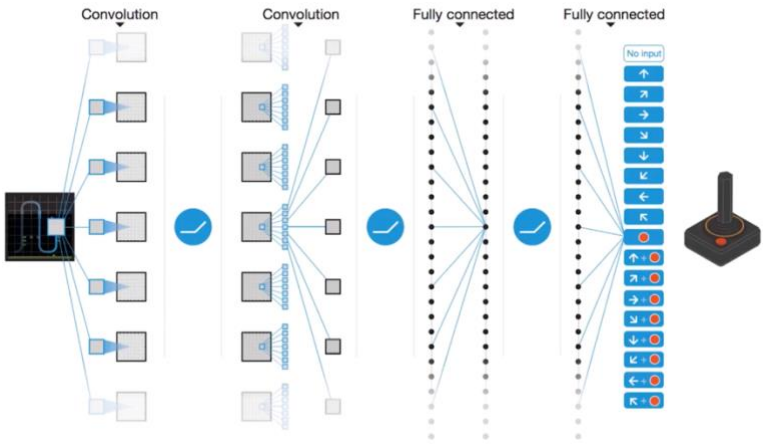
Challenge Reinforcement Learning

Machine Learning Tool KIT. Donwload: https://ole.unity.com/mlagentslive

Control of Agent in a game

Input: Game screen or states that agent has accessed to.

Output: Best move to take or button on the controller given the Input.



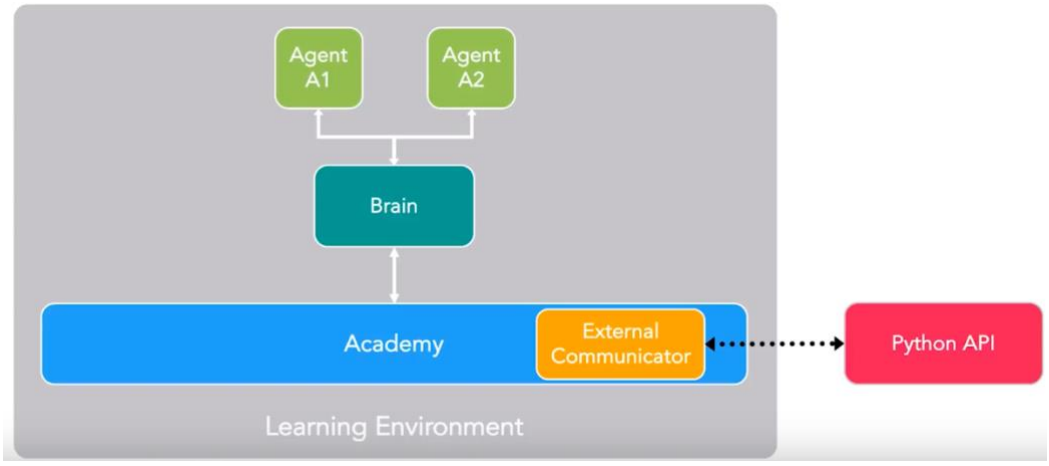
From observation to actions, we can take the traning method: Reinforcement Learning.

Learn through rewards, Trial-End-Error, Super-speed simulation,Agent becomes optimal at its task.

Learning signal: **Rewards,** scaled value,provided by agents,defined by designer. Imagine the situation when you train your pet.

+reward/-reward encourage/discourage the behavior they have until the desired property.

1.Create Env 2.Train brains 3.Embed brain



After training in the external Python API, Agent Salesman will know how to react correctly to fullfill Agent Customers requirement. The final result of the project is that no failure purchasing will come. The floor of the store will always be green after customers‘ purchasing.

Creative

<If you implemented the creative part, please fill this table with the relevant information>

Qualitative/Quantitative questions	*Answer*
Time spent on finding and developing the creative part	*20 hours in total*
In what area is your idea mostly related to…	*Reproduce real life scene*
On the scale of 1-5, how much did the extra feature add to the assignment?	*5*
On the scale of 1-5, how much did you learn from implementing your feature?	*4*

Discussion / Conclusion

Overall,a good assignment, can‘t wait to create a new version when I get back home to show my friends and family!