Interactive Programming: Snake Game
Junwon Lee, Cusai Alfred-Igbokwe
03/16/18

Project Overview:

For this project we created a traditional snake game, where the player's goal is to score as many points as possible by catching the mouse. The player immediately loses the game if the snake collides with either itself or the boundary. To create this game we imported pygame and applied collision detection and keyboard input.

Results:

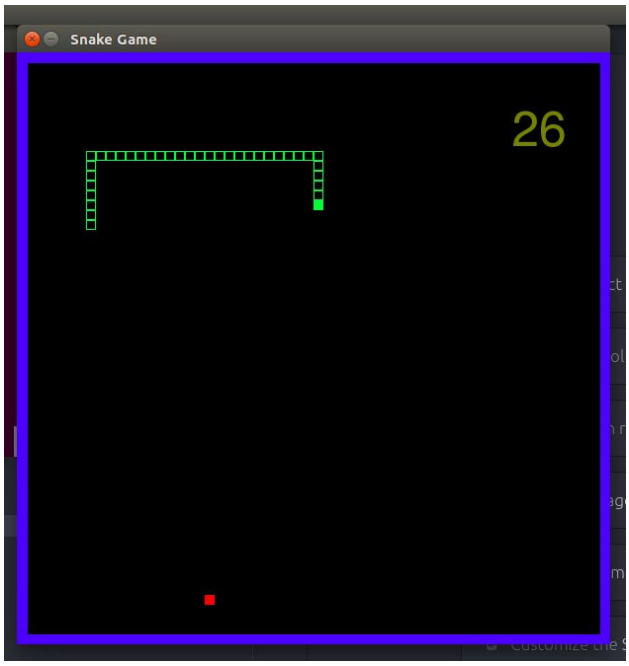


Figure 1. Game Screen

The figure above shows the game window that contains all game elements. The green chain of rectangle represents the snake, in which the solid green block indicates the head of the snake and the rest indicates the body segments. The motion of the snake is controlled by keyboard arrows, and all the body block sequentially follow the head's path. The red rectangle represents the mouse of the game, and every time the head of the snake completely overlaps with the mouse, the snake will gain an extra block to its tail, and the mouse will be randomly placed in the arena.

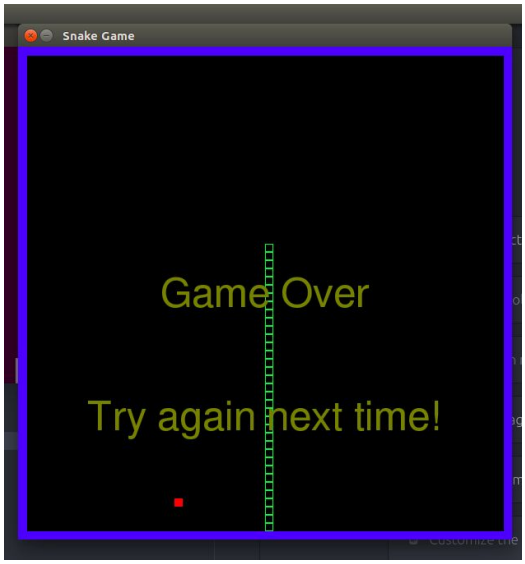
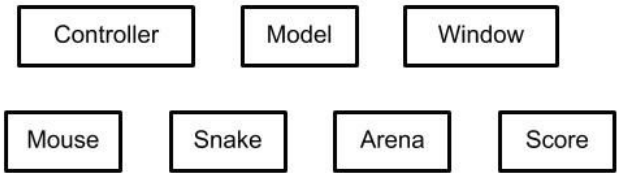


Figure 2. Game Over Screen

When the snake collides with itself or the boundary, the game over screen will pop up, which is shown in the next figure. The window will encourage the player to play the game again with the quote “try again next time”. To play the game again, the player would have to quit the window and run the game again.

Implementation:



The UML diagram above represents the classes that were implemented for this game. Classes were created for visible elements in the game: the snake, the score, the mouse, and the boundary. The controller class handles the keyboard commands that must be obeyed. For example, while the player is playing the game, the keyboard arrows change the x and y velocities of the snake head. The model of the game created lists that contained objects that had to be drawn, and it instructs how each game object should be updated. For instance, every time it updates, the snake body segments should take over the position that is in front of it. Furthermore, the update contains the command that terminates the snake’s motion and prints ‘Game Over’ when the snake collides with the mouse, boundary and itself.

Once we have completed all the model and controller, we created a class for the window view that drew the mouse, the snake, and the boundary on the window. Then, we have created a

main function such that the controller gave command to the window, the model receives the command and decides the appropriate reaction to the command, and the window displayed the appropriate model onto the screen.

One design decision that we made was to create a new class specifically for the score value. We were wondering whether an extra class was needed for the score. We initially thought that it was unnecessary primarily because the score is essentially a numerical value, but we later realized throughout the process that leaving the score a numerical value makes the model function much more complicated, especially in the update function. Therefore, we believed that it was necessary to create a class specifically for the score value.

Reflection:

We believe that the thing that went well the best was that the tasks that wished to accomplish were done before deadline. We are happy that we got all the necessary game elements and the scoreboard into the game. Our project was appropriately scoped for our skill level and we were able to complete the game with moderate difficulty. Despite that we have struggled to understand pygame initially, studying pygame tutorials thoroughly helped us create necessary figures in the window.

We believe that we evenly divided work among each other such that we could complete the tasks on time. We messaged each other whenever we completed our assigned task so that we could prevent merge conflict. One flaw in our team process was that we did not communicate with each other in the first week of the project, primarily because we both were learning pygame mechanisms. We believe that if we communicated with each other more in the beginning, we could’ve learned pygame mechanism faster and create features faster than we did.