1. Introduction

This document reflects the report on Android code-camp 2010 activities. In this document we will first describe the idea and basic features were intended to be implemented, then we will sort them based on their priorities, finally we will describe the design and steps to implement the game.

1.1. RaceDroid Game

RaceDroid is an android version of classic cave-flying game. The player controls a helicopter flying in a cave trying to fly through it as fast as possible avoiding hitting the walls.

1.2. The basic idea

At the very early stages of implementation the idea was to implement this game in both single and two player modes. In former, the aim is to pass through the cave as fast as possible without hitting the walls of the cave and the score is given for the quicker lapse and less number of hitting the walls. In the latter, the aim is to reach the cave ending prior to opponent and off course less number of hits. As a basic idea, the screen was supposed to be divided into two parts and opponents could track each other while racing. But this design changed as suggested during the idea presentation time. Instead of dividing the screen, the players can see each other in one screen and may pass over each other (as depicted in Figure 1.).

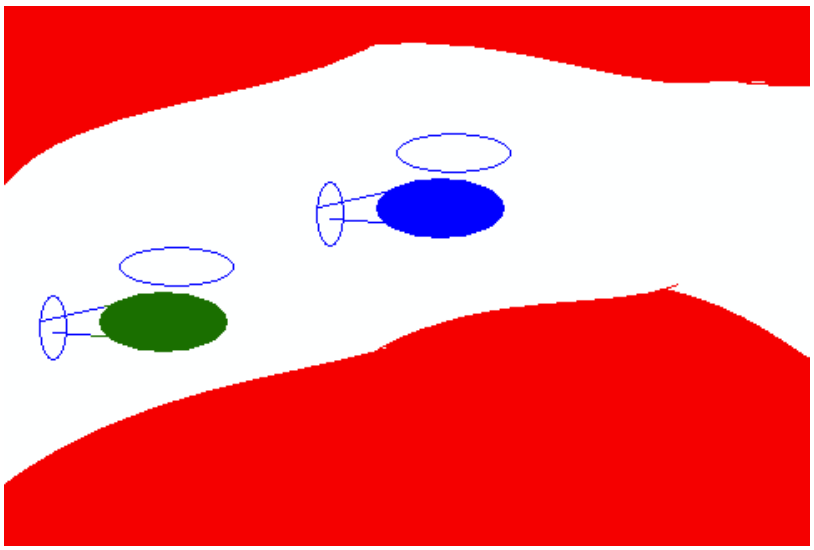

Figure 1. The design idea

Tommi Kallonen, Aku Luukka, Seyed Mahmoud Mortazavi, Alireza Kahaei
{firstname.lastname}@lut.fi

2. Design

Obviously, the application follows a typical android application and consists of views, activities and draw-able. In addition, classes were created to generate cave randomly and also manage connections between two devices.

2.1. Views & Activities

RaceDroid has four different views and activities responsible for controlling them.

- The main view : for selecting single or multiplayer game (main menu)
- The Server view: view for setting up server mode, and setting port to listen to
- The Client view: view for client connection establishment. Setting IP and port for TCP connection establishment to other device (server).
- The game view: where the game is actually played. Players come here from main view (single player) or from server or client view (multiplayer). This view has a thread that is responsible for drawing the scene based on the time and refreshing the canvas in the intervals. The thread has the following pseudo code:

```
while (! Game_finished){
        update_physics();
        redraw_the_canvas();
        if (multiplayer){
                send_coordinates_to_peer();
                receive_coordinates_form_peer();
                draw_opponent_if_in_screen();
        }
}
```

- update_physics: calculate the new position of the helicopter object based on its previous position, the elapsed time and forces which are exerted to it. These forces are gravity (downward), pulling force (upward) which is increased while the "up key" is pressed. This way it simulated the flight of the helicopter and it will fall down the ground if no pulling force is exerted by user. And finally, acceleration (forward) to make the movement of the object faster.
- redraw_the_canvas: draws all objects on the screen, cave and helicopter.
- send_coordinates_to_peer: in multiplayer mode the helicopter coordinates is sent to peer.

- receive_coordinates_from_peer: in multiplayer mode the opponent coordinates is received from peer.
- draw_opponent_if_in_screen: draws the opponent if the coordinates are fit in the current screen.

2.2. Creating the cave map

For generating the cave, two lists of random nodes are generated in the beginning of the game. It is passed to the other player once the two player mode is selected. To make to game more pleasant, when certain length is given, it is divided into even and odd sections. The nodes are only randomly generated within the range of even sections. This way nodes have some kind of distance from each other and this makes the random map more sensible and logical. In addition, in order to make the map even more logical, each node of the floor of the cave has to have a minimum distance from the ceiling of the cave. Therefore, a specific minimum distance is given to avoid having any impossible cases. These nodes are saved into ArrayLists once generated and passed to the other side once the two player mode is selected.

2.3. Visualizing the map on the screen

As explained before, map is a set of points connected to each other to form the cave. Hence there are lines that show the ceiling and floor of the cave. This is one approach to visualize the cave map. 2 shows the result of this visualization method in which lines and background image is used:

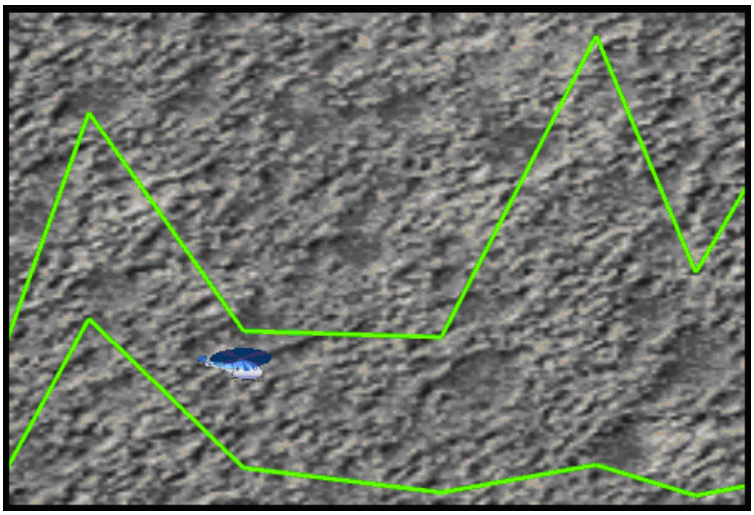

Figure 2- Visualizing the cave as only lines

But to give a better feeling to user about the cave, the areas above the upper lines and bellow the lower lines should have been filled out so that it seemed more solid. For achieving this, the "Path" object used a path is a closed area which can be defined by defining the start point and end point and several lines between. The result of applying this object is shown in Figure 3.

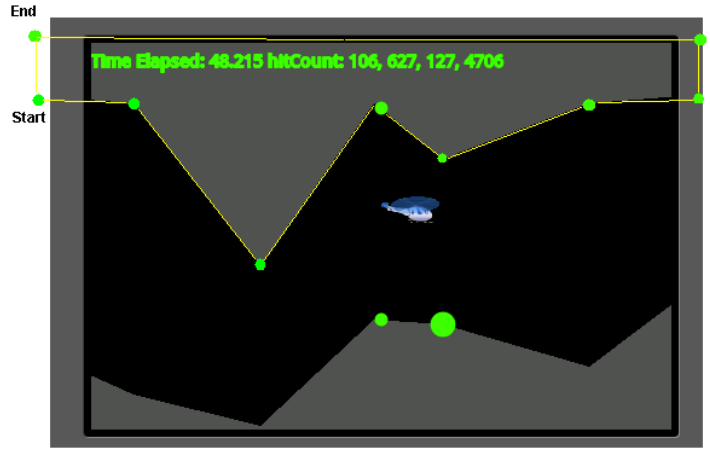

Figure 3- Applying Path Object to fill out the area (path from start to end point)

2.4. Hit counter (Hitting the cave ceiling and floor calculations)

To calculate the hitcount, the height of both floor and ceiling is measured all the time based on current position of the helicopter and the location of the next and the previous nodes for both floor and the ceiling. If the helicopter is within the range measured, the hitcount remains the same. If the helicopter's height is equal to the floor or the ceiling however, the hitcount increases. It is worth mentioning that the helicopter can not go below or above the floor and the ceiling.

2.5. Starting the game in two-player mode

Before the game starts, a random map is created. In multiplayer game this map is sent from server to client and client sends ready message to server when it completely processed the received map and stored it locally. The game starts after this. Figure 4 shows how client and server synchronize the game start.

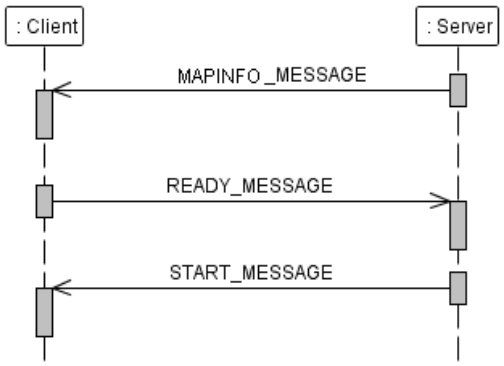

Figure 4- Starting game in two-player mode scenario

2.6. Network Connection (Connection engine)

Before the game starts the ConnectionEngine offers the functions for setting up a server and connecting to a existing server. After a connection is created, a thread for listening the connection is started. This thread waits for incoming messages and after a messages is received, it is sent to the game using a callback function (the game must implement function defined in ConnectionCallback).

3. Implementation status

In the current version of RaceDroid the player can play alone or with a pair. In single player game the time for start to finish can be registered and the hits to the walls are counted but they are not used to calculate any result.

In multiplayer game the set-up process has some problems, so the game might not start at the same time on both devices. Also in the current version the transferring of map is broken for some reason, so the players don't get the same map. The winner of the game is not detected in any way – both palyers just fly to the finish line and the game ends.