2024/10/19

COS214 Group Project

Functional Requirements

Doofenshmirtz evil incorporated

Functional Requirements and their corresponding design patterns:

1. Building Structures and City Design

The simulation allows users to build a city consisting of many different districts, each consisting of different building types i.e. Residential, Commercial, Industrial and Landmarks.

Each building serves different purposes:

- Residential: Houses Citizens
- Commercial: Pays Citizens salaries
- Industrial: Generates Resources
- Landmark: Affects citizens satisfaction

Design patterns used:

- Composite: Models the city structure
- Factory Method: Used to create new concrete building types
- Template Method: Used within the Factory Method to allow the build() method to infer what type of building to create
- Command: The concrete SpendResources command makes an algorithmic decision as to what type of building to construct based on satisfaction, resources and government balance. (See Government)
- Iterator: An Iterator may be used to traverse through the composite structure.

2. Utilities:

The simulation allows the construction of different utilities similar to buildings. Utilities include Power, Water, Waste and Sewage. Each utility generates a different utility stored in the Government.

Design patterns used:

See Building Structures and City Designs as utilities servers as an extension.

3. Citizens:

The simulation meticulously models citizens and their daily activities including a routine, having a work and a home, paying taxes and having a general level of satisfaction. Furthermore, citizens may also find themselves unemployed with drastic effect on their satisfaction.

Design patterns used:

- State: Used to model a citizen's routine. Citizens may find themselves in at home, at work, commuting or spending leisure time.
- Strategy: Used to determine which transportation method a citizen will follow. See Transportation.

Note: Population growth is dictated by means of the amount of residential building available. Upon completion of a residential building, it will automatically be populated with new citizens.

4. Transportation:

The simulation provides a multitude of ways for citizens to travel in their daily routines, including by road, railway using public transport or even by plane. The amount of citizen using each form of transport influences traffic conditions which, in turn, influences citizen satisfaction. The composite structure is iterated to find the distance between different buildings.

Design patterns used:

- Strategy: A strategy pattern is used in conjunction with the distance between current location and destination to infer which transportation method must be used by the citizen.

Government:

The government is perhaps the most important part of the simulation. This entity is responsible for collecting taxes, distributing resources for expansion, implementing policies and collecting information on the city structure and its citizens through means of calculating employment and satisfaction.

Design patterns used:

- Singleton: The government is a singleton ensuring that there is only one government influencing the city and provides a single access point to city operations.
- Observer: The government acts as the subject of the city which observes it. This is useful for methods such as collecting taxes since the government notifies citizens through means of the residential district that they should pay their taxes.
- Command: The government makes use multiple commands to implement their various policies, set the tax rate and spend their resources.

Other:

Some other patterns are used in a supplementary way or to more easily integrate other patterns. These include:

- A Singleton WebSocketNotifer class that is used to communicate with our HTML frontend by means of a web socket.
- A SimulationRunner Façade that is used to instantiate the Server and Government to allow easier integration between them.

A brief Summary of all used design patterns and their purpose:

This section contains the same information as the function requirements but summarised and organized by pattern.

1. **Composite:**
   - Used in the city structure to model the hierarchy of CityUnit, District, and Building (Residential, Commercial, Industrial, Landmark).
   - Iterated to calculate distances in Transportation.

2. Factory Method:

- Used to create new buildings (Residential, Commercial, etc.) and Utilities.

3. Template Method:

- Used within the Factory Method to customize the build process for different types of buildings.

4.Command:

- Used by the Government to execute policies (e.g., SpendResources to decide building types).
- Used for government actions like set taxes, allocate budget, and implement policies.

5.Iterator:

- Used to traverse the city structure (composite) for tasks like calculating distance or checking building status.

6.State:

- Used to model a citizen's daily routine (at home, at work, commuting, leisure).

7. Strategy:

- Used in citizen transportation to select travel methods (e.g., road, rail, air) based on distance.
- Used in CommuteState to manage citizen movement patterns.

8. Singleton:

- Used in the Government to ensure a single control entity.
- Used in the WebSocketNotifier to manage communication with the frontend.

9. Observer:

- Used to let the Government observe the city and gather information (e.g., tax collection through districts).

10. Facade:

- Used in the SimulationRunnerFacade to encapsulate the complexities of the Server and Government systems, providing an easy-to-use interface for managing the simulation and communication.

Brief Pseudocode of the main game loop:

For a detailed description of the game loop please see SimulationRunnerFaçade or our activity diagram.

```
Create basic "starter" city
Create goverment
Attach starter city to goverment

while(!StopFlag)
{
 employCitizens()
renderCity()
notify() [Calls update on all units]
collectTaxes()
collectResources()
updateUtilitiesManager()
evalutateTrafficConditions()
evaluateHappiness()
executeSpendResources()
Check for ShortWorkWeekPolicy
Check for BetterEducationPolicy

}
```