SHIFTER

Dalton Harenza

CSE321

University at Buffalo

djharenz@buffalo.edu

Problem Statement

The Problem: To create a fun and simple arcade style game where the focus is on gaining the high score, complete with graphics, music, and a method of tracking the highest score. The program will need to be a realtime system so user controls as well as collision detection and score calculation can happen with no noticeable delay to the player.

The Solution: Shifter is an arcade style game in which the player controls a space ship with the ability to switch between three lanes and two colors. In each of these lanes appears colored lasers that approach the user at a rapidly increasing speed. The player's job is to pass through these lasers while being the same color to gain points and increase the multiplier. If a user hits a laser that is a different color than they currently are, the game is over. Players can choose to speed up the game by hitting the up arrow key, which will add a factor of 5 (max speed is 15) to the speed and increase the player's multiplier by 3. If the speed becomes too much, players may choose to hit the down arrow key to reduce the speed to 2/3 the current value, causing them to sacrifice a factor of 5 on their current multiplier. (I.E. A player is at speed 15 with a multiplier of 24. Upon hitting the down arrow, the speed will be reduced to 10 and the player's multiplier will decrease to 19) In order to gain the highest scores, users must know when to slow down the game in order to keep the speed at a manageable level while making sure they let the multiplier build high enough before slowing down again.

Motivation

The motivation for this project can be split into two main reasons; first, to gain experience with game design using a library to assist with development and second, to learn a new programming language (ruby) and use it to create a full program. This is my first attempt at game design and programming and has quickly taught me a lot about the process and difficulties that can arise. My hopes for the future of the program involve using the game's core features to create an android app. Additionally, I would like to flesh out the game more with more game modes, unlockable content, a more-realized leaderboard, a multiplayer aspect, and more.

Design/Architecture

Shifter is coded in ruby with assistance from the Gosu gem (roughly equivalent to libraries in most languages.) The project consists of a main.rb, GameOver.rb, GameWindow.rb, Map.rb, Player.rb, TitleScreen.rb, highscore.txt, music, and graphics. The functionality of these is as described below:

main.rb : Sets up dependencies and creates the game window

GameWindow.rb : Creates the main game window and map and handles user inputs

GameOver.rb : Contains logic for what should be displayed upon a game over

Map.rb : Creates the player and lasers and handles the game state, collision detection, and score calculation

Player.rb : Contains information about the player sprite and location

TitleScreen.rb : Contains logic for what should be displayed before the game is started

Highscore.txt : Keeps track of the highest score obtained

Music : Used to play an mp3 while the game is in progress

Graphics : Contains all images used for the game

Technical Description

Shifter is coded with Ruby, a dynamic, open source programming language that is relatively lightweight and highly readable, similar in many ways to python. Many elements of the game are handled through the use of Gosu, a Ruby gem that handles aspects such as the UI, music, sprites, and event handling. The bulk of the event handling happens within the maps update function, which is a loop that is repeatedly executed until the user exits the program. Within this method, the game listens for user input and acts accordingly, moves the lasers down the screen, randomly draws new lasers if necessary, and check for collision.

The check_collision method handles the actual detection of collision by comparing the player's location to the location of the lasers on the screen. If the player occupies the same area on the screen as the laser, the method then checks if the player's color matches the laser's color. If it does, the player's score and multiplier are increased. If not, a game over is triggered.

Upon game over, the map's update method calls on GameOver.rb to properly display the game over text and awaits user input to start a new game. Upon the user hitting their space bar, the map will call initialize again and begin the process anew.

Pseudocode for the main loop can be found on the next page

```
def update
    if  not started
        if (user input)
            start
        end
    else if
        if (user input)
            act accordingly
        end
        if (lasers past end of screen)
            randomly generate new lasers
            increase speed
        else
            move lasers
        end
        check_collision
    else
        display game_over
        if (user input)
            initialize
        end
    end
end
```