Simulation Game Report

Ryan Sewell
20045188

University of the West of England

April 10, 2023

The task for this project was to create a simulation game. The game must allow the player to interact with the environment to create buildings and anything the game requires. It must also have a framework in order to run the simulation. This task was achieved with a supermarket management simulation game made in Unity.

1 Introduction

To create a simulation game, there are two main parts. Building and the actual AI simulation. In the case of, for example, a zoo simulation game such as Planet Zoo Frontier Development plc, 2019 the building part of the game consists of creating paths, walls, and placing individual props, such as decorations or small shops. The simulation part of the game refers to AI agents visiting the zoo and interacting with exhibits and shops within it, as well as employees working in the zoo with different jobs such as janitor or zoo keeper. These parts can seem to be separate but must be connected somewhat as the things the player can build must allow the AI agents to interact with it.

2 Related Work

Many simulation games have been made before this and were viewed for inspiration. The Planet Zoo Frontier Development plc, 2019 and Planet Coaster Frontier Development plc, 2017 gave ideas, particularly for the building aspect from these games. The way paths and walls are built in these games is through clicking a point, then clicking another point to create a wall or path between the points. These, along with other games such as Another Brick in The Mall The Quadsphere, n.d. were also viewed to consider the behaviours and actions of both employees and customers in the supermarket, as well as how to handle the management aspect such as stocking the shop and hiring employees.

3 Method

3.1 Camera & UI

This project was done in Unity as it was the program I had the most experience in, especially with AI agents. The first part of the game to create was the camera and the layout of the UI. The camera controls allow the player to fly around the scene in all directions using the WASDQE keys, however rotating the camera is only possible while holding the middle mouse button, which locks the cursor. This allows the player to interact with UI elements using the mouse while not holding the middle mouse button. It is also possible to cancel any UI element from being selected using the right mouse button. Examples of the UI are shown in figures 1, 2, and 3.

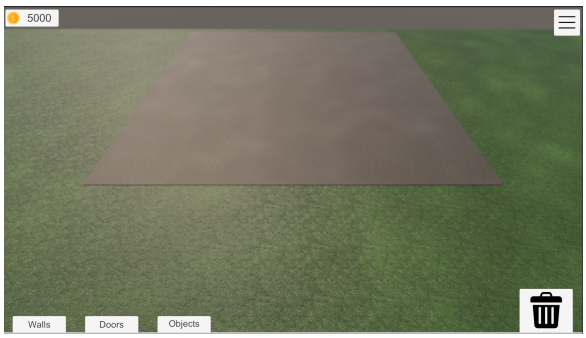

Figure 1: *An in-game screenshot with no UI elements selected.*

Figure 2: *An in-game screenshot with the objects tag on the bottom selected. From here it is possible to select an object to place into the world.*

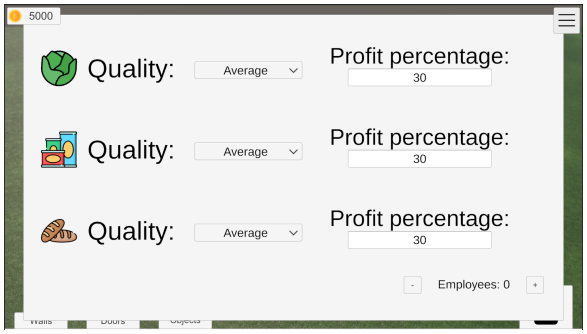

Figure 3: *An in-game screenshot with the menu element in the top right selected. From here it is possible to adjust pricing and quality of products, as well as being able to adjust the number of employees.*

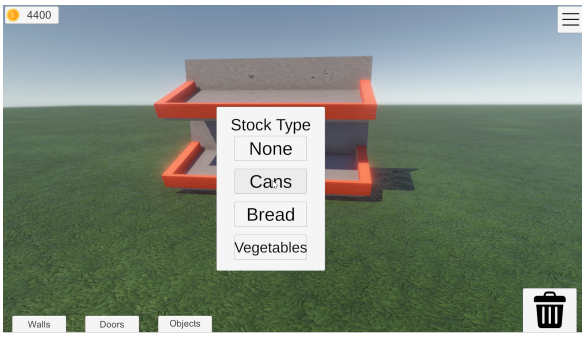

Figure 4: *An in-game screenshot showing how the product on a specific shelf can be set.*

Figure 5: *An in-game screenshot showing an invalid placement for a shelf. It turns red and will not be placed when the left mouse button is pressed.*

3.2 Building

The buttons in the UI for buildable objects will create that object and allow the player to place it. In general, when building an object it will follow the mouse and be placed when the left mouse button is pressed. When placing normal objects such as the shelf, chair and register, the method for placing it is simple. The object will follow the mouse position and can be rotated with the O and P keys. If the object is not intersecting another object and the player has enough money to buy it, it will be placed with the left mouse button, if not, it will be shown to be invalid as shown in figure 5. It is also possible to stop placing objects by pressing the right mouse button. These objects have invisible points connected to the prefab indicating where the employees and customers must stand to use them. After a shelf has been placed, it is possible to select the product that will be placed on the shelf. This is shown in figure 4. Once this has been selected it will remain empty until an employee restocks the shelf. It will also empty any stock remaining if the product is changed before it is empty.

When placing walls, a "post" will be created at first, and after the first post is placed the regular wall will follow the mouse. This is shown in figure 6. This can also be invalid to place if it is intersecting another object. The wall itself does not follow the mouse, there is an invisible object inside the end of the wall that follows the mouse. This invisible object, combined with the previous post allows the wall to calculated the position, rotation and length it should be. This also allows the wall to scale the material on it so that the bricks on the wall are always a consistent size and shape. After multiple connected walls have been placed, a roof and floor will be created. The roof is see-through so that the player can easily see inside the room and place objects indoors, as shown in figure 7. This allows the player to create buildings quickly. When the player wishes to close a room by connecting it to a wall or post, the wall will automatically snap to the wall and complete the room, allowing the player to create a new room. In the same way, it is also possible to connect a new wall to an already existing wall to extend a building.

The doors work slightly differently to normal objects, as it must be placed inside of a wall, otherwise the placement is not valid. When the mouse is on a wall, the door will snap to the correct position on the wall. When a door is placed, it splits whatever wall it was placed on into two separate walls. It uses the sides of the walls as the new post and "wall end", allowing the walls to be positioned correctly next to the door. When the door is deleted, it removes one of the walls connected to it and moves the other wall to replace where the door and other wall used to be. This is demonstrated in figures 8 and 9.

Figure 6: *An in-game screenshot showing a wall following the mouse, ready to be placed.*

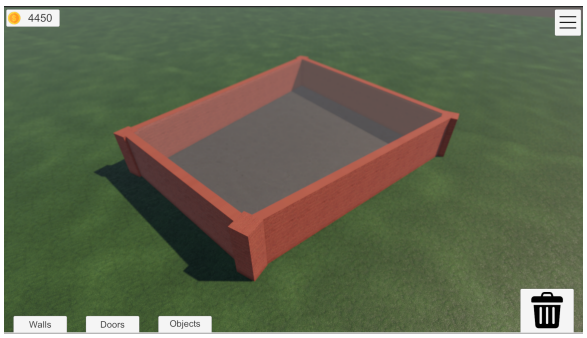

Figure 7: *An in-game screenshot showing a completed room with roof and floor. These are automatically generated as walls are placed.*

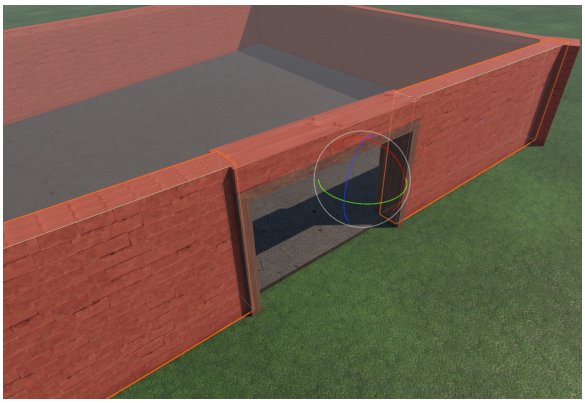

Figure 8: *An screenshot in editor after a door has been placed in a wall in game. Note the two separate objects for the wall.*

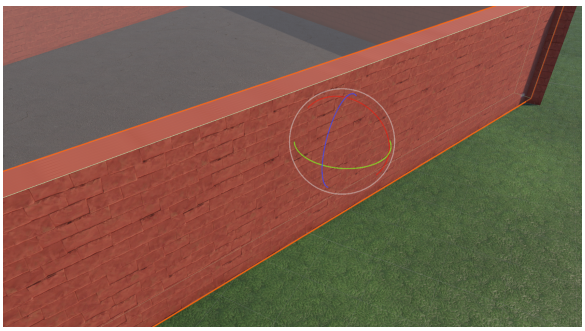

Figure 9: *An screenshot in editor after the door in 8 was deleted in game. It is now a single object again.*

3.3 AI Agents

3.3.1 Employees

As mentioned earlier, it is possible to hire and fire employees via the menu. These hired employees will take a wage from the players money in regular intervals. Once these employees are hired, there are multiple states they can take, as shown in the below code.

```
1       switch (employeeState)
2         {
3             case EMPLOYEESTATE.idle:
4                 EmployeeIdle();
5                 break;
6             case EMPLOYEESTATE.rest:
7                 EmployeeRest();
8                 break;
9             case EMPLOYEESTATE.stack
    :
10                EmployeeStack();
11                break;
12            case EMPLOYEESTATE.
    register:
13                EmployeeRegister();
14                break;
15            case EMPLOYEESTATE.walk:
16                EmployeeWalk();
17                break;
18        }
```

When the employee is in the idle state, it will first check it's energy levels. If the energy is too low to work, it will try to find a chair in order to rest. If there is enough energy, the employee will then check if there are any jobs available, such as stocking an empty shelf or manning a register. If nothing is available, it will choose a random position nearby and walk towards it. The walking state sets the employee to walk in the direction of its destination, and will change its state to another once it reaches the destination. The stack and register states are similar, the employee will stand at the shelf or cash register and perform the job until it is finished or they run out of energy. For the resting state, the employee will stay on the chair and regenerate energy until it is full again.

3.3.2 Customers

The way customers spawn is different to employees, it is not possible to choose to spawn more customers. When there is stock available, an agent spawner will consider the value of all the availble stock, based on the quality chosen and the price (profit percentage mentioned earlier). If the value of these are acceptable, it will spawn a number of customers depending on how much stock there is and the value of all of it. These spawned customers will have a list of products

they want to buy, and this will also vary based on the availability and value of the products. Customers also have an energy level, similar to the employees. However, they will not rest in chairs. A customers energy level will affect how much of their original list they will be bothered to get, for example, if a customer has 50% energy, they will only try to get 50% of their original list. If a customers energy reaches zero, they will drop everything they have collected to buy and leave immediately. Customers also have similar states to employees, as shown in the code below.

```
1           switch (customerState)
2          {
3              case CUSTOMERSTATE.idle:
4                  CustomerIdle();
5                  break;
6              case CUSTOMERSTATE.
    register:
7                  CustomerRegister();
8                  break;
9              case CUSTOMERSTATE.take:
10                 CustomerTake();
11                 break;
12             case CUSTOMERSTATE.walk:
13                 CustomerWalk();
14                 break;
15         }
```

These states all work similarly to how the employees states work, however the customer will only go to a register if there is an employee working it, and they have completed their list or don't have enough energy to finish the rest. After a customer has been to a register, it will exit.

4 Conclusion

This project has managed to create a simulation management game. The performance is acceptable, and does not seem to cause any significant lag, even with a very large number of agents. However, there is still more room for expansion in this project. For example, making customers and employees more unique in the form of personalities or skills, for example, a certain employee can restock a shelf faster but is slower on a register, or a customer with less energy to buy things. It could also be expanded by adding more mechanics, such as mess that employees must clean, or having to build a car park for customers and employees that would limit the number of people that can be in the shop.

Bibliography

Frontier Development plc (Nov. 17, 2017). *Planet Coaster*. Version 1.13.2. URL: https://www.planetcoaster.com/.

– (Nov. 5, 2019). *Planet Zoo*. Version 1.13. URL: https://www.planetzoogame.com/.

The Quadsphere (n.d.). *Another Brick in The Mall*. Version 1.1.2. URL: https://store.steampowered.com/app/521150/Another_Brick_in_The_Mall/.