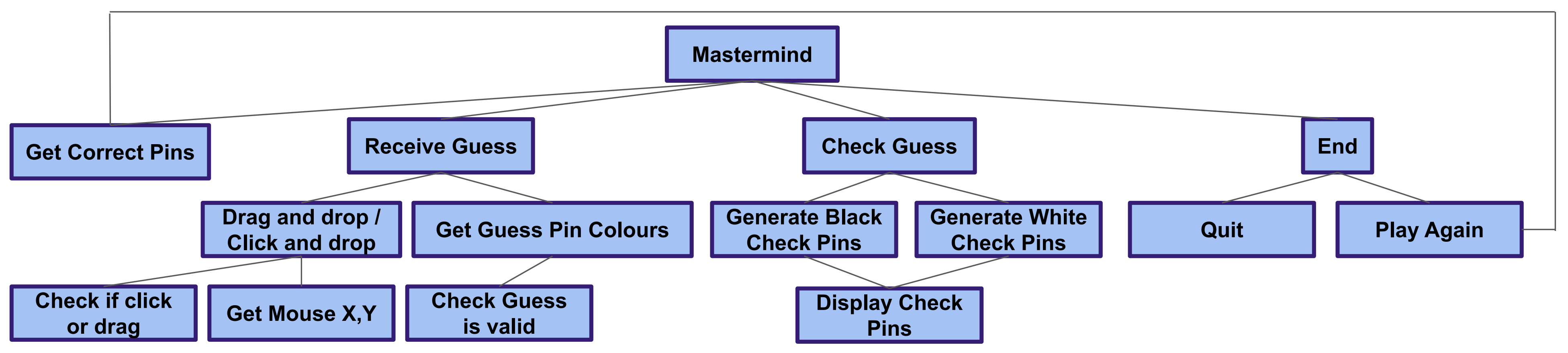
Analysis

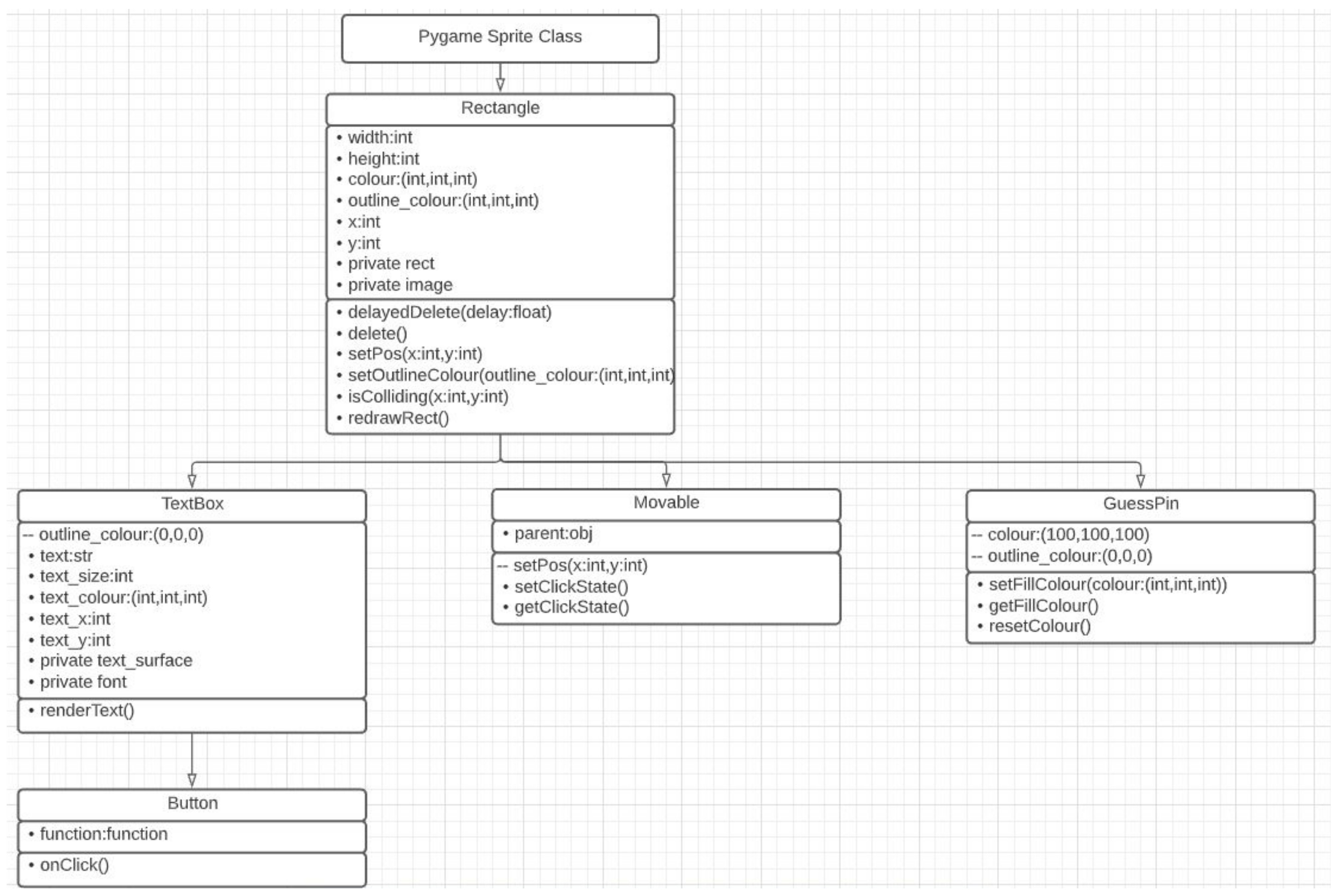
Criteria:

1: Allow user to guess
1.1: Drag and drop as well as click and drop
1.2: Allow user to change guess before submission
1.3: Each game must have a random combination of correct pins
1.4: User should be able to quit at any time
1.5: 4 guessed pins are required to submit a guess
2: Inform user how correct their guess is
2.1: Maximum of 4 'check pins'
2.2: Each pin of the combination or guess can only give one check pin maximum
2.3: Black pins (correct place and colour) take priority over white pins (correct colour only)
3: Repeat 1 and 2 until loss or win
3.1: Show the correct combination
3.2: Lose after 12 incorrect guesses
3.3: Win after 1 completely correct guess
4: Allow quit or play again
4.1: Play again should chose a new combination

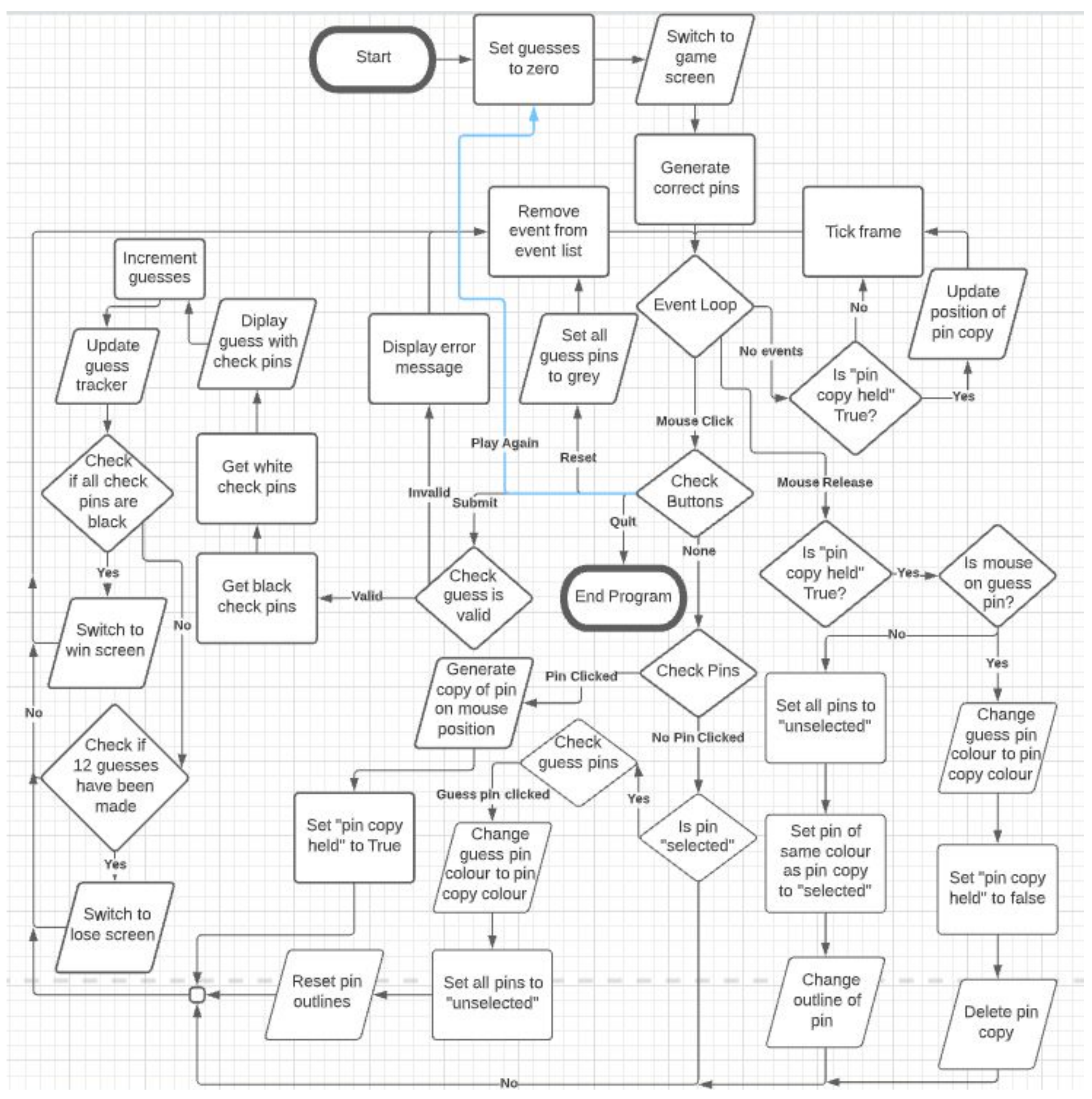


Design

Class Diagram



Flowchart



Implementation

Subprograms

getMasterPins():
Randomly generates four colours as the code and returns as a list.

createButtons():
Instantiates all of the button objects used in the program.

createActivePins():
Instantiates all of the active pin objects (ones that are used in the drag/click and drop system) used in the program.

createGuessPins():
Instantiates all of the guess pin objects used in the programs

createText():
Creates all of the text box objects used in the program.

resetPinOutlines():
Resets the colour of the outline for all of the pin objects.

guessPinCheck(holding):
If holding is not empty and the mouse has clicked or dropped a pin copy on a guess pin, the function changes the colour of the guess pin to the colour value stored in the 'holding' list. The holding list is always returned empty from this function.

drawStep(LIGHT_BLUE, all_sprites, text):
This function is run every frame and is used to update the sprites on the screen. It draws the background first, then any rect elements of sprites, then text, and finally it will draw the pin copy if the player is holding one.

center(x, y, obj):
The center function is primarily used by pin copies to allow them to be centered on the mouse when they are dragged across the screen.

invalidGuess():
This function creates a textBox that informs the user that they have attempted to submit a guess that does not contain four colours.

displayPins(guess):
This function creates objects as a visual record a of a users previous guesses, instantiating both the check pins and inert pins of the same colour and order as the users guess. It uses the current guess to make sure that the previous guesses do not override each other on the screen.

reveal(master_pins):
This function instantiates four inert pins an a text box to allow the user to see what the correct combination is after the game has finished.

getBlackChecks(check_pins, master_pins):
Check to see if any of the guessed pins are in the same position as a pin of the same colour in the code. Any pins that match are flagged ,either in the master_pins list for code pins or in the guess_pins_not_checked list for guess pins, to make sure they are not checked again in the getWhiteChecks function. If any matches are detected, BLACK is appended to the check_pins list.

getWhiteChecks(check_pins, master_pins, guess_pins_not_checked):
This function checks all of the unchecked pins against the whole code for colour matches, appending WHITE to the check_pins list if a match is found and marking checked pins in the same way getBlackChecks does.

getCheckPins(master_pins):
This function handles the whole pin checking process It calls both getBlackPins and getWhitePins if none of the guess_pins are grey (their default colour). This function returns a boolean containing the validity of the guess as well as the check_pins list.

copyStillClicked(holding):
This function checks to see if any existent pin copies are still being held. If they are it resets all active pins' outlines, moves the copy to the mouses current screen position and updates the holding list with the colour of the held copy. If the copy is no longer being held, it is deleted. Then all active pins have their outlines reset and the active pin the same colour as the copy is highlighted.

createPinCopy(mouse_x, mouse_y):
If an active pin is clicked, a pin copy of the same colour is instantiated, centered in the current mouse position.

checkButtons(mouse_x, mouse_y):
This function check to see if any of the buttons have been clicked. If one has, its onClick method will run.

doEventLoop(click, holding):
This function handles the event loop, checking whether the user has clicked or release the mouse. This function also handles the guess pin recolouring section of the code, calling the guessPinCheck function if the mouse is released on a guess pin.

checkWin(check_pins):
This function uses the check_pins to determine whether the user has won the game or not. If any check pins are WHITE or if there are less than 4 check_pins, 'win' is set to False and otherwise it is left as True.

gameEnd(win):
This function deletes all currently instantiated objects and then runs the createEndSprites fuction.

createEndSprites(win):
This function creates a text box telling the user whether they won or lost and reveals the correct combination. It also creates a quit button and a play again button.

updateGuesses(guess):
This function updates the text box that shows the user how many guesses they have left.

resetGuessPins():
Resets the colour of all guess pins. Used in the onClick method of the reset button.

submitGuess():
Runs the getCheckPins function to get the check_pins list as well as the 'valid' flag. If the valid flag is False, the function deletes any instance of the invalid textbox before creating a new one with invalidGuess and telling it to delete itself in three seconds. If the valid flag is True, the guess count is incremented by one. The resetGuessPins, displayPins and updateGuesses functions are all run. This function is used in the submit button's onClick method.

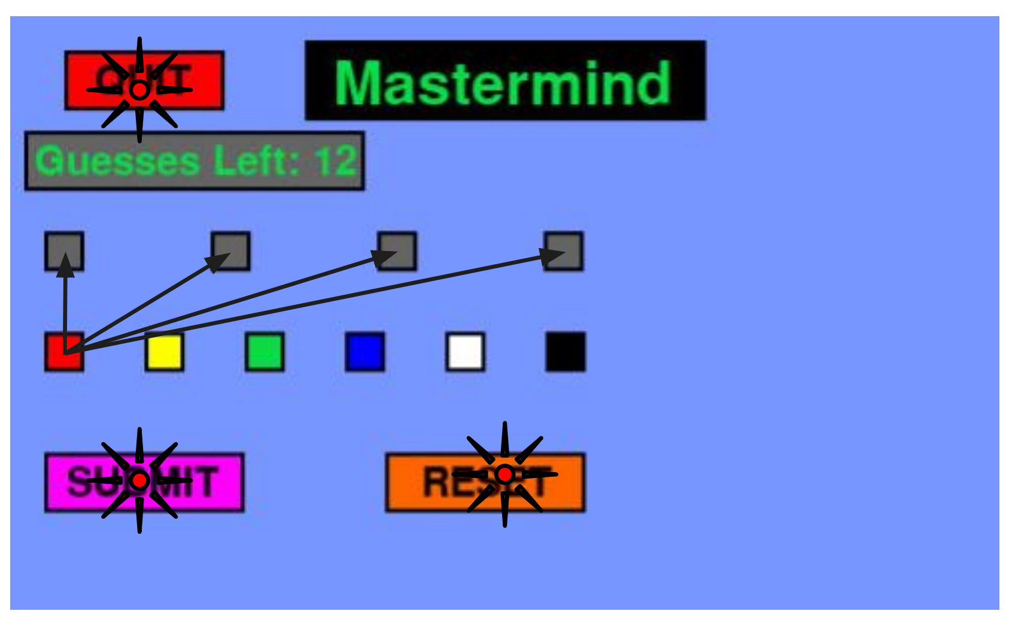
quit():
This function ends the game (and by proxy the program). This function is used in the quit buttons onClick method.

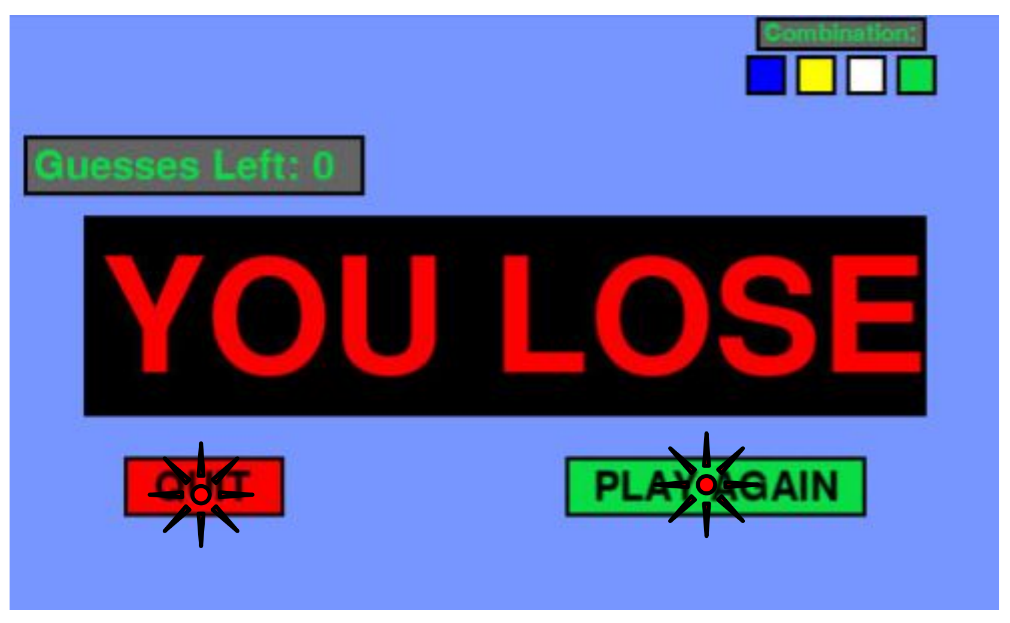
playAgain():
This function deletes all existing objects and then starts a new game. This function is used in the play again buttons on click event.

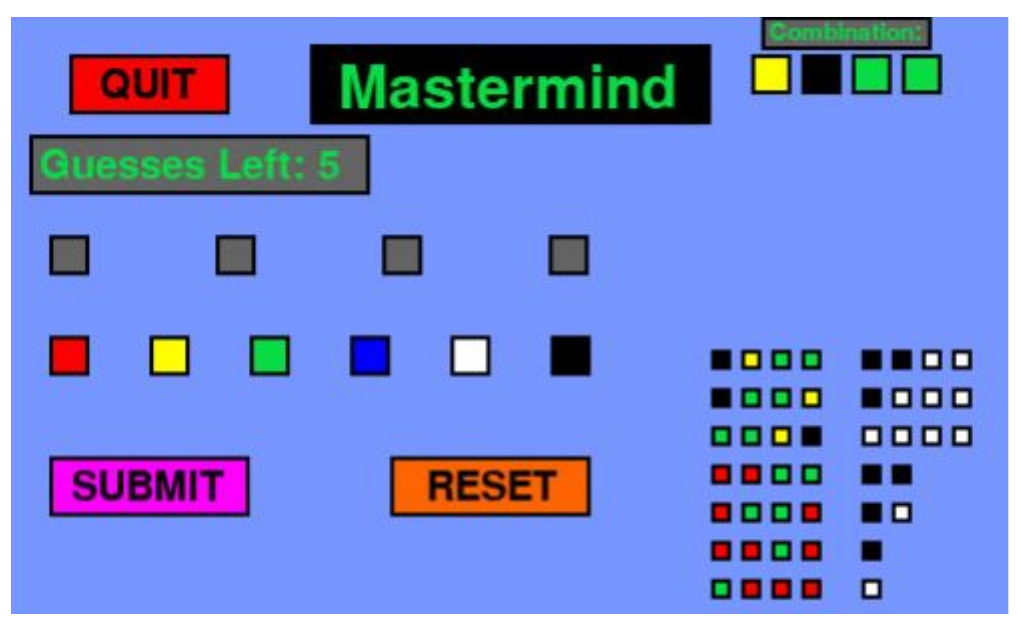
Testing

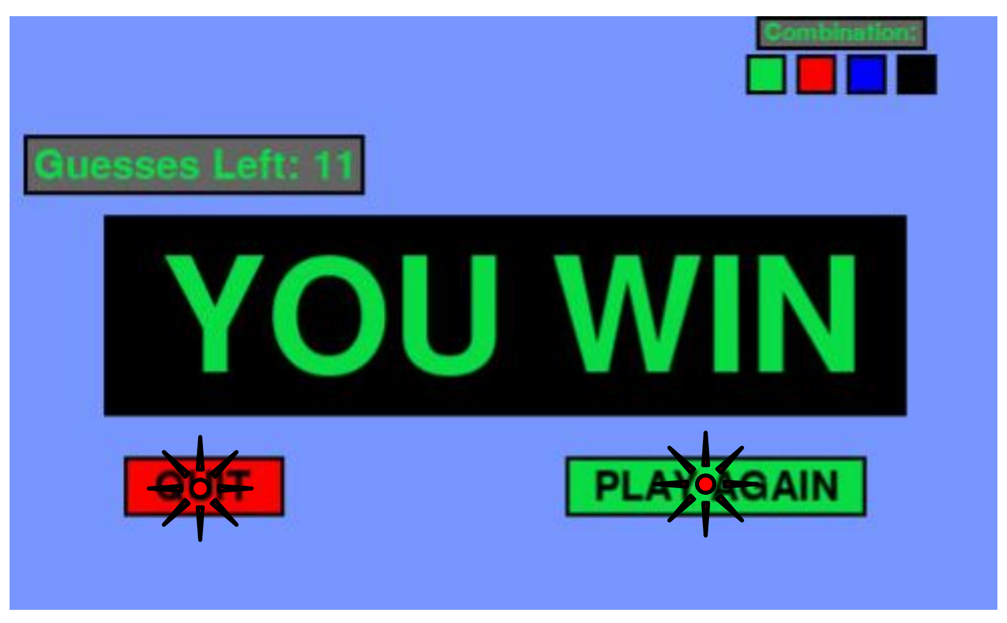
Initial testing involved testing out all of the interactable objects to see if they performed as expected. This meant testing the drag and drop and click and drop systems for all of the active pins and all of the guess pins, making sure that active pins were highlighted correctly, making sure that all of the buttons worked as expected and making sure that the game could be won and lost.

The next step was to test the program's logic. The easiest way to do this was to have the program print out the winning combination early so that I could make guesses that would test for specific scenarios and more easily see if the program had shown an incorrect combination of check pins.









Ben Hayter
A Level Computer Science
Year 12