Factory Manager Full Game Scenario

Group 28 - Dongxin Zhang, Tao Wang, Luke Repta, Mike Apreza

The second Factory Manager game scenario focuses entirely on making the program a fully-functional game where players are able to play the role of a factory manager managing and expanding their industrial empire. The player will actually be allowed and forced to make calculated decisions without guidance and constraints from the tutorial mode.

The key non-server based features of the game to be implemented in this scenario are: (1) making the research sub-system fully functional, (2) creating the interval function that controls and updates the model (gamestate) and view, (3) scaling the game to multiple regions with their own set of factories, (4) adding the truck transportation sub-system of products and resources, and (5) a win condition for the player. The research, multiple regions, and delivery features are specified in: Group 2 - Factory Manager - Final Report.pdf. The definition of the classes mentioned below can be seen in researchNode.js and gameState.js, respectively, under the group's GitHub repository. A visualization of the way the systems will potentially interact with each other can be seen in Figure 1 below.

Scenario "Factory Manager Full Game"

For this game, a fully functional research system will dynamically create a research tree based on the nodes in a predefined list of researchNode (a class) type. Each node will require an amount of time and cash to research, which will be indicated visually to the player. Upon completion, depending on the type of the research node, some additional machine, product/resource, or bonus modifier will be unlocked in the gameState. This will interact with the interval function, as each unit of time will be controlled through that, and it will also handle updating the user interface and gameState (a class) when a research is completed.

The interval function will update all variables in the gameState, such as the number of products, money, set of researched nodes (and handle what happens when a node finishes researching), unlocked machines and blueprints. This will be the main interconnect between the different systems, as they will follow a form similar to the observer design pattern based on the state of the gameState as manipulated by this interval function.

There are three planned possible regions a player can develop their industrial empire. Regions have their own set of factories, resources, products, and transportation network that is unique to each region alone. To unlock other regions, the player will have to spend a specified amount of money. Locked regions do not have a negative impact on the player's status. This will also interact with the interval function, as it will iterate over every region the player has to calculate the updated gameState variables that are both region-specific and non-region-specific.

As long as the player has money, resources, and/or products, they can set up a delivery route at any time. There are three possible stops a player can take: Factories, Stores, and Resources. The factories stop delivers the resources to the player and/or picks up the products to sell, the stores stop allows the player to sell their products for money, and the

resources stop lets the player use their money for resources, which are needed to make products using a machine. The truck that transports all these items is able to hold a limited amount of supplies and serve only one factory. For example, if a player has two factories, they can set up two delivery routes that serve both factories separately and simultaneously. The player can choose the stops they want to go to and what to drop off and/or pick up at a specified quantity. Depending on the number of stops and items to drop off and/or pick up, the trucks will take a certain amount of time.

The winning condition of the game will be to acquire a total of $1,000,000, and have a per unit of time income of $10,000. The interval function will be monitoring for the winning conditions being met, and will also keep track of how many "time intervals" have elapsed for the user to reach the conditions. This will tell the user how fast they were able to achieve victory in the game, and will allow for replayability as the player attempts to get a faster time. The user will still have the option of continuing to play the game after the conditions are met.

Figure 1 - Planned System Diagram