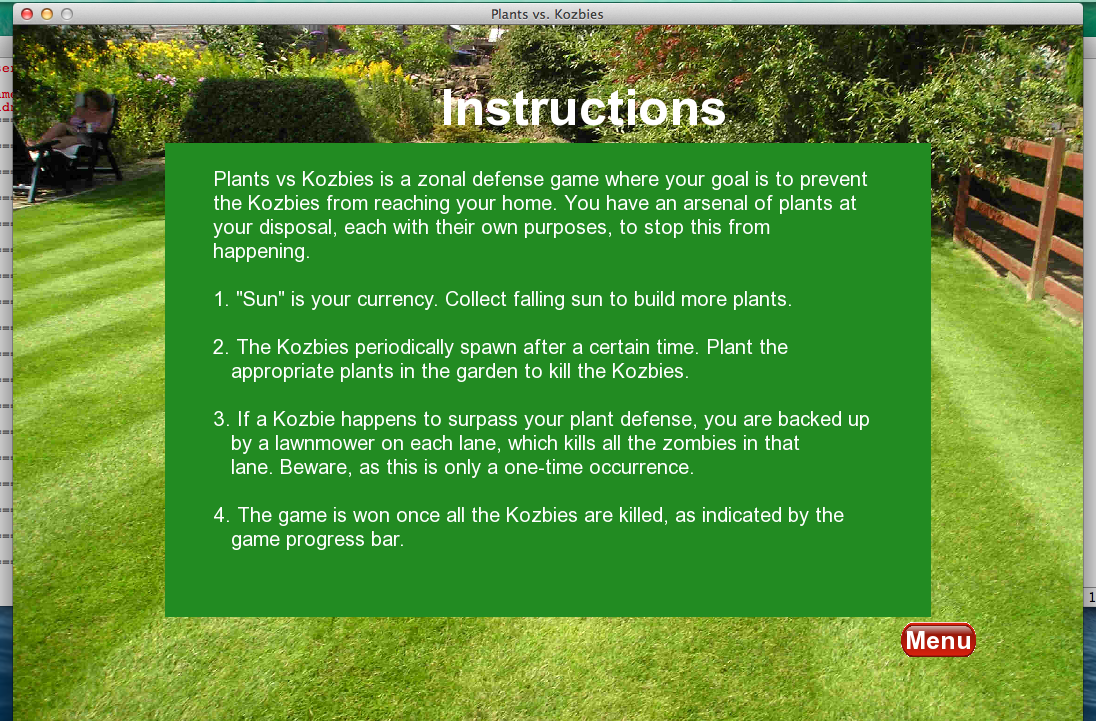
In this term project, I am designing a game where Kozbies spawn on the right side of the screen at intervals, and they move towards the left of the screen. The user may place plants in the garden to prevent this from happening by killing all the zombies. If a zombie in a particular lane happens to reach the end of the screen, a lawnmower in the same lane as that zombie activates and moves along the lane killing all the other zombies in that lane too, as a last defense. To build plants, the user must collect sun, which can either be collected from sun dropping from the sky or harvested from sunflowers. The player wins the game if the progress bar at the top left corner fills up, which indicates when the zombies stop spawning and when the game ends.

When the user opens the program, he will encounter a menu. To create the menu and its selections, I used a variable (data.phase), which was equal to “menu” for the menu screen, “game” for the game screen and so on. If the user clicked on a menu option, the (data.phase) would be set to the corresponding string. I implemented “if” conditionals in my core functions (mousePressed, timerFired, redrawAll, keyPressed), to tell them which screen to perform the specific actions on. The “play” button takes the user to the game, the “instructions” button takes the user to the basic instructions of the game, the “rules” button takes the user to the rules of gameplay, while the “quit” button exits the program.



For the user interface, I “blitted” an edited image onto the screen as a background, that contains the game name and some graphics, along with the menu options. The options are displayed as white text, but when the user hovers over them it turns blue. To do this, used “if” conditionals and tracked the mouse position.

When the user clicks on the “instructions” button, (data.phase) is changed to “instructions”, and the user is taken the instructions screen, which is made using a helper function (drawInstructionsScreen), called in the redrawAll function.



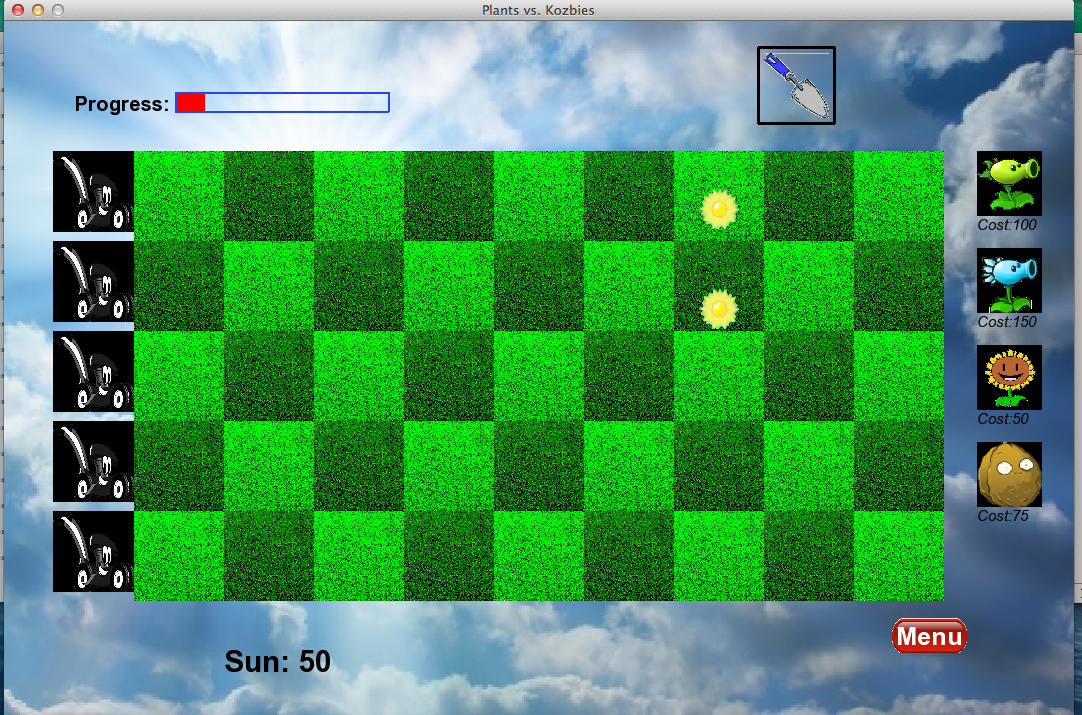
To create this screen, I blitted an image onto the surface data.screen, and on top of it drew the text. To draw the green text body, I used a class “TextRectException” taken from the pygame website (cited in code). Also, in all of the screens (except the menu screen), I added a “menu” button that, if clicked, takes the user back to the main menu. To do this, I added a function that tells the program what to do if the button is clicked, and another function to draw the button on each screen.

When the user clicks on the “rules” button, (data.phase) is changed to “rules” and the user is taken to the rules screen, which is made using a helper function (drawRulesScreen) that is called in redrawAll.



This screen was made in the same way that the Instructions Screen was made.

Finally, if the user clicks the “play” button, (data.phase) is changed to “game”, and the game is started. The user is then taken to the game screen.



In my game screen, I have several features. Firstly, at the bottom left, I display the amount of sun currency that the user currently has, which he needs to plant plants in the garden. I used string formatting to ensure that the currency automatically updated itself. To obtain more Sun, the user has to click on the falling Sun or the Sun generated by the sunflowers. To implement this idea, I created a class “Sun” that inherited pygame’s Sprite module. I also created functions to generate sun at intervals from the sky that fall down (constantly updating the sun’s rect position, a feature made easy by pygame’s Sprite module), and sun that spawns from sunflowers. If the user clicks on the sun, that sprite is killed and the currency is updates.

To implement the plant selections at the right of the screen, I drew a single-column grid with the number of rows corresponding to the number of plants, which were in an initialized dictionary. I used a function that tracks the users clicks and updates a variable that stores the selected plant, so that the next click can place the plant on the screen. An error message is displayed if the plant is too expensive. The spade tool at the top allows the user to dig out a plant with no refund cost. It uses the same idea as the menu button and the plant selection functions.

I also added a progress bar to show the user how close he is to winning the game – if the bar is full, and no zombies spawn, the game is won. The bar is updated by equating the bar width to a function of timerFired’s data.counter.

The plants, zombies and lawnmowers are created using pygame’s Sprite class (so each object is a ‘sprite’). Each sprite has its own attributes, that can be used in the helper functions used to create the objects and make them interact. To track the peas hitting the zombies, the zombies eating the plants and the lawnmowers killing the zombies, I used pygame’s Spritecollide function, which checks if two sprites are in collision. Also, I created the subclasses Peashooters and Sunflowers under the class Plants, to give them different attributes (shooting peas and spawning sun).