**REPORT**

1. *A description of the control flow for the interaction of a player avatar and a bank square. Where in the code is the co-location of the two objects detected, and what happens from that point until the interaction is finished? Which functions of which objects are called and what do they do during the handling of this situation?*

A banksquare holds a record of whether or not each player first appeared on its square on the current tick in a Boolean variable that is adjusted as follows:

Initially, it stores a value of true to say if ever a player were to have the same coordinates as the bank that they would be landing on it for the first time. Once they do have the same coordinates, the variable becomes false to say if they continue to have the same coordinates that the player isn’t new to its environment and thus no action needs to be taken. Finally, if the variable is false and the player’s coordinates vary from that of the banks, the variable is reset to true to indicate again if ever they have the same coordinates that the player re-appears for the first time.

Now bank square’s do something method checks if a player’s x and y coordinates match that of the bank square and if the player is landing on the square for the first time. If they do match and the player arrived at the square during this tick,

1. If the Player isn’t moving, the banksquare’s do something method calls the player’s adjust coins method(which is responsible for changing the coin count of the player), passing the get\_bank\_coins getter method in StudentWorld as a function parameter, which increments the specific player’s coins by the amount stored in the bank. It then calls the reset\_bank\_coins method also defined in StudentWorld.h/.cpp to set the bank coins back to 0.
2. If the player is moving and the player has > 5 coins, the banksquare’s do something method calls the players adjust coins method with a parameter of -5 that reduces the coins by 5. And then calls deposit\_coins in StudentWorld to add those 5 coins to the bank total. If the player has <= 5 coins, the banksquare’s do something method calls the player’s adjust coins method with negative of the players coins, which sets their coin total back to zero, and then calls deposit\_coins with the original coin amount the player had(Before it was reduced to zero).

Note: For any reference to a function defined in StudentWorld, do Something uses get\_ptr\_to\_student\_world() which returns a pointer to StudentWorld to call the specific function.

1. *A list of all functionality that you failed to finish as well as known bugs in your classes, e.g. “I didn’t implement the Vortex class.” or “My Bowser doesn’t work correctly yet so it behaves like a Boo right now.”*

I believe that I was able to successfully implement the game(to the best of my knowledge) as I didn’t encounter any bugs during my final few playthroughs of the game.

1. *A list of assumptions you made; e.g., “It was not specified what to do in situation X, so this is what I decided to do.”*

I assumed that any square can be changed by bowser into a dropping square, including bank squares, event squares, and such, and not only coin squares.

For the vortex projectile, I assumed that it should be created 1 square in front of the player although I was unsure as the spec just says “in front of.”