CS 317 Final Project Implementation Report

# Brosu! Phase 2

Branden Guevara and Garrett Eledui

### 

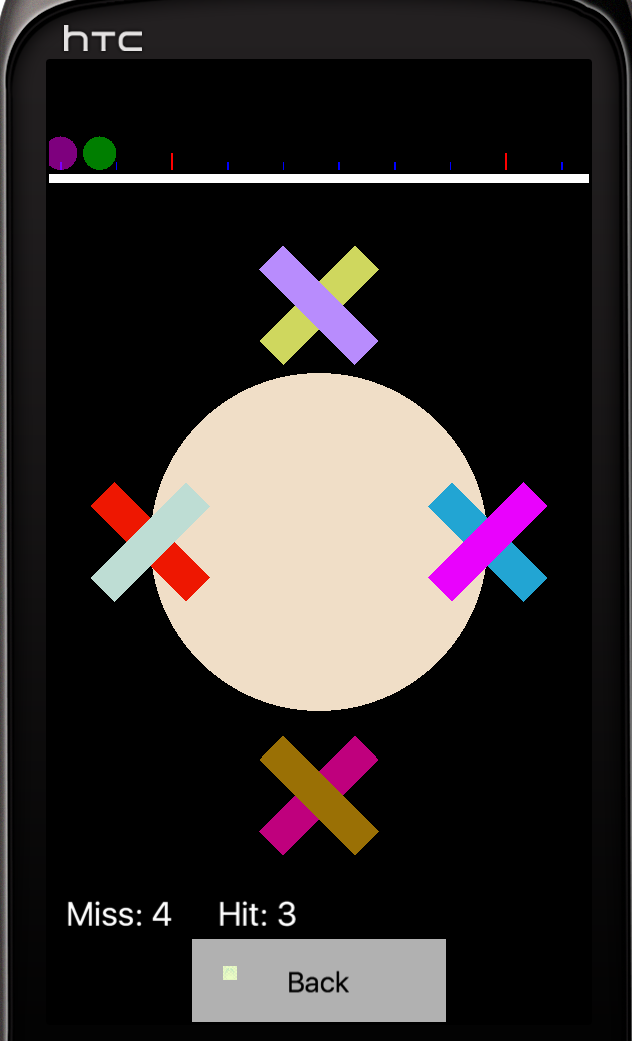
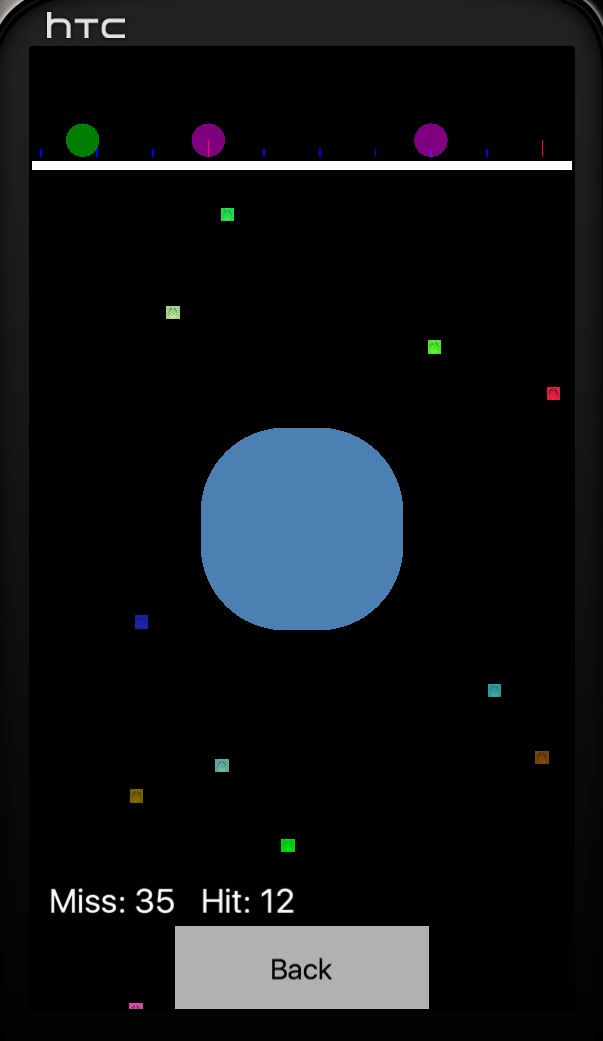
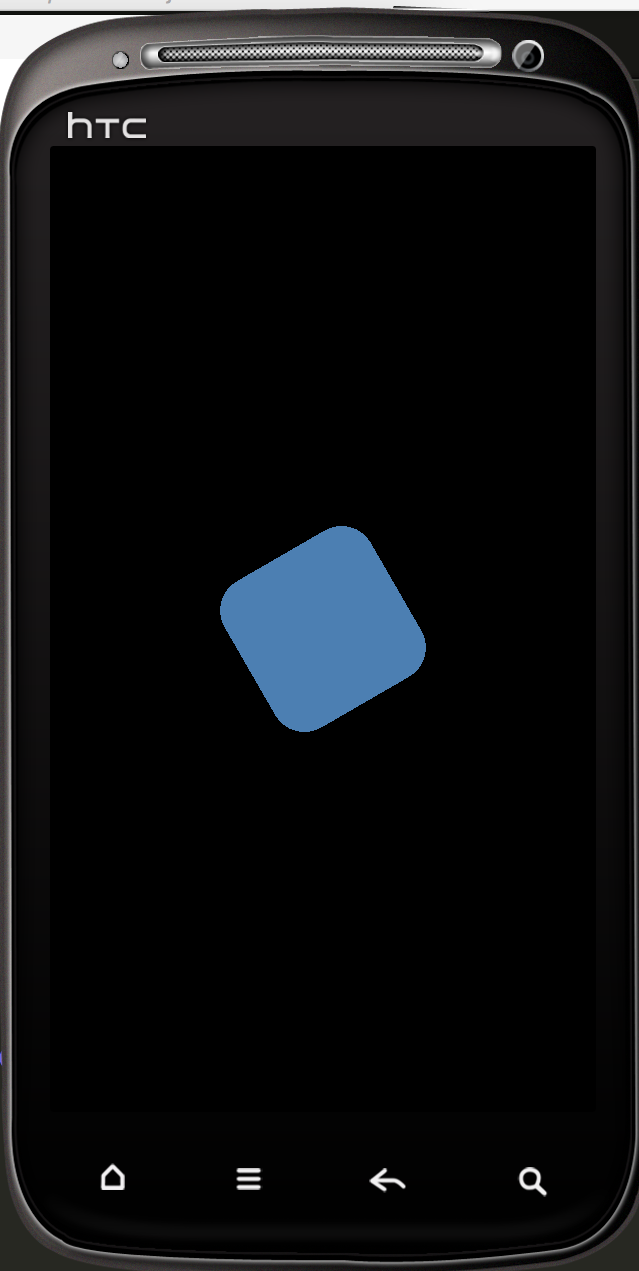
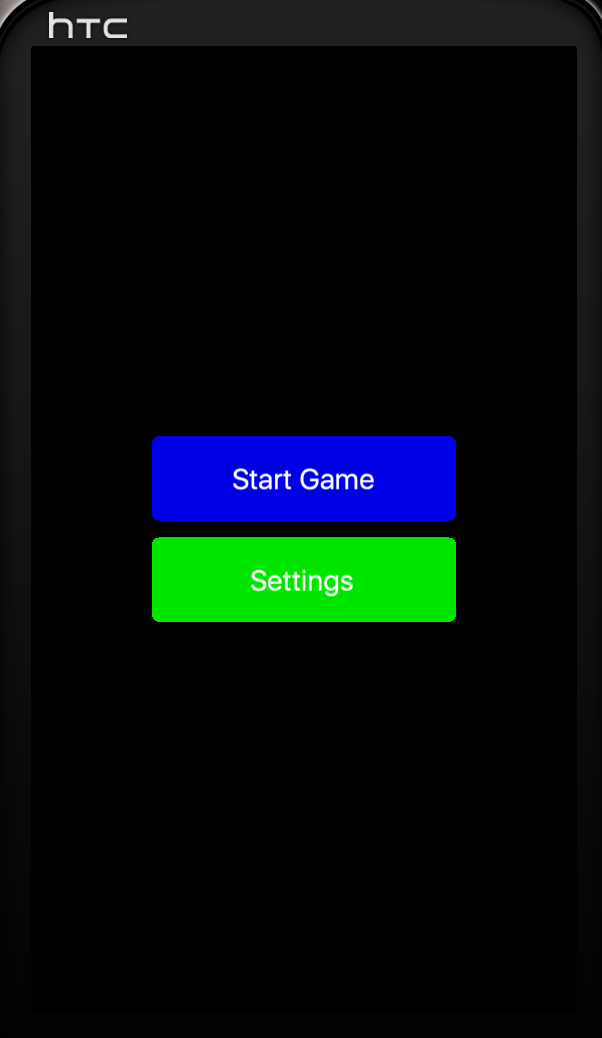
### Concept:

* Description: Our ambition is to produce an advanced and appealing game that stimulates the senses while presenting a worthwhile challenge. We seek to employ techniques learned from the curriculum taught over the semester such as graphics, object shape manipulation, and physics. The graphics will react to each of the user’s taps, gradually getting more and more elaborate as the song goes on. Additionally, we are utilizing a moving audio progress bar that will show when the next beat is approaching for the song allowing the user to recognize when to tap in order to play the game properly and with full enjoyment.
* Genre
  + Rhythm Game
  + Music
* Design
  + 2D Graphics
  + Flat Design/Minimalist
* Target audience
  + All ages
  + Music Enthusiasts
  + People who enjoy Rhythm Games
* Key Features
  + Touch screen interaction
    - For gameplay and changing between menus
  + Shape-shifting graphics
    - Occurs as user taps the center object
  + Particle effects during gameplay
  + Music
  + Audio Progress Bar
    - Aids user in indicating when to tap
* References
  + Tap Tap Revenge
  + Geometry Dash
  + Osu – Specifically with Osu’s beat mapping tool that allowed us to grab the beat times

### 

### Game Mechanics:

* Game play screens



* Control scheme interface: Our dark background accentuates the graphics that the user will interact with. This gives a pleasant environment for the user to focus on the gameplay.

screenFlow.png

Figure: Screen Flow Diagram

* Screen Flow: As shown in the figure above, the user will start at the title screen and from there can access the audio settings menu and the song selection menu. From the song selection menu, the user can select from two songs and lead them to the game menu where gameplay begins.
* Scoring system and achievements:
  + Scoring is based on how many of the beats were successfully hit out of the total as well as how many times the user mis-tapped.
* Game Flow:
  + Gameplay: After the user has selected a song, the scene will change to the game screen where the user will see a rounded rectangle in the middle of the screen as well as an indicator bar at the top of the screen where purple circles or beats will pass from right to left. As the song begins to play, whenever a purple circle passes through the green circle, it signals to the user when it is time to tap the box in the middle in time with the beat of the song. As the user continues tapping on the box, different screen effects will be triggered. A score board at the bottom of the screen will keep up with how many of the total beats the user successfully hit as well as the amount of times the user incorrectly tapped.
  + The graphics throughout the game will continue to grow more and more elaborate as the user taps at the correct time. If the user taps at the incorrect time, the graphics will become less elaborate. The game will continue until the end of the song regardless of the number of correct taps the user has gotten.



Figure: Game Flow Diagram

* Changes to original design:
  + Added an audio progress bar as a beat indicator for the user
  + Did not include graphic settings
  + In regards to tap accuracy, we gave a pretty wide window to tap, disregarding how close it was to the actual beat. There was no easy way to guarantee proper timing, especially when utilizing timers which we could not use to time each beat.

### Tech:

* Platform
  + iPhone
    - Resolution: 1920-by-1080
    - Display Size: 5.5 inch (diagonal)
    - Audio Playback: MP3(8 to 320 Kbps)
  + Kindle Fire
    - Resolution: 1024-by-600
    - Display Size: 7-inch (diagonal)
    - Audio Playback: MP3
* Graphics
  + Resolution: depends on device
  + Planning on utilizing Lua’s API for graphics
* Camera
  + N/A
* Control Interface
  + Touchscreen
* Game Assets
  + Songs:
    - Ame Michi by Otokaze
    - Lean On by Major Lazer & DJ Snake
* Development Resources
  + Corona SDK
  + Osu files
    - Had to read the files to get the specific beat times used in their maps
* Work Division
  + Branden Guevara
    - Implementation of game mechanics
    - Developed the indicator bar
    - Implemented handling of songs and beats
    - Developed screen/game flow
  + Garrett Eledui
    - Developed graphics during gameplay
    - Worked with particle effects
    - Worked with object transition effects
    - Tested design
* Implementation note:
  + Corona Libraries:
    - Composer
      * Used to maneuver through each screen
    - Physics
      * Used to move the audio progress bar at a rate that follows closely to the song
      * Used in the graphics including particles and other objects created
    - Widget
      * Used for creating buttons
  + Beatmap data type
    - Structure
      * Holds beat table which indexes where a beat is located on the audio progress bar
      * Holds time table which indexes the time stamp in milliseconds of each beat. This is read in from a time table text file
      * Holds the value for the quarter beat length
      * Holds the beat divisor which is used to determine the number of ticks in a whole beat
      * Holds the bpm (beats per minute) of the song
      * Holds the song duration
      * Holds the pixels per second that will determine how fast the progress bar must be moving
      * Holds the offset of the song to even out the progress bar
      * Holds the filename that holds the time table
    - Functions
      * setTimeTable will read in the time stamps from a specified file and fill in the time table array
      * setupIndicatorBar will develop the audio progress bar object to be used in gameView
      * setSong will initialize all of the members in the beatmap object to the values specific to which song was chosen
  + Data structure for floating rectangles
    - Data structure holds dynamically created rectangles



Fig. Code Flowchart