Project 3 Report

Marvin Deng

Diagram

Description automatically generated

StudentWorld has a member variable called bank which stores the number of coins in the bank. It also has member functions resetBankCoins()and depositBankCoins() to modify the bank. Player has member function updateCoins(int coins) which returns the number of coins taken from the player.

Bank Square is derived from Square, which is derived from ActivatingObject. ActivatingObject has functions called onObject and passingObject, which returns a boolean for if a player moves/passes onto a square or baddie. Both functions take in a pointer to the Player or Actor as a parameter.

The Square class has a pure virtual function called processAction(). This function allows each square to provide its own implementation of what the square should do during a tick. This function takes in a Player pointer as a parameter.

Bank Square’s processAction() first calls onObject() to determine if the player has stopped on a Bank Square. This adds all the coins in bank square by calling getBankCoins() and passing the bank’s coins to player’s updateCoin(). The bank is then reset to 0 using resetBank() . If this the player doesn’t stop on the square, the function checks if the player has passed over the bank square. If the player is passing over, then I call player’s updateCoin() with -5 as the parameter to decrease player coins by 5. UpdateCoins() then returns the number of coins taken from the player, which is then deposited into the bank using depositBankCoins().

The doSomething() of Square calls StudentWorld’s getPeach() and getYoshi() functions to get the Peach and Yoshi pointer, which are passed into processAction(). This means the bank square will check if it can carry out actions on both Peach and Yoshi.

1. I believe I have implemented to the best of my abilities a functional game that fulfills the requirements in the spec.
2. Assumption made:

* After a player stops at a fork to choose a direction, an enemy that moves onto the player shouldn’t activated.
* A player that swaps off of an event square should activate whichever coin it gets swapped to.
* Spec only defined that the player that swaps onto the event square can’t activate it again
* Player should activate the square it teleports to.
* Spec doesn’t define if the player is considered “new” again after a teleport.
* Player should activate the square under an enemy before it activates the enemy.
* Order of activation not defined in spec
* If there are 2 baddies on one square, then the order in which the baddies penalize the player can be in any order, depending on the order in the actors vector