1 Code source Desktop

Listing 1 – Code source de CameraScript.cs

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

public class CameraScript : MonoBehaviour
{
    public Camera cam;
    public Button btnZoomIn;
    public Button btnZoomOut;

    private Vector3 defaultPosition;
    private Vector3 dragOrigin;
    private Vector3 cameraMinPosition;
    private Vector3 cameraMaxPosition;

    private float zoomStep = 1;
    private float minSize = 1;
    private float maxSize = 10;

    // Start is called before the first frame update
    void Start()
    {
        btnZoomIn.onClick.AddListener(ZoomIn);
        btnZoomOut.onClick.AddListener(ZoomOut);
        defaultPosition = cam.transform.position;
        cameraMinPosition = new Vector3(-8, -5, 0);
        cameraMaxPosition = new Vector3(8 ,5, 0);
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            dragOrigin = cam.ScreenToWorldPoint(Input.mousePosition);
        }

        if (Input.GetMouseButton(0))
        {
            Vector3 difference = dragOrigin -
                cam.ScreenToWorldPoint(Input.mousePosition);
            Vector3 newPosition = difference;

            if (newPosition.x + cam.transform.position.x <
                cameraMinPosition.x || newPosition.x +
                cam.transform.position.x > cameraMaxPosition.x)
            {
                difference.x = 0;
            }
```

```
            if (newPosition.y + cam.transform.position.y <
                cameraMinPosition.y || newPosition.y +
                cam.transform.position.y > cameraMaxPosition.y)
            {
                difference.y = 0;
            }

            cam.transform.position += difference;
        }

        if (Input.GetMouseButtonDown(1))
        {
            cam.transform.position = defaultPosition;
        }

    }

    private void ZoomIn()
    {
        float newSize = cam.orthographicSize - zoomStep;
        cam.orthographicSize = Mathf.Clamp(newSize, minSize, maxSize);
    }

    private void ZoomOut()
    {
        float newSize = cam.orthographicSize + zoomStep;
        cam.orthographicSize = Mathf.Clamp(newSize, minSize, maxSize);
    }
}
```

Listing 2 – Code source de ManagerScript.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using UnityEngine;
using UnityEngine.UI;

public class ManagerScript : MonoBehaviour
{
    public GameObject Camera;

    public GameObject hospital;
    public GameObject school;
    public GameObject store;
    public GameObject supermarket;
    public GameObject restaurant;
    public GameObject home;
    public GameObject company;
    public GameObject bus;

    public GameObject person;
```

```
    private List<Vector2> _sitesMin;
    private List<Vector2> _sitesMax;

    private List<GameObject> _persons;

    int read = 0;
    bool creationDone = false;

    // Start is called before the first frame update
    void Start()
    {
        _persons = new List<GameObject>();

        _sitesMin = new List<Vector2>();
        _sitesMax = new List<Vector2>();

        DataPopulation populationDatas =
            JsonUtility.FromJson<DataPopulation>(@"{
            ""NbPersons"":90,""IndexOfInfected"":[]}");
        DataSites sitesDatas = JsonUtility.FromJson<DataSites>(@"{
            ""NbHouse"":1000,""NbCompany"":1000,""NbHospital"":1000,""NbRestaurant""


        CreateSites(sitesDatas);
        CreatePopulation(populationDatas.NbPersons,
            populationDatas.IndexOfInfected);
    }
    public void SetIterationDatas(DataIteration iterationDatas)
    {
        read = 0;
        if (creationDone)
        {
            _persons.ForEach(p => {
                MovementScript pScript =
                    p.GetComponent<MovementScript>();
                int pIndex = pScript.index;
                int siteIndex = iterationDatas.PersonsNewSite[pIndex];

                if (siteIndex < _sitesMin.Count && siteIndex >= 0)
                    p.GetComponent<MovementScript>().SetTarget(_sitesMin[siteIndex],
                        _sitesMax[siteIndex], siteIndex);

                if (pScript.state !=
                    iterationDatas.PersonsNewState[pIndex])
                    pScript.SetState(iterationDatas.PersonsNewState[pIndex]);


                read++;
            });
        }
    }

    public void CreatePopulation(int nbPeople, List<int>
        indexOfInfected)
```

```
    {
        for (int i = 0; i < nbPeople; i++)
        {
            GameObject p = Instantiate(person, new Vector3(0,0, -5),
                Quaternion.identity);
            MovementScript pScript = p.GetComponent<MovementScript>();

            if (indexOfInfected.Contains(i))
                pScript.SetState(3);

            pScript.index = i;
            _persons.Add(p);
            read++;
        }
        creationDone = true;
    }

    public void CreateSites(DataSites sites)
    {
        //House T
        float topStartX = -9;
        float topStopX = 9;
        float topStartY = -5;
        float topUnitSize = (topStartX*-1 + topStopX) / sites.NbHouse;
        float topXL = topStartX;
        float topXR = topStartX;
        for (int i = 0; i < sites.NbHouse; i++)
        {
            topXR += topUnitSize;
            CreateSite(new Vector2(topXL, topStartY+1), new
                Vector2(topXR, topStartY));
            topXL += topUnitSize;
        }

        // Hospital R
        float rightStartX = 9;
        float rightStartY = -4;
        float rightStopY = 4;
        float rightUnitSize = (rightStartY + rightStopY * -1) /
            (sites.NbHospital + sites.NbSchool);
        float rightYT = rightStartY;
        float rightYB = rightStartY;
        for (int i = 0; i < sites.NbHospital; i++)
        {
            rightYB -= rightUnitSize;
            CreateSite(new Vector2(rightStartX, rightYT), new
                Vector2(rightStartX + 1, rightYB));
            rightYT -= rightUnitSize;
        }
        // School R
        for (int i = 0; i < sites.NbSchool; i++)
        {
            rightYB -= rightUnitSize;
            CreateSite(new Vector2(rightStartX, rightYT), new
                Vector2(rightStartX + 1, rightYB));
```

```
            rightYT -= rightUnitSize;
        }

        float leftStartX = -10;
        float leftStartY = -4;
        float leftStopY = 4;
        float leftUnitSize = (leftStartY + leftStopY * -1) /
            (sites.NbStore + sites.NbRestaurant + sites.NbSupermarket);
        float leftYT = leftStartY;
        float leftYB = leftStartY;
        // Store L
        for (int i = 0; i < sites.NbStore; i++)
        {
            leftYB -= leftUnitSize;
            CreateSite(new Vector2(leftStartX, leftYT), new
                Vector2(leftStartX + 1, leftYB));
            leftYT -= leftUnitSize;
        }
        // Restaurant L
        for (int i = 0; i < sites.NbRestaurant; i++)
        {
            leftYB -= leftUnitSize;
            CreateSite(new Vector2(leftStartX, leftYT), new
                Vector2(leftStartX + 1, leftYB));
            leftYT -= leftUnitSize;
        }
        // Supermarket L
        for (int i = 0; i < sites.NbSupermarket; i++)
        {
            leftYB -= leftUnitSize;
            CreateSite(new Vector2(leftStartX, leftYT), new
                Vector2(leftStartX + 1, leftYB));
            leftYT -= leftUnitSize;
        }

        float BottomStartX = -9;
        float BottomStopX = 9;
        float BottomStartY = 5;
        float bottomUnitSize = (BottomStartX * -1 + BottomStopX) /
            sites.NbCompany;
        float bottomXL = BottomStartX;
        float bottomXR = BottomStartX;
        // Company B
        for (int i = 0; i < sites.NbCompany; i++)
        {
            bottomXR += bottomUnitSize;
            CreateSite(new Vector2(bottomXL, BottomStartY- 1), new
                Vector2(bottomXR, BottomStartY));
            bottomXL += bottomUnitSize;
        }

        PositioningBuildings(sites);
    }

    private void CreateSite(Vector2 min, Vector2 max)
```

```
    {
        _sitesMin.Add(min);
        _sitesMax.Add(max);
    }

    private void PositioningBuildings(DataSites sites)
    {
        // Top
        float topScaleX = 9;
        float topPositionY = 4.5f;
        GameObject houses = home;
        houses.transform.position = new Vector3(0, topPositionY, 1);
        houses.transform.localScale = new Vector2(topScaleX*2, 1);
        Instantiate(home);

        // Left
        float sumLeftScale = (sites.NbStore + sites.NbSupermarket +
            sites.NbRestaurant);
        float leftScaleYStore = 8f / sumLeftScale * sites.NbStore;
        float leftStartY = 4f;
        float leftPositionX = -9.5f;
        GameObject stores = store;
        stores.transform.position = new Vector3(leftPositionX,
            leftStartY - (leftScaleYStore / 2), 1);
        stores.transform.localScale = new Vector2(1, leftScaleYStore);
        Instantiate(stores);

        float leftScaleYRestaurant = 8f / sumLeftScale *
            sites.NbRestaurant;
        float leftStartYRestaurant = leftStartY - leftScaleYStore;
        GameObject restaurants = restaurant;
        restaurants.transform.position = new Vector3(leftPositionX,
            leftStartYRestaurant - (leftScaleYRestaurant / 2), 1);
        restaurants.transform.localScale = new Vector2(1,
            leftScaleYRestaurant);
        Instantiate(restaurants);

        float leftScaleYSupermarket = 8f / sumLeftScale *
            sites.NbSupermarket;
        float leftStartYSupermarket = leftStartYRestaurant -
            leftScaleYRestaurant;
        Debug.Log(leftStartYSupermarket);
        GameObject supermarkets = supermarket;
        supermarkets.transform.position = new Vector3(leftPositionX,
            leftStartYSupermarket - (leftScaleYSupermarket / 2), 1);
        supermarkets.transform.localScale = new Vector2(1,
            leftScaleYSupermarket);
        Instantiate(supermarkets);

        // Right
        float sumRightScale = (sites.NbSchool + sites.NbHospital);
        float rightScaleYHospital = 8f / sumRightScale *
            sites.NbHospital;
        float rightStartY = 4f;
        float rightPositionX = 9.5f;
```

```
        GameObject hospitals = hospital;
        hospitals.transform.position = new Vector3(rightPositionX,
            rightStartY - (rightScaleYHospital / 2), 1);
        hospitals.transform.localScale = new Vector2(1,
            rightScaleYHospital);
        Instantiate(hospitals);

        float rightScaleYSchool = 8f / sumRightScale * sites.NbSchool;
        float rightStartYSchool = rightStartY - rightScaleYHospital;
        GameObject schools = school;
        schools.transform.position = new Vector3(rightPositionX,
            rightStartYSchool - (rightScaleYSchool / 2), 1);
        schools.transform.localScale = new Vector2(1,
            rightScaleYSchool);
        Instantiate(schools);

        // Bottom
        float bottomScaleX = 9;
        float bottomPositionY = -4.5f;
        GameObject companies = company;
        companies.transform.position = new Vector3(0, bottomPositionY,
            1);
        companies.transform.localScale = new Vector2(bottomScaleX * 2,
            1);
        Instantiate(companies);
    }
}
```

Listing 3 – Code source de MovementScript.cs

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

public class MovementScript : MonoBehaviour
{
    private const float SPEED = 5f;
    private readonly Color HEALTHY_COLOR = Color.blue;
    private readonly Color INFECTED_COLOR = Color.magenta;
    private readonly Color INFECTIOUS_COLOR = Color.red;
    private readonly Color IMMUNE_COLOR = Color.yellow;

    private Vector2 targetPosition;
    private float speed = SPEED;
    public SpriteRenderer sprite;
    public int targetIndex;
    public int index;
    public int state;

    // Update is called once per frame
    void Update()
    {
        transform.position = Vector2.MoveTowards(transform.position,
            targetPosition, speed);
        if (targetPosition.x == transform.position.x &&
```

```
            targetPosition.y == transform.position.y)
        {
            this.enabled = false;
        }
    }

    public void SetState(int state)
    {
        switch (state)
        {
            default:
            case 0:
                sprite.color = HEALTHY_COLOR;
                break;
            case 1:
                sprite.color = IMMUNE_COLOR;
                break;
            case 2:
                sprite.color = INFECTED_COLOR;
                break;
            case 3:
                sprite.color = INFECTIOUS_COLOR;
                break;
        }
        this.state = state;
    }

    /// <summary>
    /// Modifie le site cible de l'individu.
    /// </summary>
    /// <param name="target">Nouveau site cible.</param>
    public void SetTarget(Vector2 positionMin, Vector2 positionMax, int
        targetIndex)
    {
        this.targetIndex = targetIndex;
        this.enabled = true;
        targetPosition = FindPlaceInSite(positionMin, positionMax);
        speed = SPEED;
    }

    /// <summary>
    /// Choisis un point a l atoire dans le lieu actuelle et le choisis
        comme nouvelle cible.
    /// </summary>
    public Vector2 FindPlaceInSite(Vector2 positionMin, Vector2
        positionMax)
    {
        Vector2 newTarget = new Vector2(
                Random.Range(positionMin.x, positionMax.x),
                Random.Range(-positionMin.y, -positionMax.y)
            );

        return newTarget;
    }
}
```

Listing 4 – Code source de ScriptClient.cs

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.IO;
using System.IO.Pipes;
using System.Threading;
using System.Text;
using System;
using System.Security.Principal;
using System.Threading.Tasks;
using System.Collections.Generic;

public class ScriptClient : MonoBehaviour
{
    //public GameObject ChangingText;
    public NamedPipeClientStream pipeClient;
    public StreamString ss;

    DataPopulation populationDatas;
    DataSites sitesDatas;
    string resultIteration;

    string resultPopulation;
    string resultSites;
    DataIteration iterationDatas;

    private ManagerScript mngScript;

    // Use this for initialization
    void Start()
    {
        Debug.Log("Pipe Opening Process Started");
        pipeClient = new NamedPipeClientStream(".",
            "SimulationToUnity", PipeDirection.In,
            PipeOptions.Asynchronous);

        Debug.Log("Connecting to server...\n");

        ConnectToServer();

        mngScript = GetComponent<ManagerScript>();
        //string result = ss.ReadString();
    }

    // Update is called once per frame
    void Update()
    {
        if (!pipeClient.IsConnected)
        {
            ConnectToServer();
        }
    }
```

```
    private void ConnectToServer()
    {
        pipeClient.Connect();
        if (pipeClient.IsConnected)
        {
            ss = new StreamString(pipeClient);
            Thread.Sleep(250);
            ReadPipeData();
        }
    }

    private async void ReadPipeData()
    {

        string result = await ss.ReadStringAsync();
        string resultDataType = result.Split(' ')[0];

        switch (resultDataType)
        {
            default:
                Debug.Log("Error");
                break;
            case "Initialize":
                resultPopulation = result.Split(' ')[1];
                resultSites = result.Split(' ')[2];

                populationDatas =
                    JsonUtility.FromJson<DataPopulation>(resultPopulation);
                sitesDatas =
                    JsonUtility.FromJson<DataSites>(resultSites);

                mngScript.CreateSites(sitesDatas);
                mngScript.CreatePopulation(populationDatas.NbPersons,
                    populationDatas.IndexOfInfected);
                break;
            case "Iterate":
                resultIteration = result.Split(' ')[1];
                iterationDatas =
                    JsonUtility.FromJson<DataIteration>(resultIteration);

                mngScript.SetIterationDatas(iterationDatas);
                break;
        }

        ReadPipeData();
    }
}

[Serializable]
public class DataPopulation
{
    public int NbPersons;
    public List<int> IndexOfInfected;
}
```

```
[Serializable]
public class DataSites
{
    public int NbHouse;
    public int NbCompany;
    public int NbHospital;
    public int NbRestaurant;
    public int NbSchool;
    public int NbStore;
    public int NbSupermarket;
}

[Serializable]
public class DataIteration
{
    public List<int> PersonsNewSite;
    public List<int> PersonsNewState;
}

public class StreamString
{
    private BinaryReader stream;
    private UnicodeEncoding streamEncoding;

    public StreamString(Stream ioStream)
    {
        this.stream = new BinaryReader(ioStream);
        streamEncoding = new UnicodeEncoding();
    }

    public string ReadString()
    {
        int len;
        len = stream.ReadByte() * 256;
        len += stream.ReadByte();
        byte[] inBuffer = new byte[len];
        stream.Read(inBuffer, 0, len);

        string outString = streamEncoding.GetString(inBuffer);

        //ioStream.Flush();

        return outString;
    }

    public async Task<string> ReadStringAsync()
    {
        return await Task.Run(() =>
        {
            int len;
            len = stream.ReadByte() << 24;
            len = stream.ReadByte() << 16;
            len += stream.ReadByte() << 8;
            len += stream.ReadByte();
            byte[] inBuffer = new byte[len];
```

```
            stream.Read(inBuffer, 0, len);

            return streamEncoding.GetString(inBuffer);
        });
    }
}
```