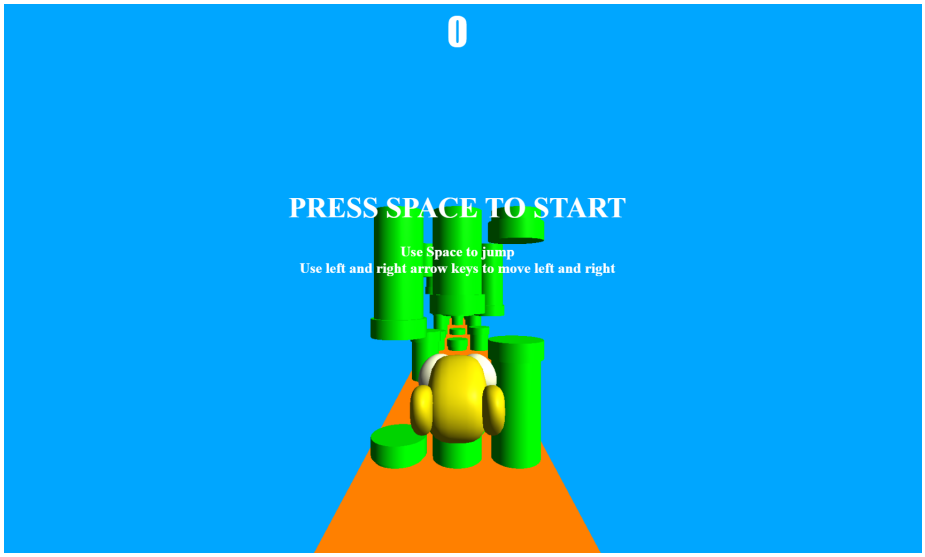
Flappy Bird

Lars Eivind R Lien 1977543

September 2021



Contents

1 Introduction

This document represents a technical report and a user manual for the game Flappy Bird 3D. It is made in context with the course Interactive Graphics 20-21, at La Sapienza, Rome. Flappy Bird 3D is inspired by the well known mobile 2D game of the same name. You control a bird flying through obstacles, and you get points for the number of obstacles you pass successfully. Flappy Bird 3D is available online, and can be played by clicking the following link: Flappy Bird 3D

2 Libraries and Tools

In the game development, I have used several different technologies / libraries. For the rendering of the 3D world, I have used Three.js. For simulating gravity and calculating collision between obstacles, I used Cannon.js. The animations are implemented using Tween.js. GLTFLoader (which is a part of three.js) is used to load the Flappy Bird 3D model. Firebase Database is used to keep a leaderboard.

3 Environment

The development in this project is based on Three.js, which is a cross-browser JavaScript library used to create and display animated 3D computer graphics in a web browser using webGL.

4 Physics

A physics engine is important in game development if you want to ensure a realistic simulation of real world physics. Collision detection and free falling are two central physical phenomena used in this game. There are several physics engines available in JavaScript which are easily implemented with Three.js, such as Cannon.js, Ammo.js and Physijs. Flappy Bird 3D uses Cannon.js, because it is considered to be more compact and comprehensible than the others. The physics engine makes it possible to detect collisions between the bird, the obstacles and the ground. It also simulates gravity, which makes the flying seem more realistic. To implement the physics engine in to the game, you need to create Cannon.js bodies similar to the Three.js bodies you create. They need the exact same dimensions, rotations and position in order for them to work as intended. It is also possible to give them a weight. If the weight is above zero, the object will be affected by gravity, if it is equal to zero, the object will stand still. The gravity of course only works on the Cannon.js object. For the three.js objects to act accordingly, the positions and rotations of the three.js objects need to be copied from the corresponding cannon.js in the animation loop.

5 User Interaction

For the user interaction, the top focus was to make the game easy to use. When you open the game, you will get directions on how the control works. For web browser, you use "space" to jump (or fly) and left / right arrow keys to move horizontally. For mobile devices, you tap the screen to jump (or fly) and swipe left / right to move horizontally. It is not possible to choose the speed of the bird. This is in order to make the game incrementally harder. For every two obstacles the bird passes, the speed increase by a fraction. When the game is over, you can view the leaderboard, and you also get the possibility to register your name to the leaderboard if you get a new highscore. By pressing "enter" the page refreshes, and you can restart the game.

6 Rendering

For performance reasons, the rendering of 3D objects happens constantly as the bird moves forward. There is always a set amount of obstacles showing in the 3D world. Whenever the bird passes a row of obstacles, that row is removed both from the Three.js scene and the Cannon.js world. Simultaneously, a future obstacle row is added. This happens so that the number of 3D models inside the scene are kept low, leading to good performance.

7 Lights

In the game there is two types of lighting, ambient and directional. The ambient light is used to illuminate all objects in the scene equally. The directional light gets emitted in a specific direction. It behaves as though it is infinitely far away, which makes it a good simulator for sun light.

8 Textures

It is used three different textures for this game. Flappy Bird has its own texture that is implicitly loaded with the gltf model. The walls, ground and roof are created with MeshBasicMaterial, which is a material for drawing geometries in a simple shaded way, and is not affect by lights. The obstacles are made with MeshPhongMaterial, which is a material for shiny surfaces with specular highlights. I found this to work best with the obstacles.

9 Hierarchical Model

In this game, Flappy Bird is the only hierarchical model. Flappy Bird consist of a body, mouth, eyes, left wing and right wing.

10 Animation

The Flappy Bird animations exploit the structure of the hierarchical model. When the bird flies upwards, the body is rotated slightly upwards as well, and when the bird flies downwards, the body is slightly rotated downwards. When the bird flies to the left or right, the bird rotates slightly to that direction, and rotates back to normal position once the movement has finished. The wings are constantly moving in a flapping manner in order to make the flying seem realistic. The animations are created using the Tween.js library. This manages the shift from one property value to another property value for an object.

11 3D Models

The Flappy Bird 3d model is fetched from Sketchfab(https://sketchfab.com/). There were several free flappy bird models to choose from, but not all of them had the features I was looking for. Among other things, it was important to have the possibility to move the wings seperately from the body. The obstacles are created manually, by stacking one cylinder on top of another, and placing these relative to each other.