Description: Essentially, my CS106 final project (iSpy City), is an I spy game. If you are not familiar with I spy games, they are basically games in which the player must search for and find listed objects one after the other in any given scene in order to win. For instance, an I spy game could ask you to find certain tools in a toolbox scene, filled with several other tools or miscellaneous objects in order to make the game more challenging.

Inspiration: My idea for creating iSpy City, was heavily inspired by the several I Spy computer games made by Scholastic that I used to play as a young kid. I can remember how much joy those games brought me, as a result of their magical gameplay and immersive settings; so I wanted to try and replicate that feeling with my final project.

Instructions: When the "CS106_FinalProject" sketch is first run in processing, the user will see a window in their preferred browser, displaying the home screen of my iSpy City game. From the home screen, the user can adjust the volume of the music in the bottom left of the screen, or click the "play" button under the magnifying glass to start the game. Once the play button is clicked, the user will be brought to the game screen. On the game screen, the user will see a message board at the top of the screen telling them what object to find. Once the user finds the listed object and clicks on it, the message board will tell the user what object to find next. At any point if the user wishes, they can still change the volume of the music in the bottom left, or return to the home screen by clicking the "back" button in the top left. If the user has partially progressed through the game, having found only a few objects, and they wish to return to the home screen, pressing the "back" button will not erase their progress; once they wish to continue, pressing the "resume" button on the home screen will bring them back to the game screen. If the user finds all of the objects, they will be brought to the end screen where they will see a congratulatory message and a "home" button. If the user clicks the "home" button, they will be brought back to the home screen where they can click the "play" button to start the game again.

Topics:

Lecture/Module 3: Input and Output - Reading and writing text and image files, sprites

Out of this topic, I utilized the concept of reading image files in sprite format. I used the copy() function along with a separate array file with coordinates named obj.js in order to split up a png file with the illustration that I made, into a grid like formation, extracting the background and hidden objects that I used in the game scene.

Lecture/Module 4: Advanced Shapes - beginShape(), vertex(), endShape(), PVector and polar coordinates

From this topic, I focussed on the beginShape(), endShape(), and vertex(), functions in order to create complex shapes of my choosing. With these functions I created the body of the car, as well as the shrubs behind the buildings in the game scene.

Lecture/Module 5: User Interfaces and the DOM.create, createElement, createSlider, createInput, and createRadio. Dot style (i.e. .style) and dot html (i.e. .html)

In this topic, I made use of various types of content in order to solidify the navigation and functionality of the game. I used the createButton() function to make the navigation buttons (“play”, “resume”, “back”, “home”), and the createSlider() function to create the music volume control. To customize the buttons, I used multiple instances of the dot style presets, such as ‘font-family’, ‘font-size’, ‘color’, and ‘background-color’.

Lecture/Module 6: Geometric Context - push(), pop(), translate(), rotate(), scale(); hierarchical modelling

Concerning this topic, I used all functions in the geometric context except rotate(). With push(), pop(), and translate(), in small instances I was able to easily move groups of shapes in user defined functions that made up separate graphical symbols, such as the magnifying glass and music note. In larger instances, I used all of said functions plus scale(), to reflect and move several groupings of shapes in order to hierarchically model the car.

Lecture/Module 8: Randomness and Noise - random(), randomSeed(), noise()

Regarding this topic, I employed the random() and noise() functions both with small to medium implications. With the random() function I randomized a congratulatory ending message given to the user when they complete the game, by means of cycling through an array of strings based on a value from one to three. With the noise() function, I used a simple loop, with properties including amplitude, spacing, maximum, and minimum, to determine a randomized path based on the decided values for the creation of the shrubs behind the buildings.