Mini project 4: Interactive programming
Harris Davidson and Viktoria Wastring
10/29/2017

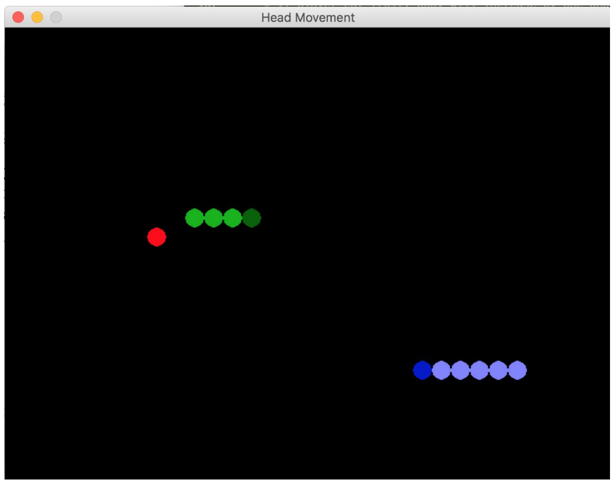
Mini Project 4: Interactive Programming

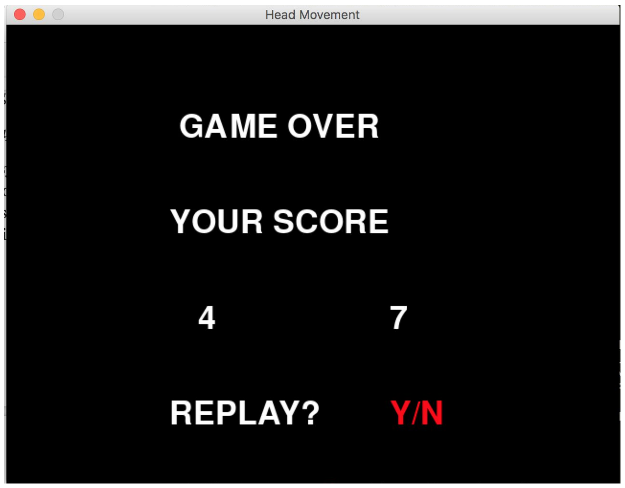
Write-up and Reflection

Project Overview

We re-created the old school game of snake but with a twist. The control scheme game play of our version mirrors it's 8-bit counterpart, but adds competitive play to the mix. Two players go head-to-head to build the longest snake by collective fruit and avoiding collisions.

Results



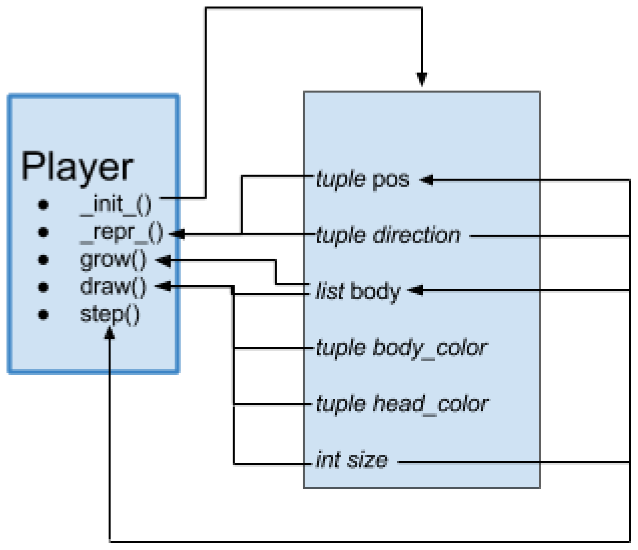


The result of our project is our version of snake. When starting the game, three dots show up on the screen, one red, one blue and one green. The red dot represents food, which is the goal you want to get to. The green and blue dots represents player 1 and player 2. The players does not start to move until you initialize the movement with the keys (w a s d for player 1 and the arrows for player 2). The goal is to "eat" the red dot (food) to grow longer and therefore get points. Once one of the players eats a dot, their body increment by one element and a new red dot appears on the screen. The goal of the game is to grow as long as possible without hitting your own body or hitting the sides of the frame. When someone loses, the scores are shown and you can decide to play again by pressing y for yes or n for no.

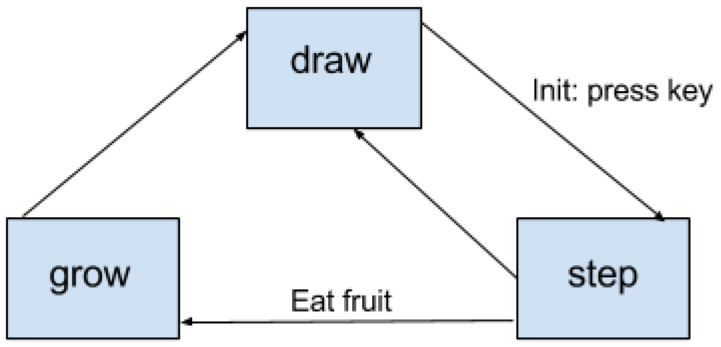
Implementation

We built our program with a player class, which wraps the information and instructions to run and build the players.

The Model View Control explained in class seemed like a good way to implement games, but we decided to implement one class and write code in a script. Our code implemented many of the same philosophies of MVC, but we had control in the main script and had model and view rolled into the player class. This structure matched the simplicity of the game while allowing for our planned extension into multiple players. In our case, this was a good decision since when we chose to implement a second player, it was done very easily.



Interaction between Player class functions and variables.



Interaction between functions in Player class.

The position of the player head is stored in a tuple *pos* with coordinates x, y. The direction the player is moving in is also stored in a tuple (which can be altered in the main script based upon control input from the users). The head position is stepped by assigning the tuple to the piece wish sum of the head position and direction.

The body of the player is stored as an ordered list of body positions. Each step of the player we pop the oldest element of the list and append the current position of the head. To grow when eating a fruit, the list is append with the head position twice. We choose to define colors and directions as constant tuples in order to make the code easier and more natural to read and write.

Collision detection occurs at the script level and functions by checking for the position of the head within it's own body and the bounds of the screen. Snakes can run through each other, but hitting oneself or the screen boundary ends the game.

Reflection

We think the project went well. We consider it a good project which was appropriately scoped since we wanted to create a basic game that was working first, and after that add on different features to the game making it more fun. Our way of working was that we built the basic functions first (like setting the screen, adding a head of a player, controlling the player with keys and simulating the length by adding length when pressing a key) and checked if they worked by running the game. This way we knew what things in the game worked and it was easier knowing what went wrong. After building the basics, we put code into classes and started structuring it more.

We liked working on the project and scheduled time to meet and work on the project together. We primarily paired programed though sometimes worked alone to refactor or flesh out elements. Pair programming allowed us to diagnose issues in new features as we were creating, and plan our extensions strategically to match our interests.