PROJECT REPORT
PIXEL SIMULATION

April 2021

By
Notter Arnaud
arnaud.notter@epita.fr

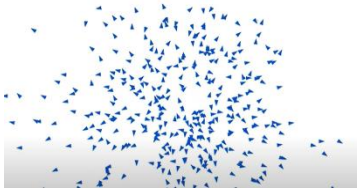
Summary

1. Introduction

What is the idea behind Pixel Simulation?

I am very interesting about the fouloscopy and I watched a lot of video about this subject. At this moment, I discover the work of Craig Reynold. It was one of the first to implement simulation of entity and tried to interact them with individual behaviour and not with global instructions. For example, he tried to implement the flight of birds. The flight of birds gives the impression of all birds are synchronised and that created a coordination movement.

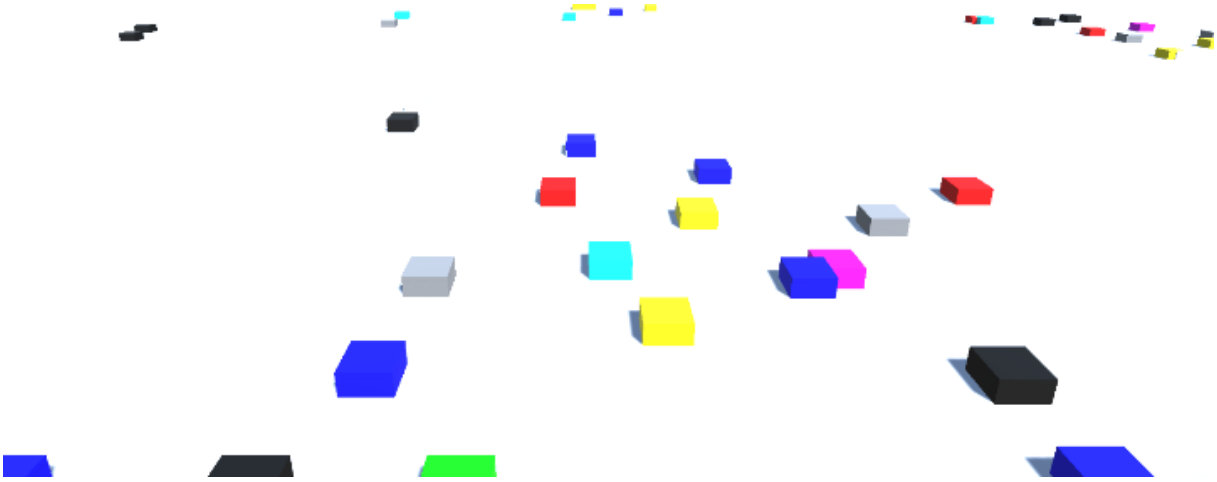
So, my project is to recreate this simulation with 2 dimensions. Let me explain more. The idea is to try to coordinate the movement of different entity without synchronize instruction. At the end, the movement of one is the same than the movement of another without connexion between them.



That is the main goal of this simulation for the last submission. In fact, we can go further. This simulation can be used to create the movement of an army in movie (cf. The lords of the Rings), and we can mix with an artificial intelligence to simulate the movement during a fight.

2. Unity

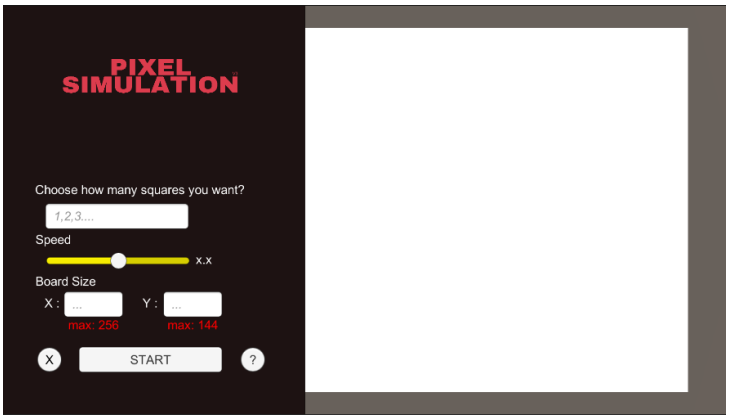
I have chosen to used Unity because I have already worked with it at the past. Indeed, I already created a game with Unity so, I knew the interface and how used this powerful software. Unity uses C# and that is really like Java for the way of thinking with objects.
To implement our simulation, the entities that we talk in the introduction will be represent by square and we called them “Agent”.



In this part, I will talk about the different panel create with Unity.

- *Launch panel*
  This panel is instantly launch when the user starts the application. The panel shows the name of the application with an animation.

- *Menu panel*

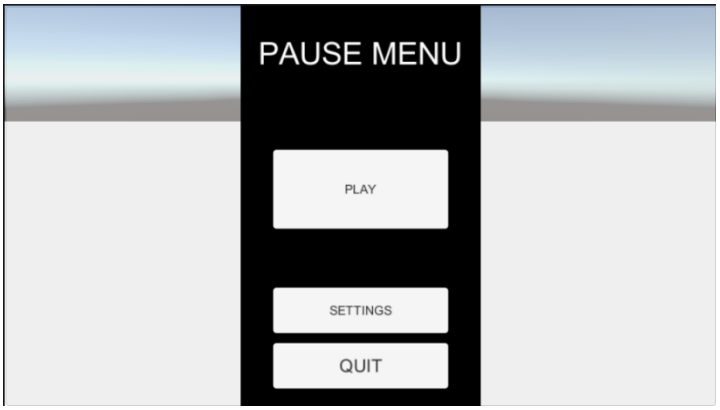


The menu panel allows to choose the different option and start a game. The different widget are: 6 text views, 3 input fields, 1 rolled bar and 3 buttons. With these 3 buttons, you can quit the application, start a simulation or show the help panel. In the background, you can see the plane how the simulation will be launch.

- *Add panel*

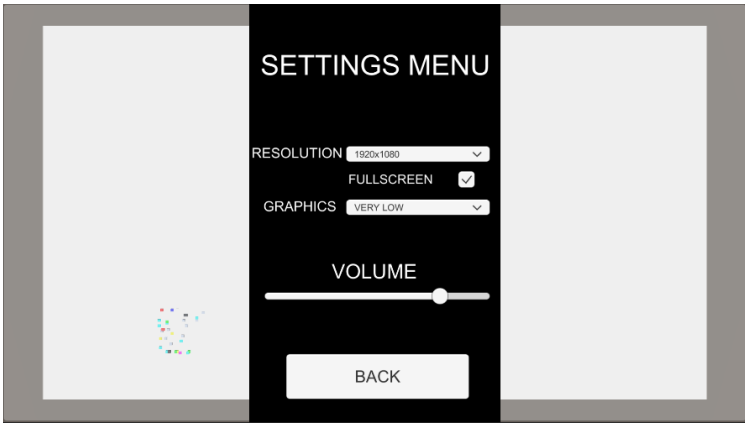
This panel appears when you push “A” during a simulation. You can add manually an Agent (the square). You need to choose an id (not really important), the position inside your board and the direction when you fill all this information, you can try to find an error

- *Pause panel*



This panel can be show when you push “space” during the simulation. You are three buttons: one to continue the simulation, one to show the option panel and one to quit the simulation.

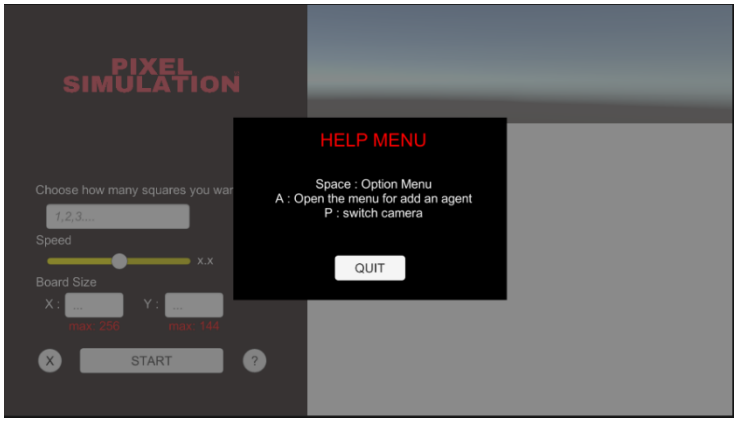
- *Option panel*



This panel allows to change different option like the resolution, the graphics and the volume.

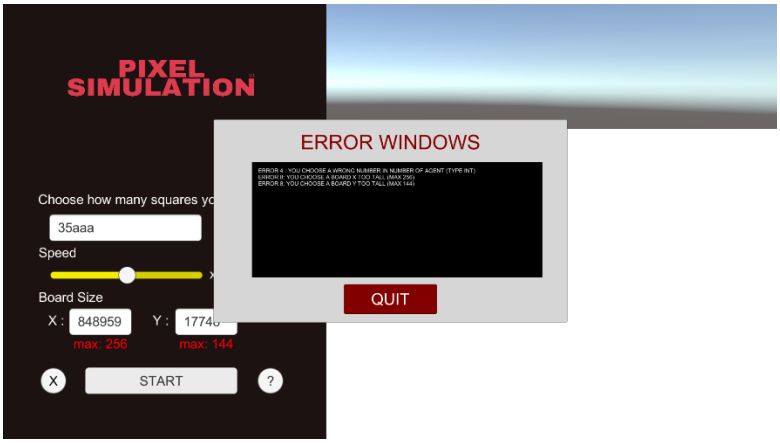
- *Help panel*

This panel shows the different control in the application.



- *Warning panel*

This panel shows the different error in the user's input.



3. Code C# - Class

On this part, we will describe the 10-class used in the project.

- *Agent*

```
public class Agent : MonoBehaviour
{
    //CREATE RANDOM NUMBER
    /* THE CLASS AGENT IS COMPOSED BY
     * Int id  = Use to identify the agent
     * Float posx and posy = define the position of agent in the board
     * Direction direction = is use to know the next position of the agent (the class direction
     * is explain on the other script)
     * The other paramater is used by the function of interaction
     */
    public int id;  Unchanged
    public float posx;  Unchanged
    public float posy;  Unchanged
    public Direction direction;  Serializable
    public GameObject Object;  Unchanged
    public Direction futurnewDirection;  Serializable
    //TO CREATE INTERACTION BETWEEN THE AGENT
    public bool IsInGroup;  Unchanged
    public GameObject Trigger1;  Unchanged
    public int HowManyInGroup;  Unchanged
```

There is three constructors for this class:

```
public Agent(int id, int posx, int posy, Direction direction, GameObject GameObject, Random random)
public Agent(int id,int posx,int posy,float speed, GameObject Object, Random aleatoire) {
public Agent(int id, GameObject Object, float speed, Board board, Random aleatoire){
```

And one function used to move the agent and one function called “help function” because it used for print the position.

- *Board*

```
public class Board {

    /* THE CLASS BOARD IS COMPOSED BY
     * int sizex, sizey = define the dimension of the board
     */
    public int sizex;
    public int sizey;
```

There is one constructor and this class is just use to check the agent’s position and check if the agent stays inside the board. There is the function CheckktheCoord(Agent a, float speed) : If the position of agent is in the boarder they change the direction by the opposite direction.

- *Direction*

```
public class Direction
{
    /* THE CLASS DIRECTION IS COMPOSED BY
    * float x,y = define the next position of agents
    */
    public float x;
    public float y;
```

This class is created for move easier the agent.

- *Game*

```
public class Game : MonoBehaviour
{
    /* THE CLASS GAME IS COMPOSED BY ALL THE DIFFERENT PANNEL OF THE APP,
     * THE STATE, THE LIST OF AGENT AND ALL INFORMATION IMPORTANT
     * I CREATE THIS CLASS TO CAN CALLED ALL THIS DIFFERENT ELEMENT IN ALL SCRIPT
     */
    public StartPannel Startpannel;  Unchanged
    public PannelAddAgent PannelAddAgent;  Unchanged
    public PannelPause pannelPause;  Unchanged
    public PannelSettings pannelSettings;  Unchanged
    public PannelHelp pannelHelp;  Unchanged
    public STATE State;  Unchanged
    public List<GameObject> listeofAgent;  Serializable
    public float speedofAgent;  Unchanged
    public int numberofAgent;  Unchanged
    public Board board;  Serializable
    public Random aleatoire;  Serializable
```

This class allows to save all the information of the application. After the initialisation of this class, you can find all the component of the application (like agent, UI….)
You can also find the STATE. This parameter is used to know if the simulation is working or not (three states: PLAY, PAUSE, STOP) and you find the random uses in all the project.

- *UserInterface*

The UserInterface Class define the user interface and all the other panels herite of this class. This class was created just to implement the two functions ShowPannel() and HidePannel() common of all the panel.

- *PannelAddAgent*

```
// CLASS TO DEFINE THE PANNEL TO ADD AN AGENT
No asset usages  4 usages  2 exposing APIs
public class PannelAddAgent : UserInterface
{
    public InputField IDnumber;  Unchanged
    public InputField PosX;  Unchanged
    public InputField PosY;  Unchanged
    public Dropdown ChooseDirection;  Unchanged
    public Button trybutton;  Unchanged
    public Button ApplyButton;  Unchanged
```

This class is created to combine all the component of the Pannel Add Agent.

- *PannelHelp*

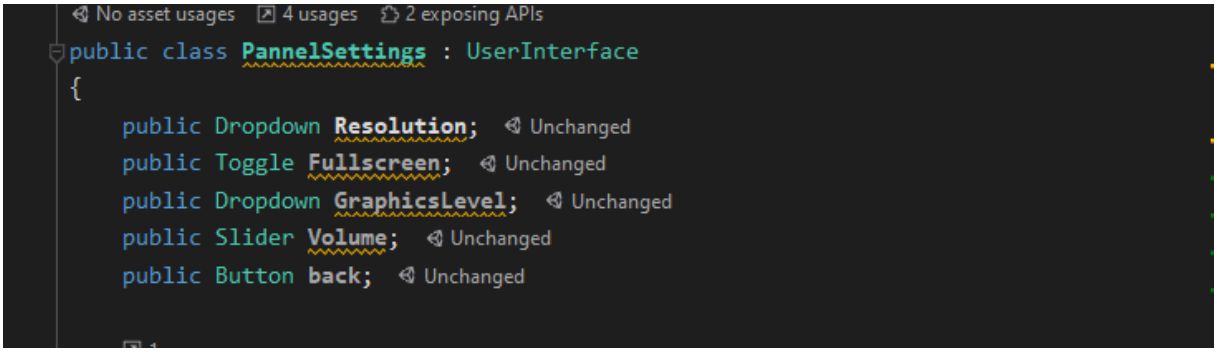
```
// CLASS TO DEFINE THE PANNEL OF SETTINGS
No asset usages  4 usages  2 exposing APIs
public class PannelHelp : UserInterface
{
    public Image Background;  Unchanged
    public Text titel;  Unchanged
    public Text description;  Unchanged
    public Button buttonQuit;  Unchanged
```

This class is created to combine all the component of the Pannel Help.

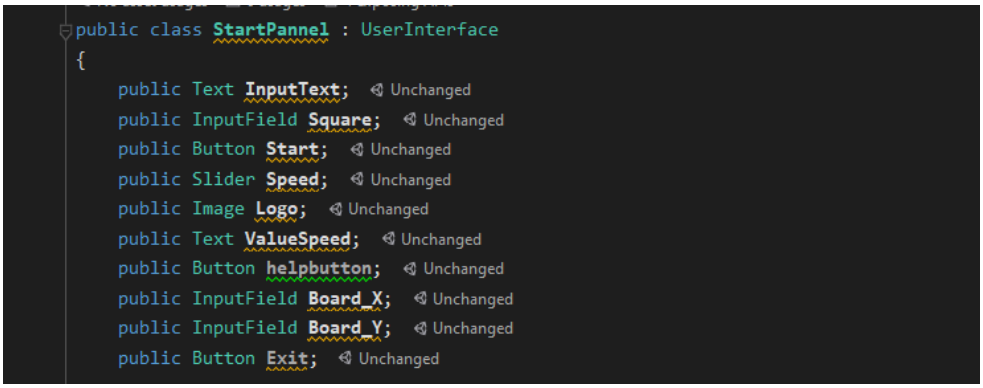
- *PannelPause*

```
// CLASS TO DEFINE THE PANNEL TO PAUSE
No asset usages  4 usages  2 exposing APIs
public class PannelPause : UserInterface
{
    public Button play;  Unchanged
    public Button option;  Unchanged
    public Button quit;  Unchanged
```

- *PannelSettings*



- *StartPannel*



4. Code C# - Application Class

- *AgentApplication.cs*
  - I. <u>Creation of all the variable</u>
    In this part we just write all the variable, we will use after.
  - II. <u>Function Start()</u>
    For this first time, I explain what the usefulness of this function is. It is an event function connect to Unity and it use at the beginning of the running of the application.
    This function is always used to synchronise variables and components.
  - III. <u>Function Update()</u>
    For the first time, I explain what the usefulness of this function is. It is an event function connect to Unity and call each frame. In this function, we will actualize the position of each agent and if the State is play, we will move the agent and check the coordinate.
  - IV. <u>Other Function</u>
    The other function are used when something enters on triggers of the Agent.

- *Application.cs*
  I. Creation of all variable
  II. Function Start()
  III. Function Update()
  IV. Other Function

  Public void clickButttomStart() : Check the different input. If there is a problem, send an error to the user else start the simulation.
  Public void OnValueChanged(): Synchronise the value of speed chooses by the user and the value in the StartPannel
  Public void SetPause() : Actualize the STATE of the game and hide the panel of setting and the panel of add agent but show the panel of pause.
  Public void SetStop(): Actualize the STATE of the game and hide the panel of pause and panel of add agent but show panel of start. The function destroys all the agent too.
  Public void setPlay() : Actualize the STATE of the game and hide the panel of start and the panel of pause but show the panel of settings
  Public void OpenSettings():Hide the panel of pause and the panel of start but show the panel of settings
  Public void buttontryexception(): Check each input in the panel of add agent. If it's good, activate the button of apply else instantiate the window of error.
  Public void CreateAAgent() : After the verification in the panel of add agent, create the Agent and restart the simulation
  Public void SetVolume(float volume) : Set the volume in function of volume
  Public void SetQuality(int qualityIndex): Set the quality in function of qualityIndex
  Public void SetFullScreen(bool isFullscreen): Set the FullScreen in function of the Boolean isFullscreen
  Public void SetResolution(int resolutionIndex): Set the resolution in function of resolutionIndex
  Public void returnSetting() : Set the state of the game at Pause
  Public void QuitTheApplication(): Quit the Application

- *Start*ingApp.cs
  This class is used to show the little introduction with the name at the launching of the application. How does it work? There is 16 pictures and each seconds or two, the picture showed changes by a new with one more letter. And when the user pushes "Space", the Scene with the simulation is launched.

- *SwitchCamera.cs*
  This class is used to switch between the different camera in our scene to have different axis to watch our simulation.