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

Introduction and Concepts

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 Mechanics

Scene Transition

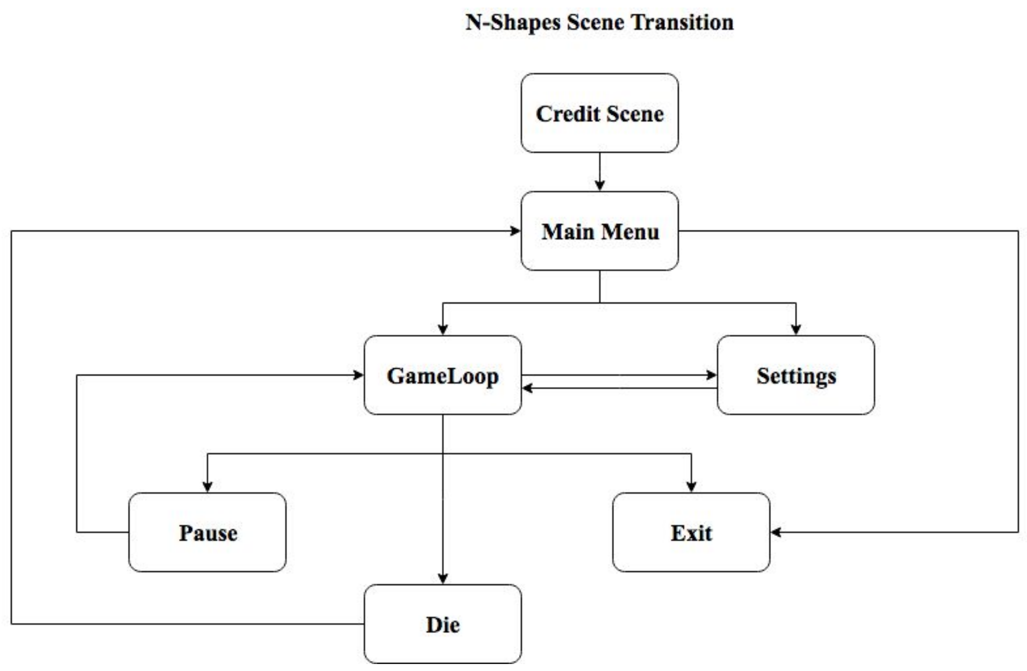

Figure 1 Game Scene Transition

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 is 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

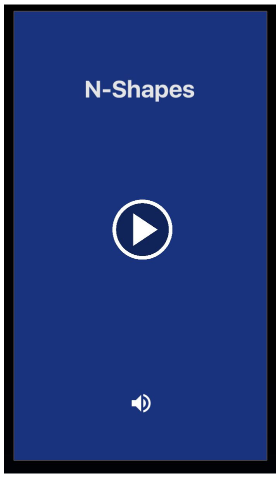

Figure 2 Main Menu

As shown in image above, if player clicks on play icon, the game is started. The figure shown below shows the screen just after game starts.

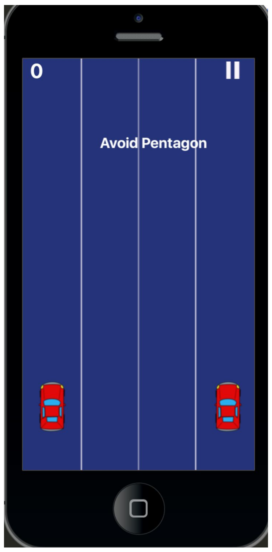

Figure 3 Shape to avoid

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 any other shapes, the score increases. However, if any of those shapes are missed, the game is over.

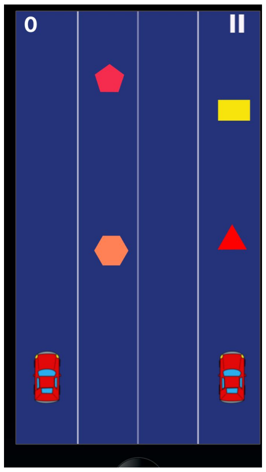

Figure 4 Gameplay

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:

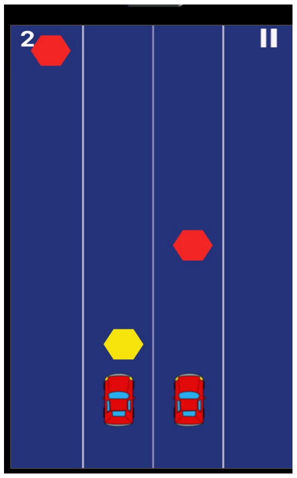

Figure 5 Scoring in game

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

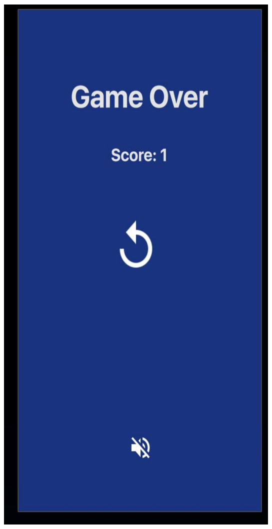

Figure 6 Game over scene

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* to the Shape class.

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
    a. Generate shapes randomly.
    b. If the player hits safe shape, increase the score.
    c. If the player hits shapes to be avoided, stop the game loop, clear the memory and move to end scene.
    d. If any safe shape is 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.* Moreover, updating the score is done through event dispatching.

Tools Used and Assets

- Corona SDK
- Game sprites like *sprites_new.png, car64.png, left.png, pause.png, play.png, replay.png and right.png*
- Game sounds like *hit.wav, car_sound.wav, pickup.wav, explosion.wav and gameover.wav*
- Sublime text
- Oneplus 3 for game testing on android.

Team members and contributions

To develop N-Shapes we have worked in-group of two. Each of us assigned tasks ourselves and fixed the bugs encountered. To summarize our work division, please refer to the list below:

Bidhya N. Sharma

1. Drew the car image and background image.
2. Discussed and wrote the game logic and collision.
3. Fixed the bugs encountered during *dispatchEvent.*
4. Fixed any other bugs that encountered when playing.

Sarvagya Pant

1. Drew shapes images necessary for the game.
2. Wrote the logic for rendering shapes and its class file.
3. Fixed a bug encountered during the *newOutline*.
4. Discussed and wrote the game logic and collision.

Challenges

During the implementation of the game, we faced various challenges. Some of the main challenges are illustrated in following points:

1. **Sprites:** Since we had no prior experience in creating sprites, making a new sprite by ourselves was a bit tough. We used *Photoshop* to generate the necessary sprites and background images.
2. **Event Dispatching:** Our idea of event dispatching in corona was believed to be like PUB-SUB messaging pattern i.e. any body could publish event and the event could trigger interested subscriber. To accomplish this, we first dispatched event from one image and listened to the event on other image. However, this didn't worked. Upon reading documentation, it was revealed that event dispatching and listening could work in the same image or on global *Runtime*. Thus, we were doing event dispatching in a wrong way. Later, we chose *Runtime:dispatchEvent* to make our task simpler.
3. **Shape Generation:** The other challenge during development of this game was shape generation using random function. Generating shapes randomly can cause the same shape to generate multiple time and thus make the game play hard. It was also a challenge to chose the to avoid shape randomly. Currently, the problem has been mitigated by using relatively slow speed of shapes and large gap between change of to avoid shape.

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