Software Requirements and Design Document

For

Group #14

Version 2.0

Authors:
Joseph Schmitt, jcs20fh, JoeSchmitt-2
Tyler Starling, jts20cz, Tetherly
Talal Mahmoudi, tkm16c, Talal-mahmoudi
Sergio Pantoja, sp19p, sergiopan19
Nathan Zhao, nyz18, nyz18

1. Overview (5 points)

Keyboard Racer – Game where the user needs to match the timing of falling indicators to keyboard input to increment the score and streak counter with a Piano themed layout. Once the user misses an indicator, the streak counter is reset back to zero while the score stays the same, incrementing by 10 every time an indicator is hit when it drops down onto the piano keyboard.

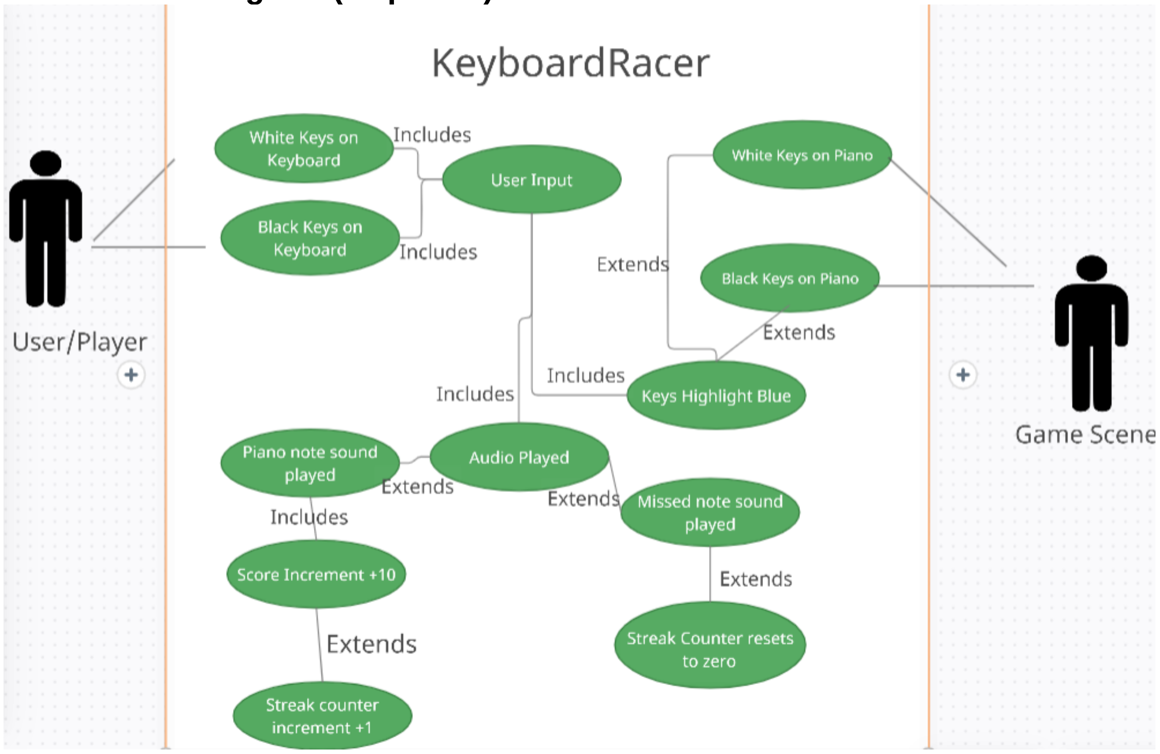
2. Functional Requirements (10 points)

When the user presses down the corresponding key must play the sound associated with that key, changing the color to blue and incrementing the score by one. Must spawn correct indicator to the associated key that must be pressed. The spawn indicator must spawn accurately and at the correct times. When the escape button is hit, it will go back to the main menu, resetting the score and streak counter for when the game is played again. Additionally, when you miss a note key, the streak will reset, and the missed note sound will be played. When you hit the note correctly, the streak will increase by 1, the score will increase by 10, and the respective audio note on the piano will be played.

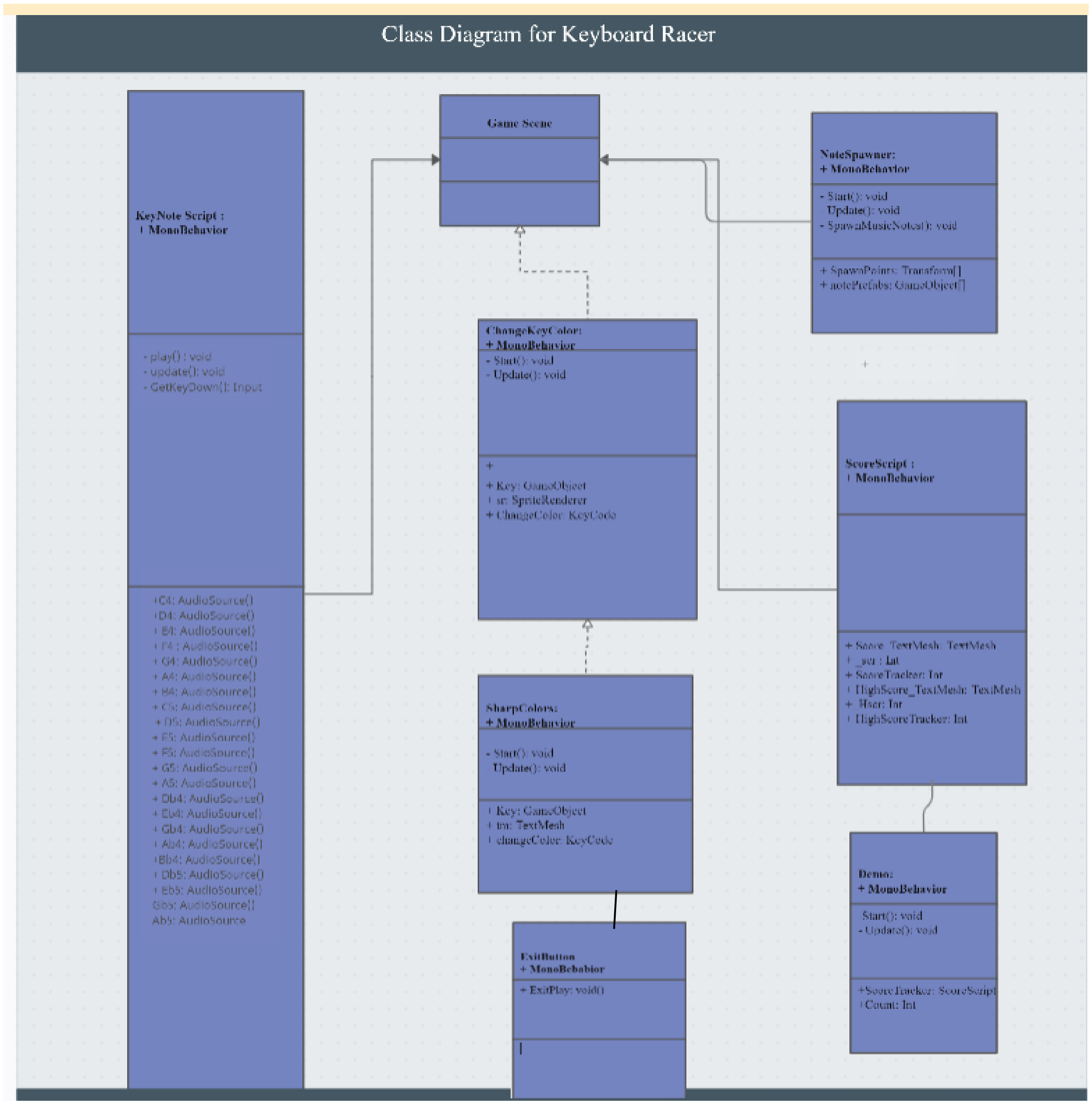
3. Non-functional Requirements (10 points)

The game must be secure. The game mustn't lag for a variety of user's systems. Autogenerated sound doesn't have to be generated if a user misses an indicator.

4. Use Case Diagram (10 points)



5. Class Diagram and/or Sequence Diagrams (15 points)



6. Operating Environment (5 points)

The game, Keyboard Racer, is intended to be played on PC with development done in Unity and C#.

7. Assumptions and Dependencies (5 points)

An assumed factor when playing the game is that the user is attempting to play the game on a ‘QWERTY’ keyboard.