**N-Shapes**

**CS 571 MOBILE COMPUTING SFTWR ARC & DEV**

**Final Project**

**Project Members:**

Bidhya Nandan Sharma (bns0028)

Sarvagya Pant (sp0090)

Submitted on

1st May 2017

Computer Science Department

The University of Alabama in Huntsville

Table of Contents

List of Figures 3

Introduction 4

Game Architecture 5

Game Workflow 6

Team members and contributions 7

Challenges 7

References 8

# List of Figures

Figure 1 Game Scene Transition 5

Figure 2 Shape to avoid 6

Figure 3 Gameplay Start 6

Figure 4 Scoring in game 7

Figure 5 Game over scene 7

# Introduction

N-shapes is a simple car race game inspired from 2-cars [1]. Our game is targeted to everyone even though adults might find the game more interesting. The gameplay is very simple. There are two cars on different roads. On game start, we get an instruction on which shape to avoid. Player must control two cars on both side and avoid the shape instructed. However, if any other shapes are missed, the game is over.

We have used various Corona Features such as Event Dispatcher [2], Collision detections [3], Inheritance [4], etc on creating this game. The game is targeted mostly for Android devices with minimum resolution of 320\*480

# Game Architecture



Figure Game Scene Transition

Following are the game scenes that have been implemented:

1. Credit Scene: This scene shows the name of members who developed N-Shapes.
2. Main Menu Scene: In this scene, player has option to enable or disable the sound. Then s/he can proceed to play the game.
3. Game Scene: All of our game logic are implemented in this scene. Image shapes are generated randomly. Player must hit the correct shapes to increase the score.
4. Settings: In this scene, player can again enable or disable the sound and continue the game.

# Game Workflow

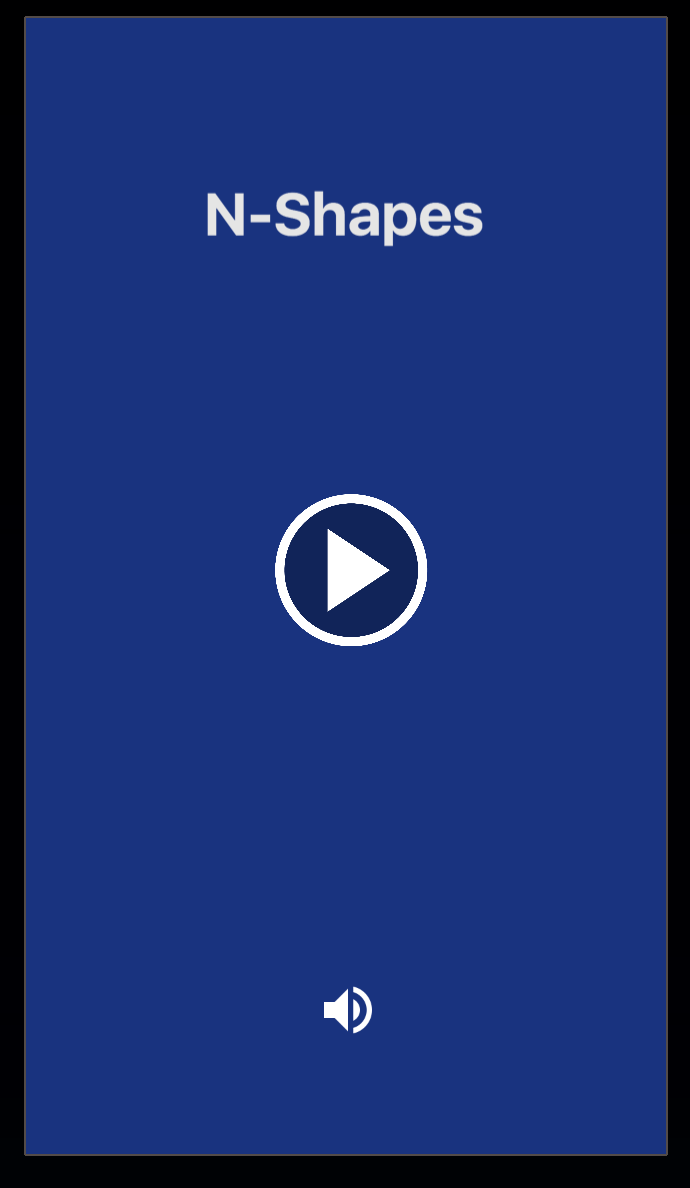


Figure Main Menu

As shown in image above, if player clicks on play icon, the game is started. The figure shown below illustrates the game flow.

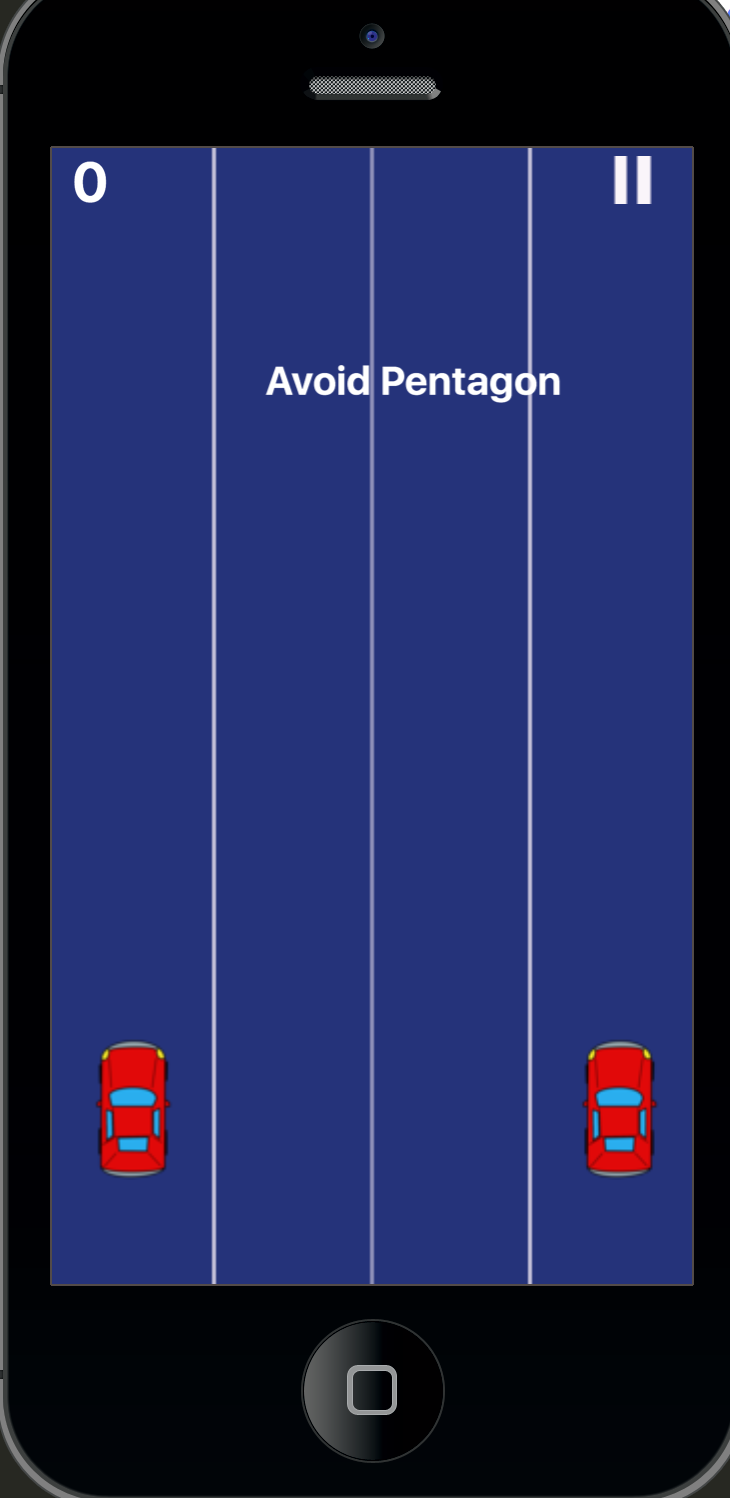


Figure Shape to avoid

Once player starts the game, an instruction is given on screen that shows the shapes to be avoided. In this case, the player has to avoid “Pentagon”. If player hits other shapes, the score increases. However, if any of those shapes are missed, the game is over.

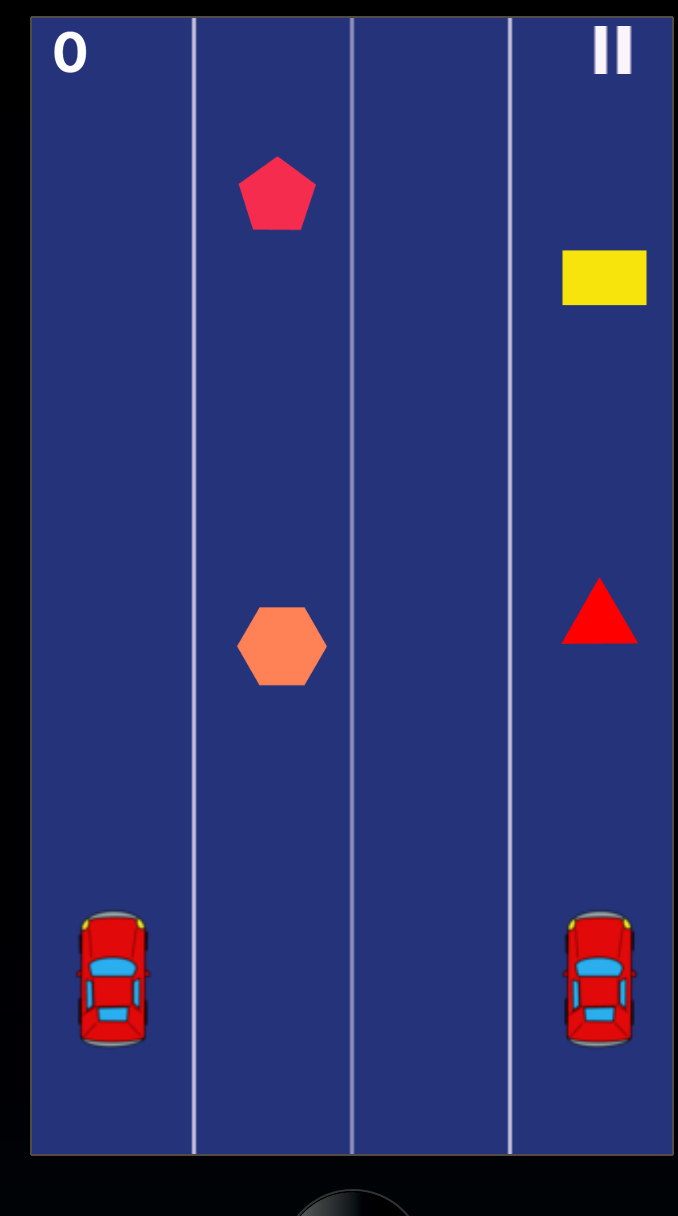


Figure Gameplay Start

Figure 4 shows the game screen when the game has been started. If correct shapes are hit, then score increases. Following figure shows the scoring:

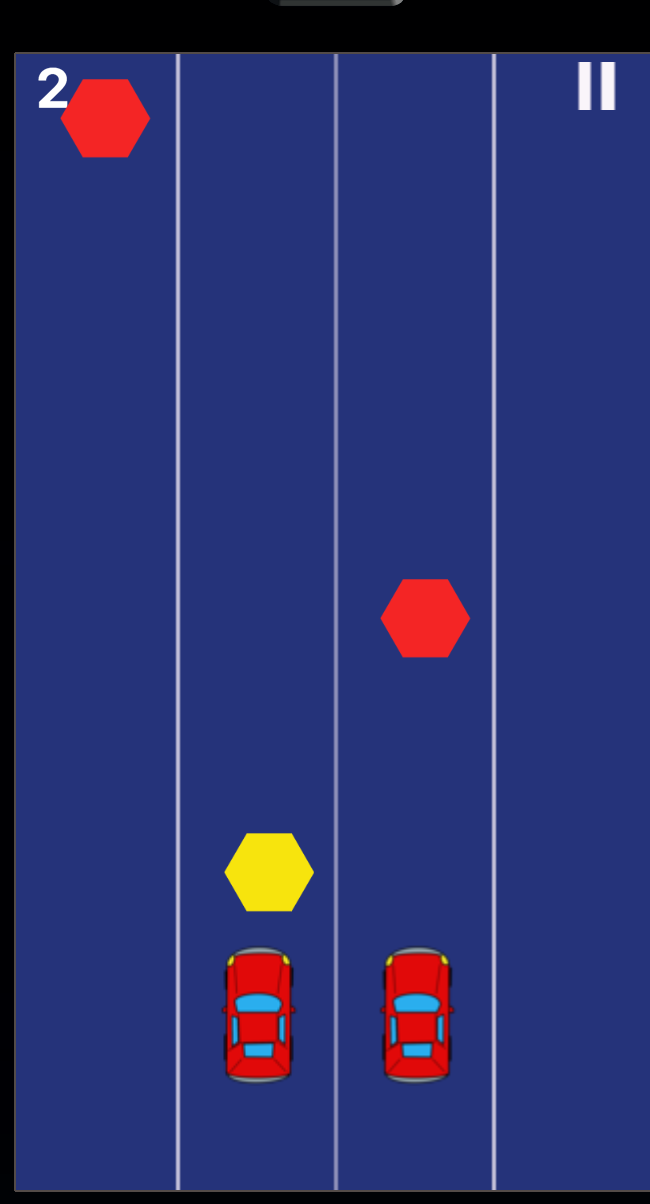


Figure Scoring in game

However, if the player misses the correct shapes, the game is over and following scene is shown:

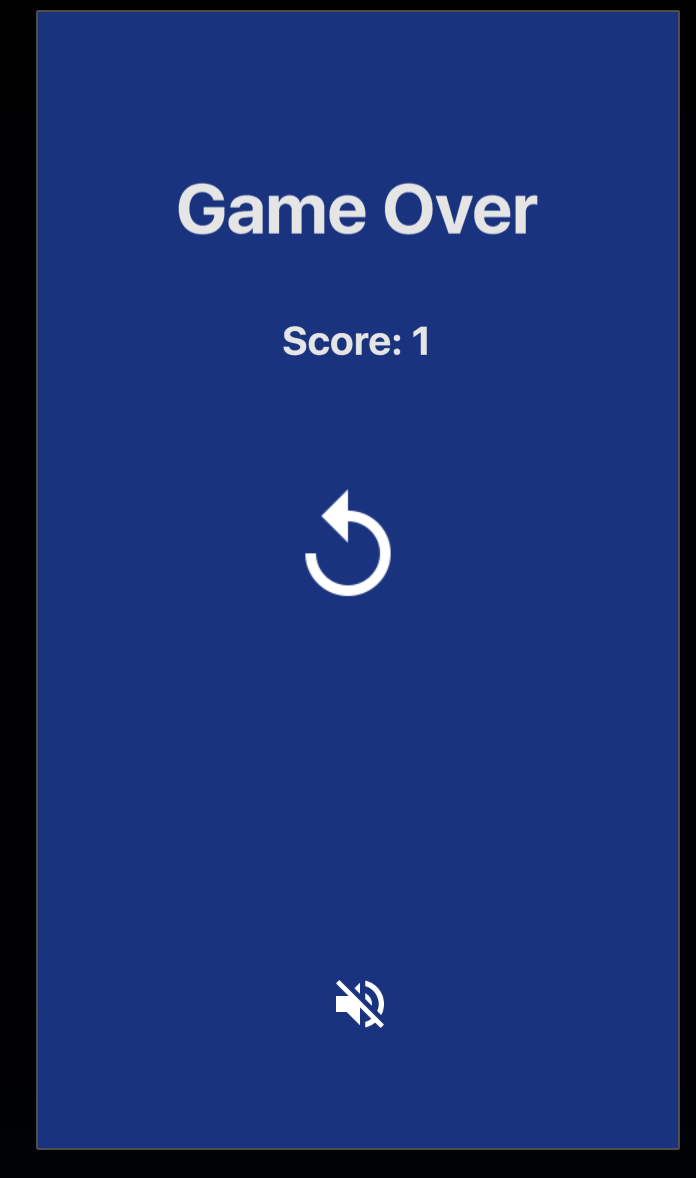


Figure Game over scene

# Implementation

The necessary shape sprites needed for the game were created manually. However, other images like icons and menus were freely available. In *shapes.lua,* we are creating various shapes randomly and assigning various functions such as *move* and *handle\_collision\_with\_other.*

The main logic of our game resides in *gamelogic.lua.* Our approach for the game is shown below:

1. Generate a random shape to be avoided and alert player to avoid this object.
2. Run a main game loop and
   1. Generate the shapes randomly.
   2. If the player hits the safe shape, increase the score.
   3. If the player hits the shape to be avoided, stop the game loop, clear the memory and move to end scene.
   4. If any safe shape are missed, stop the game loop, clear the memory and move to the end scene.

To handle the collision, we have implemented our own event listeners using the *Runtime:dispatchEvent.* In addition to it, each car has its own custom event listeners. That is, left car is controlled if tapped on the left screen and right car is controlled if tapped on right side of screen. This is also handled using *Runtime:dispatchEvent.*

# Team members and contributions

# Challenges

# References

[1] https://play.google.com/store/apps/details?id=com.ketchapp.twocars&hl=en

[2] https://docs.coronalabs.com/api/type/EventDispatcher/dispatchEvent.html

[3] https://docs.coronalabs.com/guide/physics/collisionDetection/index.html

[4] https://www.lua.org/pil/16.2.html