Anna Smith
asmith48

Chip's Challenge

Updates to UML Diagram

I added a number of methods and variables to each of my classes. The main changes I saw between my two UML diagrams was adding the Dashboard class and updating which methods were observers or observables. I added a separate Dashboard class so that I could control all graphics added to the dashboard in methods in that class. I kept the relation between the player and chips the same with the player being the observable and the chips being the observers. I changed the relation between the player and bugs so that the player was the observer and the bugs were the observable. This helped because with animation the bug object was moving much more frequently than the player and so called notify() more often, meaning that when the player was the observer it could update() more frequently and check more frequently if the two were on the same board tile.

Design Patterns

Pattern Name: Observer	
Class Name	**Role in pattern**
Chip	Observer
Player	Observable
Purpose: Chip objects need to observe player object so that they know when player is on same tile as them and can be "collected" (disappear from screen and add to total number of chips collected).	

Pattern Name: Observer	
Class Name	**Role in pattern**
Player	Observer
Bug	Observable
Purpose: Player object needs to observe all bug objects so that it is notified when the player and bug are on the same tile. It can then update the game to close the screen as this will mean "game over".	

Pattern Name: Singleton	
Class Name	**Role in pattern**
Dashboard	Singleton
Purpose: There should only ever be one Dashboard object used on the game board and instantiated in the Main class so using the Singleton design pattern will ensure the Dashboard class can only be instantiated once.	

Final Design Discussion

Within my Main class which extends the Application class I used the main() and start() methods. Within the start method I instantiated my root and scene objects and added the game board and dashboard to the root. With the drawMap() method of the BoardMap class I created a 25x25 board grid of tiles. With the draw()

method of the Dashboard class I created a dashboard at the bottom of my board grid which displayed the title “Chip’s Challenge”. From main I then called levelCreator() to create my first game level. Within level creator I drew the game maze, using the createMaze() method of BoardMap, and created a portal, using the createPortal() method of BoardMap. Within the createMaze() method I filled lines of the board with tile images to form walls. I kept track of which tiles of the board were filled by using the boardGrid array of BoardMap which was a 2D array filled with integers representing how each tile was filled. If a tile was empty its array entry would hold a 0, if it was part of the maze if would hold a 1, and different objects had other integer representations. I created a portal object by placing the portal image in a random empty tile on the board. I then added a player to the board, multiple chip objects, and multiple bug objects. I kept a list of bugs and a list of chips and added each chip as an observer of the player and each bug as an observable for the player. I then called movePlayer() to animate the movements of the player and bugs. In movePlayer() I used an AnimationTimer() to control continuous movement of the bugs. Within the Bug class the move() method either moved the bug up and down or left and right depending on a Boolean variable the bug object was instantiated with. The object would move up or left until it hit a border and then would move the opposite direction and then repeat this motion. Within this continuous animation the game also checks if it is game over, meaning a bug and the player have collided, and if so closes the screen, and checks how many chips have been collected. Also inside the movePlayer() method, inside an event handler, the game checks for all keyboard input. The arrow keys move the player object on the board and the ‘esc’ key closes the game window. With each input it also checks if the player has moved onto the tile with the portal object and if so moves the player to the next level by calling again the levelCreator() method with a higher level number. When the levelCreator() method is called for higher levels of the game it first resets the board with resetLevel() and removes all object’s image views from the screen and then recreates the objects and their image views at new positions for the new level. The only difference currently between level 1 and level 2 for my game is that level 2 has twice as many bugs. Currently if levelCreator() is called for level number 3 it will end the game and output “You Win” as the game only has two levels.

I think my use of the observer design pattern worked well in the game to keep track of when objects collided on the same game tile. Due to the way my animation worked it worked well to have the chip objects observe the player and the player observe the bug objects. If I could change my design, I would have a better way of constructing my maze, as I struggled to come up with a better implementation of this method. I would also find a better way to animate the game. The current animation is good, but I think a different animation strategy might allow the player to move quicker in response to user input. I would also want to add more functionality to my game in the form of more obstacles and more tools for the user to collect. I would also have a better way to construct new levels, as right now all levels are constructed randomly instead of having a set board for each level. It might be better to construct the maze the same each time a certain level is played.