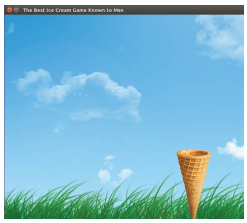
Kyle Combes and Allison Basore
3/8/2017
Project 4: Software Design

Project Overview:

Our project features an interactive game where a user controls (with the keyboard) an image of a cone that move left and right along the bottom of the game window. As the player moves the obstacles, their goals are to catch scoops of ice cream that are falling from the sky and avoid obstacles. The obstacles include leaves, drones, balloons, bees, and asteroids. As they ascend further into the atmosphere, the obstacles and background change. In order to win, you have to make it up to Mars.

Figure 1

Figure 2



Results

The game challenges the user to navigate past several obstacles that fall from the sky. For example, Figure 1 shows a drone obstacle. A major part of the user experience is the background that changes as the user advances in the game. The sequence of backgrounds is: ground level, atmosphere, early space, deep space, and Mars. Figure 2 shows ground level. Figure 3 shows mid-atmosphere. Lastly, Figure 4 shows the final stage of the game.

Figure 3

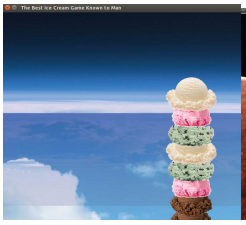
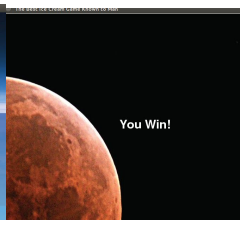


Figure 4



Another major part of the game's challenge and charm is the random movements of the obstacles. The obstacles fall from the top of the screen at random places along the x-axis. The speeds in the x- and y-directions are also random. So, some obstacles move diagonal. You never know how fast and from where the obstacles will be coming. The same randomness controls the scoops that fall from the sky. As they fall, the user is challenged to maneuver around the obstacles while collecting the scoops of ice cream.

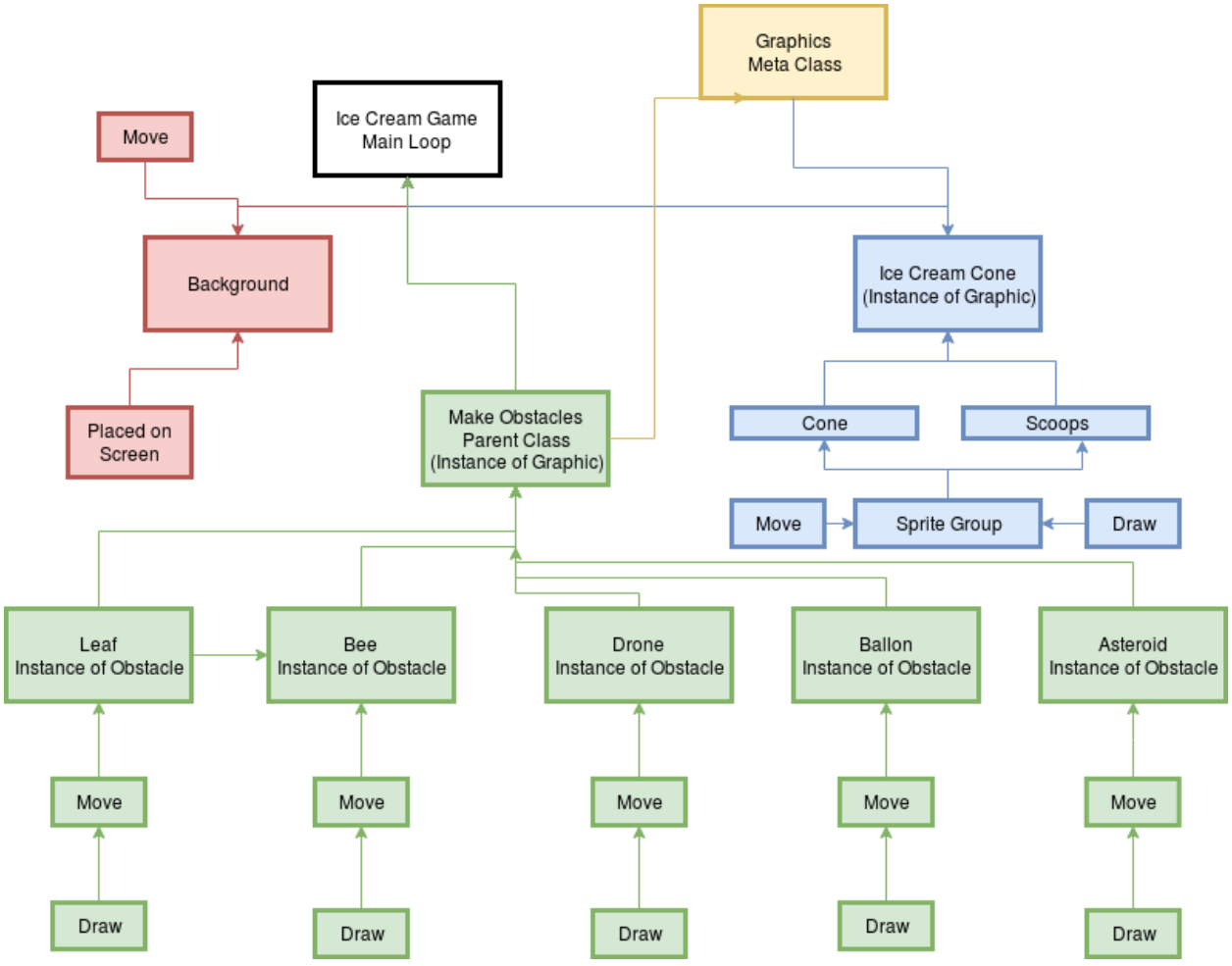
Implementation

The game operates mainly with the main game loop. Before the main loop starts, several parameters are set that do not change during the game. Things like game speed in frames per second and creation of obstacle instances. Before the loop is also where pygame is initialized.

As long as the game_quit variable is set to False, the loop runs indefinitely. During each pass through the loop, the user inputs are checked (arrow keys), the background is re-set, the cone and scoop sprites are re-set, and the obstacles are placed on the screen. These objects are all instances of the abstract class `Graphic`, which requires an x and y starting position and a `move` function. The obstacles likewise extend an abstract parent class called `Obstacle`. This class requires an x position, a y position, a `move` function, and a `draw` function.

In the later part of the game loop, there is the event collision handling. Since the location of each sprite is given by the point of the top left corner, we needed to set a range of values that could intersect to signify a collision. The top scoop and each obstacle is given a range of x values and if any numbers in these ranges collide, the game loop is set to False and the game quits. Figure 5 shows a UML diagram of the game.

One of the design decisions we had to make was whether or not to make the background a series of pictures or one large picture. There were pros and cons to both options. If we made the background one large picture, it would make things a bit more confusing for where to start the objects since the first screen would need to be at the bottom of the picture (at y = -4400). Another downside is that we would have to spend a large time investment for editing images to build one large picture. A plus would be that it would be easy to change the range of the background shown on the screen as the game progress. A minus to making several small pictures is that they would appear choppy to the user. Overall, we decided that aesthetically it was important that the background should not be choppy and therefore we decided to make the background one large picture.



Reflection

Figure 5

We would consider this project to be a success because 1) we built the game as originally idealized and 2) we gained significant programing/pygame experience. This project worked well because there we knew enough of pygame to work efficiently, but were still challenged by learning new aspects. A couple notable learning points were:

1) Class inheritance
2) Abstract classes
3) Handling user events
4) Collision handling/dealing with image positioning

As a team, we worked well together. It was established in the beginning that Kyle had more programming experience and skill than Allison did. With this knowledge, we delegated parts of the project to challenge both of us. Kyle also provided helpful explanations of abstract classes and inheritance in the beginning of the project. Due to unavoidable circumstances, we had limited meeting time. So, having more meetings would be the main improvement for next time. Overall, we had no partnership issues.