Final Report

Project Analysis

Turlington Turmoil is a game representation of the congestion in the Turlington area on campus. The user is an orange and blue skateboarder who has to zigzag through the oncoming walkers and bikers. Watch out, because every now and then you will encounter a super fan, a.k.a. a Gator. The objective of level one is to reach the left side of the screen from the right side as many times as possible, within thirty seconds, without touching or invading the personal space of oncoming traffic. If the user does come in contact with the traffic, the skateboarder returns the far right of the screen. Every time the user makes it to the left side of the screen he/she gains one point and is placed back on the right side. Level two incases the same parameters, expect the on traffic increases there speed by a multiple of two.

* User’s input and game reaction
  + Input direction arrow key and the skater moves accordingly
  + Menu Screen buttons can be clicked to change the screen to level 1, level 2, or the quit screen.
* Outlines of what you want to accomplish at each step and the corresponding timeline.
  + If Level 1 is chosen, the objects move at a low speed
  + If Level 2 is chosen, the objects move at a high speed
  + The skater must make it to the other end of the screen (Turlington plaza) while dodging oncoming pedestrians, cyclists, and large gator heads.
  + Pressing the arrow keys (UP, DOWN, LEFT, RIGHT) cause skater to slide in that direction
  + The skater gains a point for each time he can cross the plaza in 30 seconds
  + If the skater comes into contact with an opposing object (pedestrian, cyclist, or gator) he returns to the starting side of the screen.
* Decomposition of tasks
  + Create Skater Class
    - Make Skater from basic shapes
    - Use the keyPressed function to add location (+or-) speed depending on up, down, left, or right.
    - Check for collision with any array object using the dist(x, y, arrayX, arrayY) function
      * If a collision is true, the skater returns to the starting side of the screen (x=600) and remains at his y location
  + Creating Biker Class
    - Make biker from basic shapes (with a switch case randomly deciding clothing and hair color)
    - Individually initialize a new biker every 600 milliseconds
  + Creating Walker Class
    - Make walker from basic shapes (with a switch case randomly deciding clothing and hair color)
    - Individually initialize a new walker every 300 milliseconds
  + Creating Gator Class
    - Make gif image of a gator head with no background
    - Individually initialize a new gator every 2500-5000 milliseconds (randomly chosen)
  + Creating Timer Class
    - Determines if a specified amount of time (in milliseconds) has been passed to create Biker, Walker, and Gator array objects at a time variable
  + Create Music
    - Load background music and collision music
    - Implement background music when Initial level is chosen, and play over any current screen henceforth
    - Implement collision music within skater class, where collision is detected
  + Creating Screen Class
    - Menu Screen has roll-over buttons to change screen and array object speed when clicked (as well as restart time and score)
    - Game Screen displays Biker, Walker, Gator, and Skater class, showing them interact via collision
      * Game Screen Also displays the players score, time left, and a roll-over button for prematurely end the game
    - End Screen displays Top score, Player score, and gives the user the option to quit or play again
      * If the user chooses to play again, they return to the start menu and are prompted to choose a level
      * If the user chooses to quit the game, a final ending game screen is displayed
  + Compile
    - Initialize arrays
    - Initialize timers
      * Three separate timers will be initialized to allow for three different time variables to be implemented and related to the three different class array objects
    - Initialize score
    - Initialize Boolean value for which screen is being displayed
    - Import/initialize music
    - Import/initialize image files

Work Distribution

|  |  |  |
| --- | --- | --- |
| **David Saraydar** | **Together** | **Reggie Wilkerson** |
| Objects class for biker, walker, and gator | Fine tuning | Start/End Menu screen |
| Game Screen | Collision detection for Gator | Collision detection for biker and walker |
| Object array individually adding and re-adding array characters | Sound (background and collision) | Movable object skater |
| Final Project | Timer Class | User Manual |

Difficulties

While creating the coding for our project, all simple coding problems were relatively easily overcome. The only truly time-consuming and unsolvable problem was with the use of the Processing Tabs. Processing would not recognize any class despite it clearly being defined, when placed in a separate table from the main project tab. We discovered that the tabs were causing the incorrect error that a class had not been correctly defined through trial and error and moved all classes to one main tab.