CS324 Coursework Report

Instructions

- Download and expand the zip file
- Open the terminal and navigate to the zip file
- Enter the 'game' file using the following:
  cd game
- Enter the following command into the terminal:
  python3 -m http.server
- Open the files in the following order to enter the main menu:
  - game
  - solution
  - system
  - main.html

Concept

In the game, the player owns a food truck and sells burgers. In the first level, the player aims to collect ingredients for the burgers within the allotted time, and in the second level, they construct the burger by catching the ingredients in the correct order.

Overall Structure

An Entity-Component System was chosen to implement the game because it maintains the single responsibility principle and allows the reuse of components between entities and levels. All entities, such as the food truck and the collectable ingredients, are built upon the 'Entity' class found in entity.js and support children/parent relationships between entities. The entities are composed of components, such as models, health/inventory, and collision detection, which give the entities characteristics. All assets, namely the images and models, are stored in the 'models' folder. The game loop is created and managed by the files in the 'system' folder, where there is a JavaScript file and HTML file for the main menu, first level and second level, respectively. Each level has its own game file, which controls the game loop, a world file which contains the cameras and controls and subsidiary entity and component files.

The game can be navigated entirely using the buttons created and can be paused using the pause button and quit using the quit button. Further, there is a visual theme of pink and purple, seen in the food truck model, main menu, information, game over screens, and all text in Comic Sans. The menu has two buttons describing how to play the levels, and the instructions include the controls and how to win. Upon finishing a level, a game over screen appears with the score, why the player won or lost and buttons to navigate back to the main menu or restart the game. These elements allow for a smoother gaming experience because all the buttons used are in the web page's body, and the theme makes the game appear well thought out and aesthetically coherent.

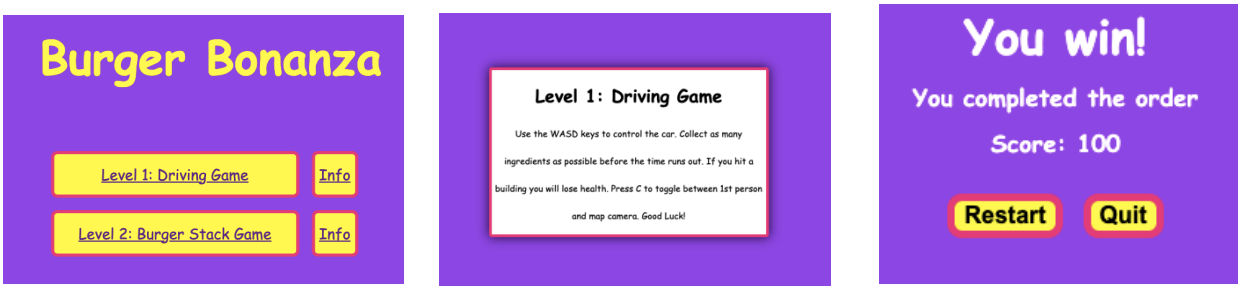

Figure 1: Colour scheme over multiple parts of the game

Level 1: Driving Game

In the first level, the player has 90 seconds to collect the five ingredients, which have been placed at one of 36 random places on the map. When the car hits a building, the player loses health, and when the player collects an ingredient, the health also increases. Upon finishing the level, the final score is a percentage of the ingredients collected.

Environment

Two direct lights illuminate the map from two sides, and two cameras are available. The initial camera is a first-person view which captures the surrounding buildings and roads, whereas the second camera is an overall view of the map. The combination of these allows for easy driving and navigation, respectively, and they can be swapped easily by pressing the 'c' key.

Models – Creation and implementation

Map

To create the map, a plane was divided into squares, segmented into 2x3 or 2x4 blocks, and arranged into a map. The roads were made by designing plane segments which resembled straight roads, three-way junctions, four-ways junctions and corners, and then replacing the map segments with the road segments. The map junctions and block locations are hard-coded into the 'drivingMap' file because it is faster and lighter to have one static global set of locations to determine collisions rather than a dynamic system. A possible expansion on the current implementation could be to make the map system more dynamic by including random allocation of building blocks or path finding to determine the roads.

Blocks

The city scene was made up of blocks of buildings to support collision detection and prioritise the reuse of models. This was done instead of making one large map because creating bounding boxes of a smaller rectangular model is more straightforward than fitting a bounding box to a whole map with lots of vertices. A model for a block of flats and a house was repeated to create a rectangle of buildings. When copying structures, the material colours were changed to give the illusion that there is more variation of the building blocks, making the scene more interesting.

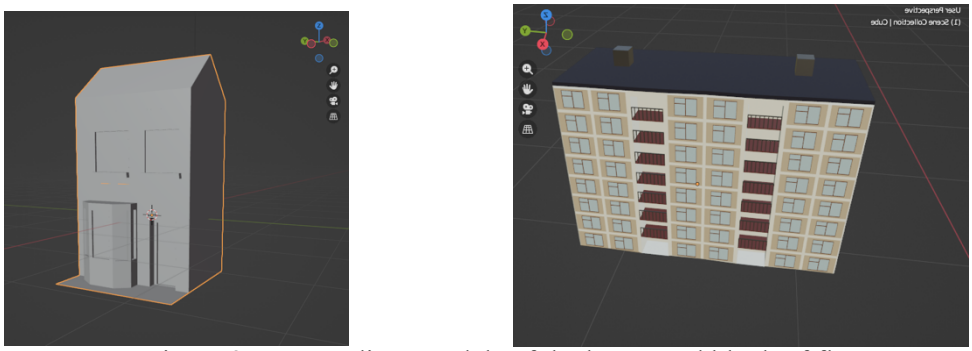

Figure 2: Intermediary models of the house and block of flats

Food Truck

The first model created was the food truck with a metallic sheen to emulate car paint. The game takes input from the WASD keys using the event listeners and moves the model to the position which has been computed by the 'foodtruckController' component.

Figure 3: Intermediary model of the food truck

Food Items

Numerous food items were modelled, including burger patties, onions, and tomatoes, to be collected by the player. The models are all scaled to the same size and float above the road before being collected.

Physics Systems

Collision detection

Collision detection is used to determine whether the food truck has hit a building and should decrease its health component or whether the player collected an ingredient and is rewarded with an increase in health. The chosen implementation created bounding boxes for particular objects and tests for the intersection between any of the boxes. If there is an intersection, then a collision has been detected.

Driving Controls

To operate the truck, a control system was created using the principles of 2D kinematics. Parameters include acceleration/deceleration, maximum speed and turning speed. This approach was chosen over moving the truck linearly because it improves the overall experience as it mirrors how real-life cars work.

Heads Up Displays

In the first level, there is a heads-up display which shows the current health and proportion of ingredients collected and a timer. These were implemented to provide the player with as much information about the game as possible whilst still allowing them to focus on being kept on the game. The buttons to pause and quit the game are also here for navigation.

Extensions

Some extensions to the level which could have been implemented were improving the collision detection to be more accurate (due to minor model scaling issues) and creating icons which sit above the truck and ingredients models so that they can be tracked easier from the map camera view.

Level 2: Stacking Game

In the second level, the player must correctly catch the falling ingredients according to the heads-up display order. The game will be over if the order needs to be corrected or time runs out. The final score is a percentage of the order which was successfully caught.

Environment

The scene consists of an oven stove illuminated by two direct light sources, and the environment contains two cameras that can be toggled between. The first camera view primarily provides a three-dimensional view of the scene; however, the alternative birds-eye view is better

suited towards gameplay because shadows were not implemented in the game, and it is thus easier to determine where the objects are falling from a top-down view.

Models – Creation and Implementation

Food items

The same models are used for the food items as in the first level to maintain the simplicity of code and to keep the workload sensible. The intermediary burger fillings (patty, onion, tomato, lettuce and cheese) fall from the 'sky' by simulating gravity to be caught by the bottom bun so that the models can be stacked into a burger. These falling fillings are kept in their own class, specifically maintaining and updating the falling objects.

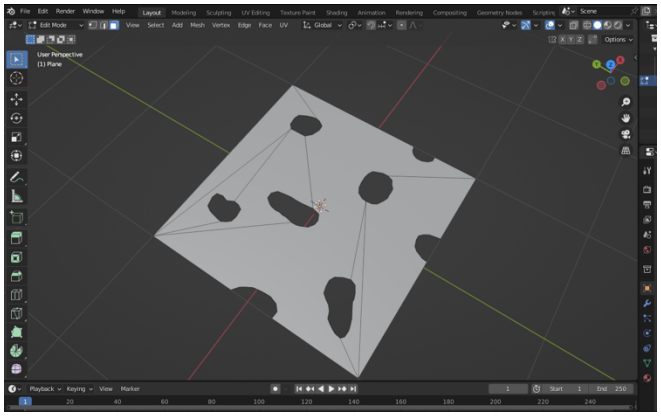

Figure 4: Creating holes in a plane to model a slice of cheese

Stove

A professional stove model was created to provide a background to the level and builds upon the cooking theme of this level and the overall theme game.

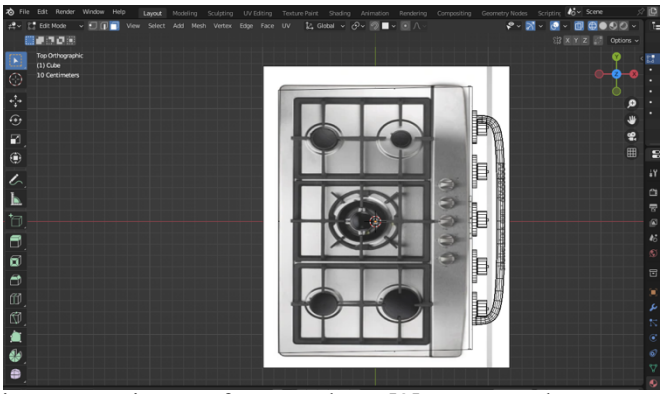

Figure 5: Using a reference photo [2] to create the stove top

Physics Systems

Gravity and collision detection

In the level, gravity is simulated using 2D kinematics and a decreased gravitational pull to give ample time for the player to catch the falling ingredients. Upon collision, the top of the tower's collision box is swapped for the falling objects' collision box, making the new collision box at the top and the fallen object follow the entire burger stack. Through this method, the game stacks the falling ingredients until the game is over.

Heads Up Displays – The Order

This level requires a random list of ingredients to be created so the player can match the order to win. To implement this, a heads-up display which shows images [1] of the ingredients was created using CSS. When an ingredient is caught, the order list updates. Additionally, a timer is provided to

count down until the end of the game. To further cohesiveness, the ingredient images have a pink border, and the background is dark purple.

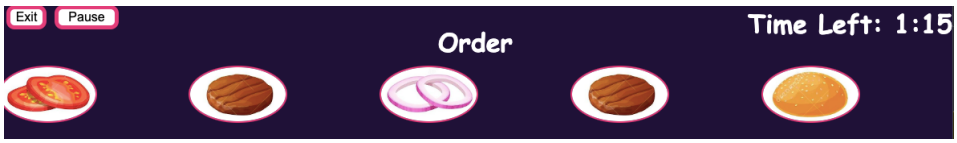

Figure 6: HUD for the burger stacking level

Extensions

To improve this level, a significant upgrade would be to implement shadows so that the 3d camera view is more practical; beyond this, another improvement would be to extend the model of the oven stove to be the back wall of the food truck.

References

[1] Burger ingredients collection. DIY burger elements isolated on white backgroud in cartoon style,

Dreamstime, https://www.dreamstime.com/burger-ingredients-collection-diy-elements-isolated-white-

backgroud-cartoon-style-sliced-vegetables-sauces-bun-cutlet-image145867584

[2] stove hob cooking kitchen cooker metal burner gas kitchen electric hot cooktop kitchenware stainless stove, shutterstock, https://www.shutterstock.com/image-photo/stove-hob-cooking-kitchen-cooker-metal-636688504