Developing of a Jump and Fly Game

Results of a practical course at the Chair for Computer Graphics and Multimedia (RWTH Aachen University, Germany)

Lukas Saretzki [*]   Christian Schmidt [†]   Frederik Engels [‡]

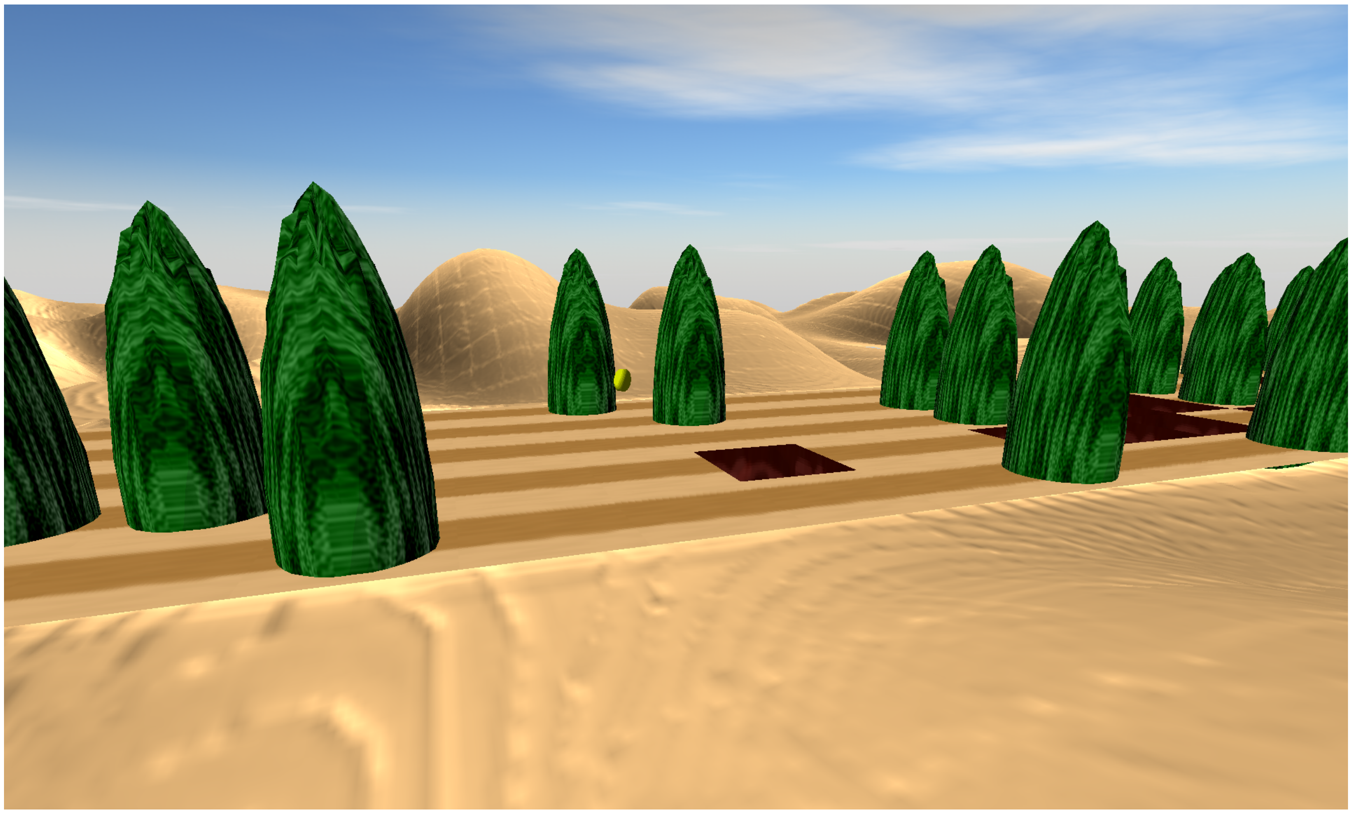

Figure 1: *Level from the side*

Abstract

LostFlying is a Jump'n'Fly game in which the player controls a robot in levels consisting of five parallel lanes. You have to reach the end of the map by collecting coins, avoiding cacti and jumping over holes. Further more the levels can be edited via a png-file to create your own levels and to customize the difficulty. The above figure shows a view over a level (cf. Figure 1).

Keywords: game programming, jump and fly game

1 General Information

LostFlying is a game of the genre of 'Jump and Fly'-games. Those games feature a protagonist which moves in a linear world and has to avoid obstacles and collect some bonus objects. Protagonist of LostFlying is a high-tech robot which was teleported to a unknown world and he seeks his way back home. So the main goal of LostFlying is to reach the end of the level, covered with holes and cacti you have to avoid.

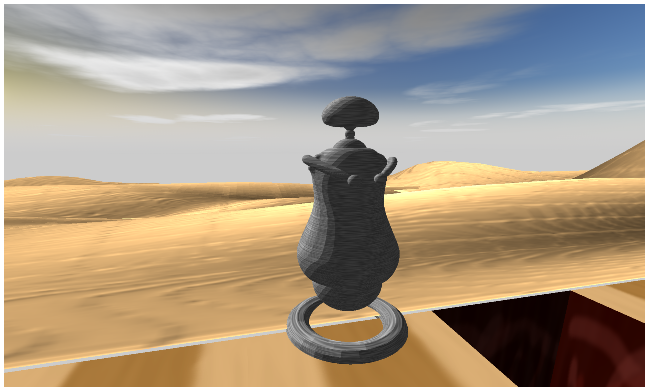

Figure 2: *Close up on the robot.*

2 Features

We will now have a look at the features LostFlying has and see how they are achieved.

[*]lukas.saretzki@rwth-aachen.de
[†]christian.schmidt1@rwth-aachen.de
[‡]frederik.engels@rwth-aachen.de

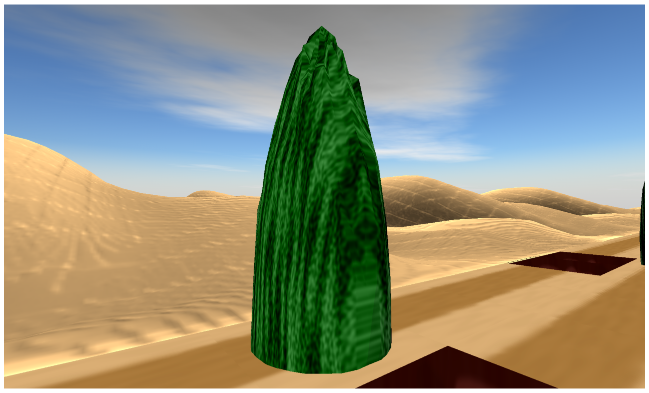

Figure 3: *Close up on a cactus.*

2.1 Level loading

The level which is played when starting the game is not randomly generated or written into the source code. It is stored in a color map as a png-file. The file is then read from the program at start time and translated into the playable level. The level is made up by small tiles and each pixel in the color map translates directly into one tile in the level.

2.2 Noise based textures

While the ground texture is loaded from file, the robot and cacti have textures which are generated from a random noise. The noise generation is based of a work by Ian McEwan [McEwan 2011]. Input for the noise function are the coordinates of the model relative to its own point of origin. The noise then is transformed to fit the desired look and finally translated to a color used to paint the model. This is used multiple times in one generation process in order to achieve a more sophisticated texture.

2.3 Terrain

The terrain surrounding the level are loaded from a gray-scale height map and are rendered with a small bumpmapping effect. This gives them quite a natural look. And it is quite easy to generate bigger terrains to avoid high frequent repeating the the same terrain design on longer levels.

2.4 Light

The lighting is a basic per-pixel-lighting but as you can see on the above picture of the robot the shadows and colors are cropped to give the objects a cartoon-like appearance (cf. Figure 2).

2.5 Collision detection

The collision detection is a self developed and very simple axis-aligned bounding-boxes algorithm. Each object and also each floor tile has a box shaped bounding volume and in every frame it is tested, using these boxes, whether the robot collides with some of the objects in his surroundings. If a collision is detected the program responds according to some given rules such as counting a the coin or stopping the game.

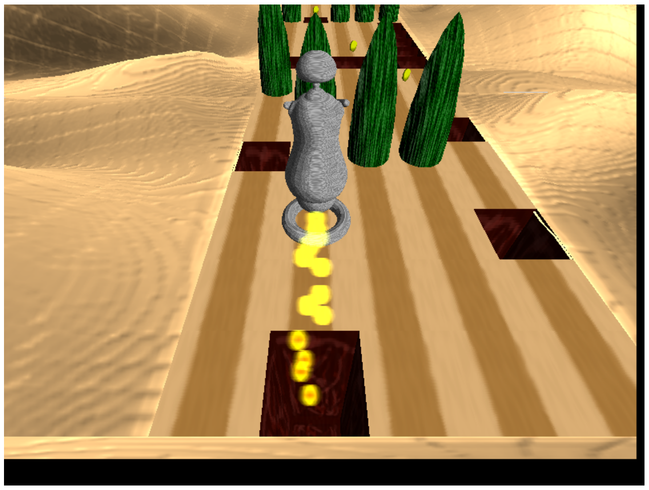

Figure 4: *A in action shot of the particle system.*

2.6 Particle system

When the robot jumps he seems to be propelled upwards by a rocket thrust. This effect is created by using a particle system which spawns a high number of semi-transparent 2D particles.
These particles automatically align their front to the camera to maximize the appeal of the effect. The transparency and number of particles then create the illusion of a flame coming out of the robot. The figure **??** shows the particle system in action.

3 Work and Team

After looking at the key features of the game, we will now have a brief discussion of the team which created the program and how the work progressed.

The biggest problem with the team was, that the team composition was subject to major changes. We started as four programmers but very soon one of them left the team.
About a month after the first meeting we had a new member, Frederik Engels, assigned to our group. Then after the first milestone was reached the second member left the group.

But communication was also a problem within the group. We used the “redmine“ tool in order to coordinate the work but often it was just ignored and mail contact was very slow way to talk. And so the workload was not evenly distributed between the group members.

References

McEwan, I., 2011. 2d simplex noise function. https://github.com/ashima/webgl-noise.