Hyunbin Lee 11/3/22

Project Proposal

Project Description:

Burgeria 112

Welcome to Burgeria 112! You are in charge of a fast food drive-thru for seven days. Fulfill orders using the grill, fryer, cutting board, and soda machine available to you. You must be both speedy and careful not to mess up, or even worse, miss an order as this will impact your score! As you progress through the days, you can unlock new toppings like lettuce and tomatoes, but your orders also become more complex! After each day, your performance will determine if you will advance to the next day!

Similar Projects:

I have always enjoyed playing cooking games in different forms, whether it's Papa's Burgeria or Overcooked 2. As a result, I decided to combine the two, adding the kitchen mechanics of Papa's Burgeria, but the chaos and time restraints of Overcooked 2, where you can lose if you perform poorly.

However, my game deviates from both example games in terms of my game's structure, where my game lasts for seven in-game days of five minutes each, meaning my game is designed to be beaten in one sitting. In addition, instead of tallying points based on the number of orders fulfilled like Overcooked, I added algorithmic complexity to the way points are tallied, with more points rewarded with greater size, accuracy, and timelienss of the completed order. In addition, orders may expire if not completed in time, therefore awarding zero points. The player's score for the day increases after each successfully completed order, and if the player reaches the score necessary to reach the next level, the player advances. If the player does not reach this score threshold, the player loses the game. Finally, orders become more increasingly complex as the player progresses, and if the player reaches day 3 and 5, the player unlocks the ability to serve lettuce and tomatoes, respectively.

Structural Plan:

To organize my finalized project, I will utilize separate appStarted, timerFired, mouse events, key events, OOP classes, redrawAll, and createInterface files. A large proportion of my code will be done in OOP classes, as I have to initialize many ingredients, orders, and interactive

pieces of my interface. My complexities of order randomization and game difficulty moderation will be done in a separate file as well that will be called in appStarted and timerFired, as your order complexity is dependent on your daily performance.

Algorithmic Plan:

The most algorithmically difficult part of my project is the point reward system. A combination of larger orders and orders completed more accurately have more influence on the player's score. In addition, orders may expire if the player does not complete them in time, meaning an expired order would reward zero points. This incentivizes players to complete orders quickly and accurately as they offer the best possible way to increase one's score. I plan to tackle this part by using a point system that awards points based on the player's performance, accuracy, and timeliness. Furthermore, an algorithm is designed to reward those who complete orders quickly, but also provides a little padding to orders that are not, as it takes into account that certain orders may take longer to complete as they are larger.

In addition, another algorithmically complex aspect is that orders are randomly generated, and I will need to appropriate this to the time limit. In order for the player to complete all of the orders and get through the day, the remaining amount of orders generated must be humanly possible to complete. Order volume will also need to be proportionate to the difficulty of the level because certain toppings are not unlocked until certain levels and the higher the level, the more items each order will require to be completed. This will require coming up with a recursive algorithm to randomly generate orders based on timerFired, and then vigorously playtesting and trying different timer values to randomly generate the order in a way that is accessible by the player.

Finally, interface interaction is an algorithmically complex aspect, as the game will need to identify if the placement and interaction with items allows them to be cooked. In addition, the game will need to pathfind the nearest object if one is not currently clicked on. As a result, the game requires many legality and bounds checks to make sure that the proper item that needs to be cooked is cooked. Furthermore, with burger creation, the game will need to be able to recognize if a legal burger is to be created, and initialize a burger with the correct toppings appropriately.

Timeline Plan:

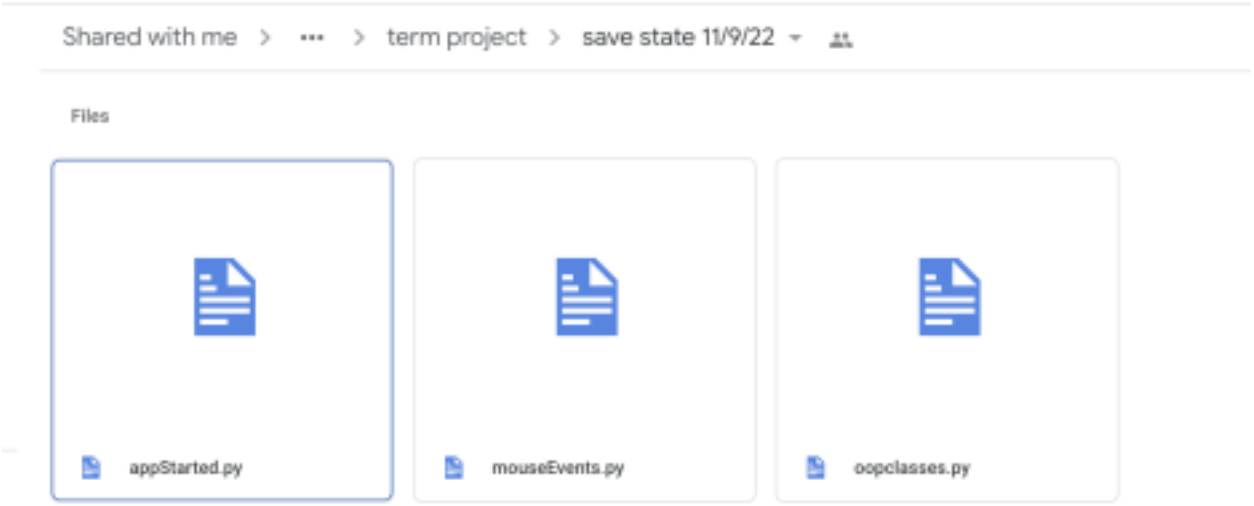
By TP0: Complete graphics, skeleton code, pseudocode for all features

By TP1: Implement graphics along with skeleton code to make an interactive, working interface

By TP2: Submit project, with working, interactive features, hopefully at MVP By TP3: Submit project with clean efficient code along with possible extra features, readme file, video demo, all related design files

Version Control Plan:

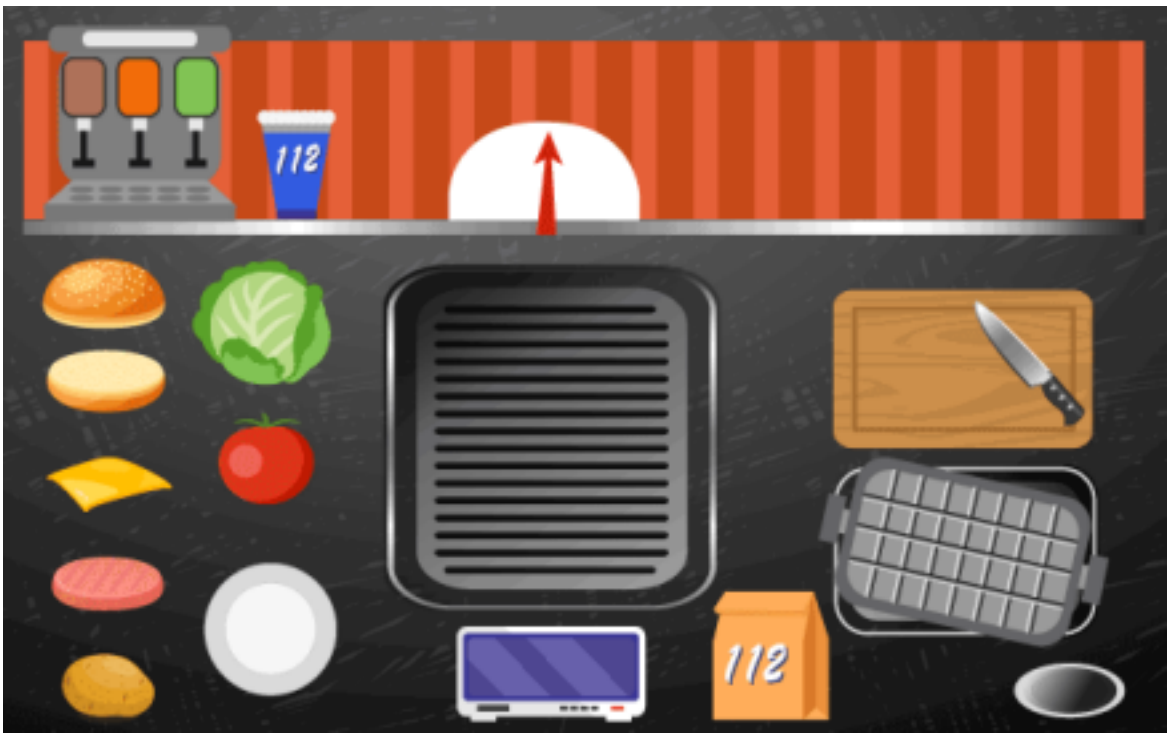
I am using a Google Drive folder stored on the cloud divided into work days to back up my code after every day to keep track of my editing history.



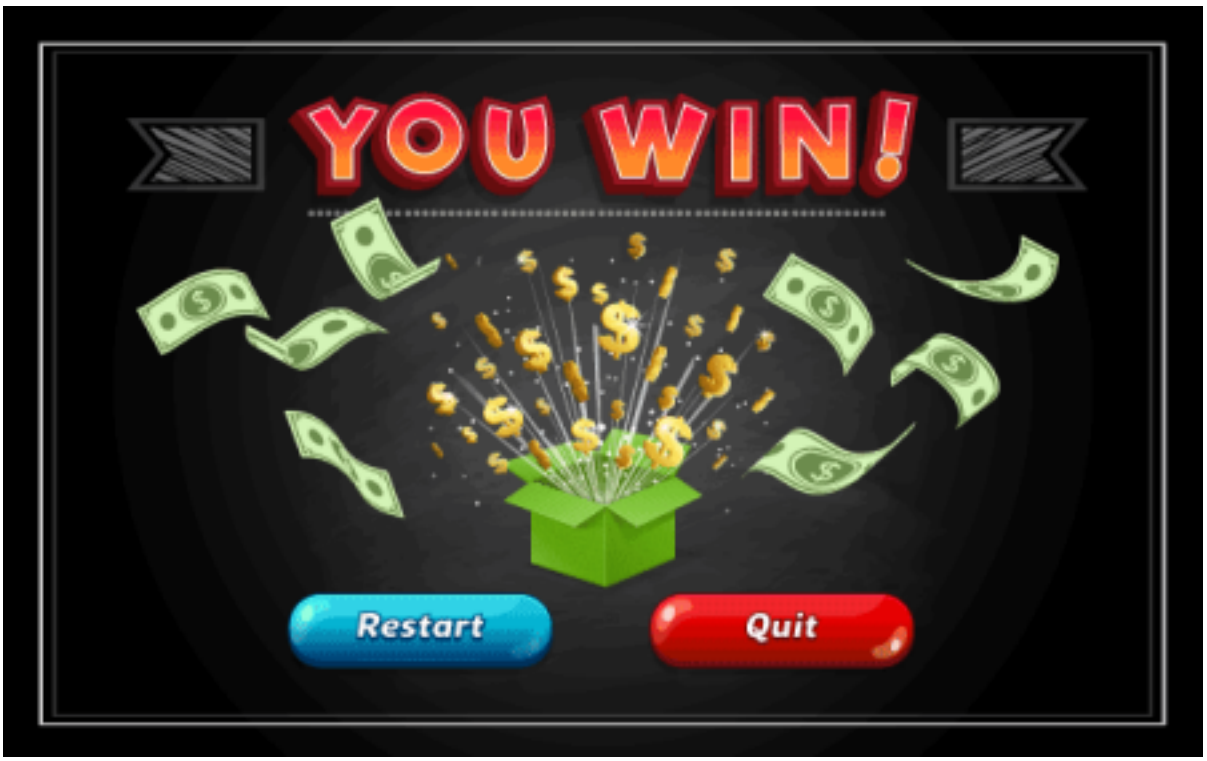
Module List:

I do not plan to use external features and modules.

User Interface:


112
112


DAY
meat 100%
burger
1
potatoes
fries



TP1 Update

● No design changes were made.

TP2 Update

● Interface color at top changed from orange to pink for better visibility of food sprites. ● Added pathfinding algorithm that finds the closest item to the mouse, even if no item is being clicked on.

● Full interface interaction, with full burger assembly features and fries creation. Randomly generated orders appear on the top right corner displaying what needs to be fulfilled and accordingly how much points the completion of the order rewards.

TP2.5 Update

● Added ability to serve items, through bag and serving window.

● Added drink dispenser.

● Added player score in bottom right.

● Added timer in bottom middle, now set to 5 minutes instead of 10 minutes. ● Added the ability to progress between in-game levels or days. Added day screens in between levels.

● Scoring system / difficulty moderation changed from star system to point system. Levels must be completed in order and proportionately become more complex. The player

unlocks the abilities to serve lettuce and tomatoes on day 3 and day 5 respectively. Although order volumes are still randomly generated, they increase proportionately to the level, as the player progresses.

- Added order expiration.
- The game records the player's score for the day and determines if the score meets the threshold needed to advance to the next level. If not, the player loses the game. At the end of the game, the player's entire score, covering all of the days, is totaled and given to the player.
- Project has reached MVP.

TP3 Update

- Added "How to Play" screen.
- Added trash can, which allows the player to trash items in the bottom right. ● Score is now impacted by order timeliness using an algorithm. In addition, excess unnecessary items impact the score with a reduction per excess item. ● Rewrote scoring algorithm to be more accurate and heuristic. Scoring system can fully accommodate for order expiration (the expiration of one order no longer affects the possible scores of others). Scoring system examines each ingredient one by one and compares it to the necessary ingredients to award a score.