Space Rave Documentation

Interactive Visual Computing WiSe 14/15

Arne Beer, MN 6489196
Rafael Epplee, MN 6269560
Sven-Hendrik Haase, MN 6341873

January 21, 2015

1 Introduction

Figure 1: Title scene

For our IVC project "Space Rave", we set out to explore the emotional impact of a music video that is synchronized to the contemporary electronic dance music song "Strobe" by Deadmau5. We chose this song due to its regular profile and agreeable electronic character. The video consists of three distinct scenes while the visuals are set to up to represent an abstract spacescape throughout the scenes. The visuals consist of procedurally generated stars which pulse rythmically and swiftly with every kick, a procedural, fluctuating background nebula and a tastefully hand-modelled spaceship.

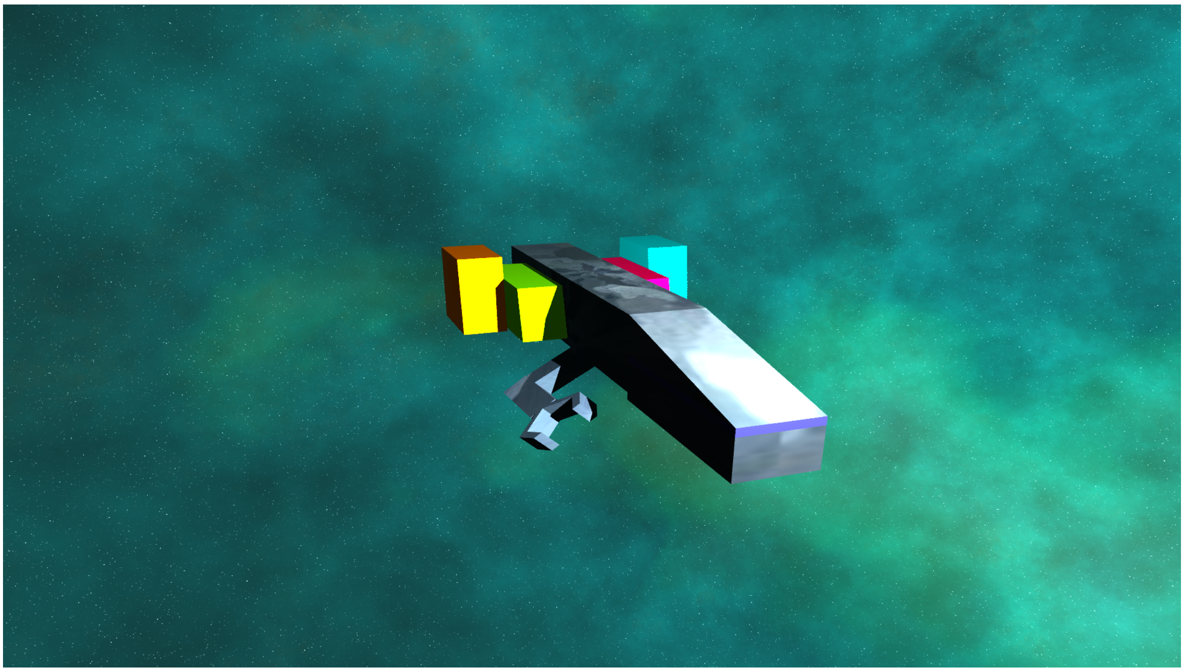

Figure 2: Breakdown scene

From a technical standpoint, we used POV-Ray to generated the images and Audacity to analyze the music to find beats and chords. The timings were particuarly hard to get right.

It should be noted that while we do not own the rights to the song we used, we believe that we are using this song in good faith in our transformative work under Fair Use.

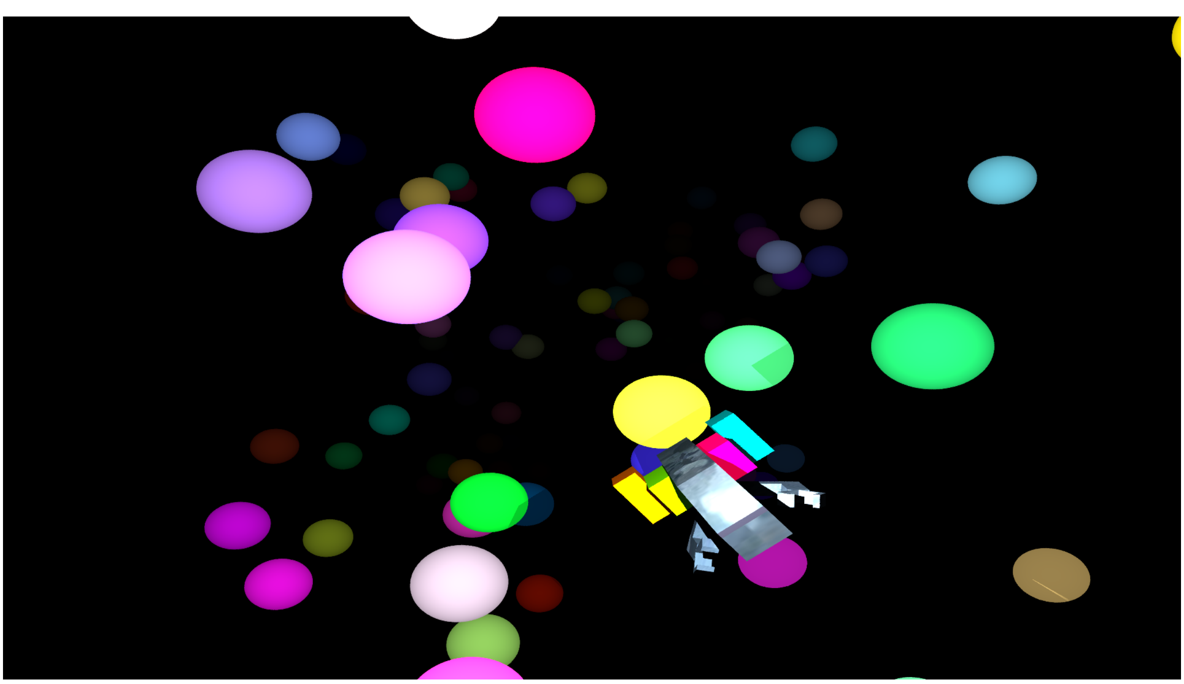

Figure 3: Second part scene

2 Extracting rythm information from the sound file

Since POV-Ray does come with any kind of Fourier transformation, we had to analyze the song by hand with Audacity (see 4). While this worked fairly well, we had to be very careful about the sub-second timings which are easy to offset by a few milliseconds which ruins the whole experience. For beat extraction, we used Audacity's helpful beat finder tool (see 5) which resulted in Audacity giving us annotations where the beats were located.

In order to find the chords, we used the spectrogram view (see 6) which allowed us to visually find the correct positions.

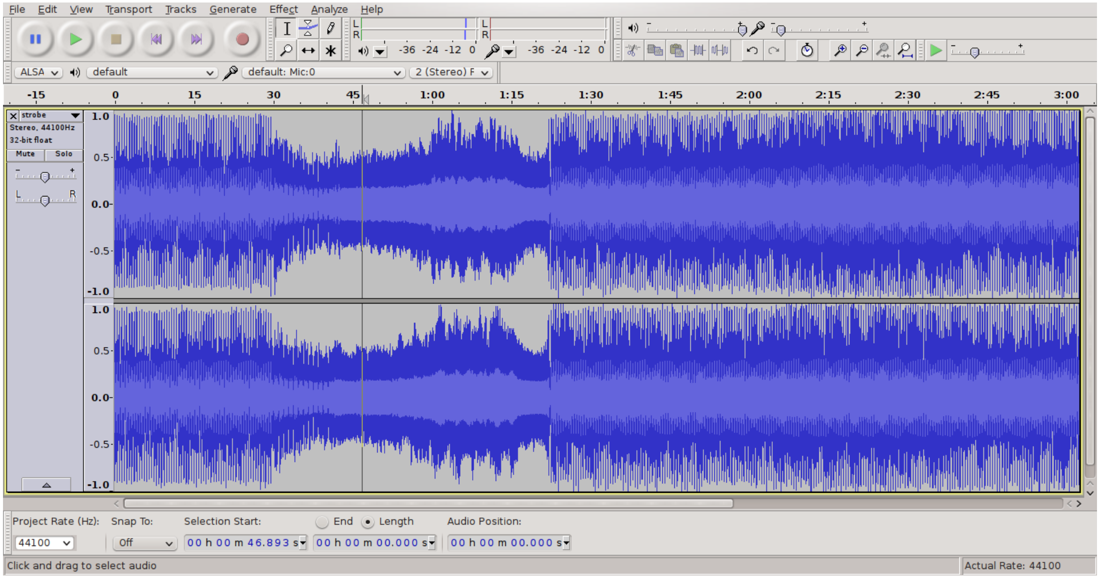

Figure 4: Audacity main view of the track

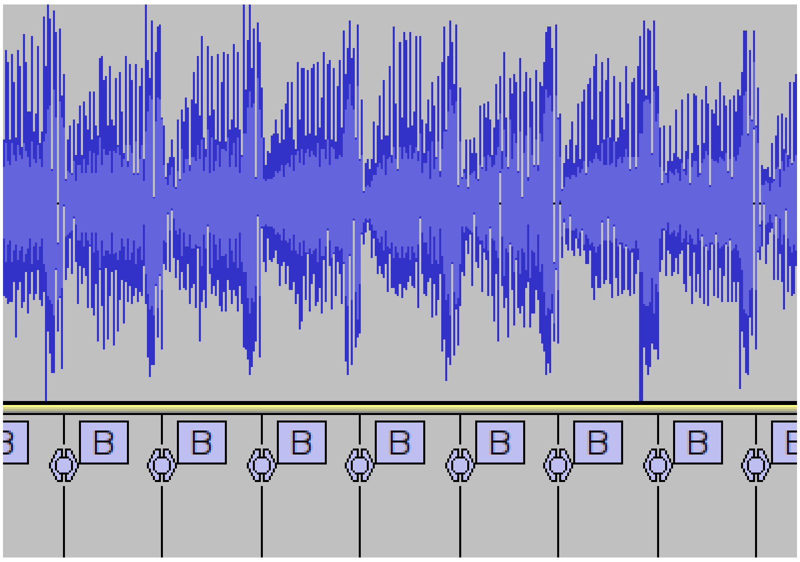

Figure 5: Beat analysis using Audacity's beat finder

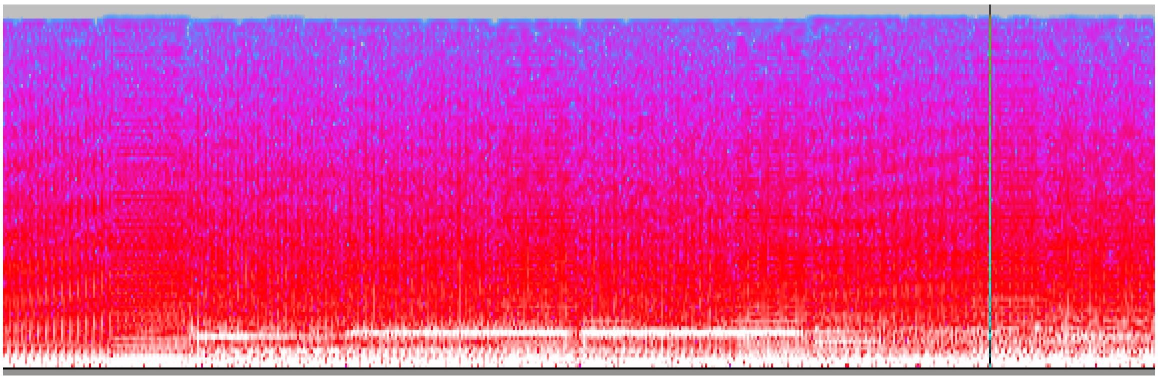

Figure 6: Spectrum view

3 Scenes

3.1 Intro

The intro is composed by creating a starfield and moving the titles towards the camera. The starfield is generated by adding spheres with a random position and scaling their size depending on the beat.

The title is a simple union of text objects using timrom.ttf with a length of 5. To make it look more interesting we applied a Polished_Chrome texture with some bumps and a little reflection for a shiny look. The camera is constantly moving forward, while the titles are slowly moving towards the camera. All stars got a static position and aren't moving at all.

3.2 Breakdown

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

3.3 Part Two

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

4 Creating the Starfield

5 Modeling the Spaceship

We didn't use any modeling tool for making the spaceship. Instead, we opted for a blocky, "minecrafty" style with almost no details. We used unions to combine mostly boxes and prisms into quite a basic shape. Even the most complex part, the claws, are built with boxes and prisms.

The size of the ship itself and various parts is easily customizable through local values declared at the top of `ship.inc`.

Most parts of the ship are actually mirror-symmetric, like the base body, the claws and the window in the front. To minimize effort, we just modeled those parts in half and then mirrored them using duplication and scaling.

The only non-symmetric parts of the ship are the engines at the rear end. They are each modeled on their own, each having a distinctive color.

6 Timing animations according to the song

In order to get accurate timings, we used milliseconds with POV-Ray. Since POV-Ray only directly supports seconds, we multiplied all numbers with 1000 to get better resolutions. The first step towards that was figuring out the exact length of the song which is $3m32.067s$ which in turn is $212.067s$.

7 Background Generation

To create a starfield and nebula we used a comination of 2 spheres and a sky_sphere.
The sky_sphere got a pigment that gets a perlin noise pigment map, which consists of three different colormaps. As a result we generate the procedural starfield.
In order to generate a nebula we create spheres and define the interior of those. There are two different spheres with different interior medias.
The first nebula is generated by creating a hollow, transparent sphere and filling it with an interior, which is defined by a media that got 2 different densities. A density is used to vary the density of particles inside a media. By adding two densities we get the intersection of both densities. The first density is defined by a rampwave on a colormap, to get a nice transition for the colors. We decided to spice this up by translating, warping everything and translating it back. As a result we get a nice nebula effect.
The second density simply defined by a rampwave on a colormap and different density values inside the color map.

The other sphere is quite similar to the first one, except that it got other colors, a single density and we actually vary the turbulence of density to the beat to create a wobbly effect on the nebula. We use some other values like octaves, frequency and labmda to create diffent nebula look.

8 Ship movement

The movement of the last ship in the last scene has been achieved by using a spline. We use a natural_spline to create a smoother trajectory. The spline is applied by using Trans_Spline with pretty high values for foresight and banking, because the ship actually moves at -60*z per second.

9 Summary