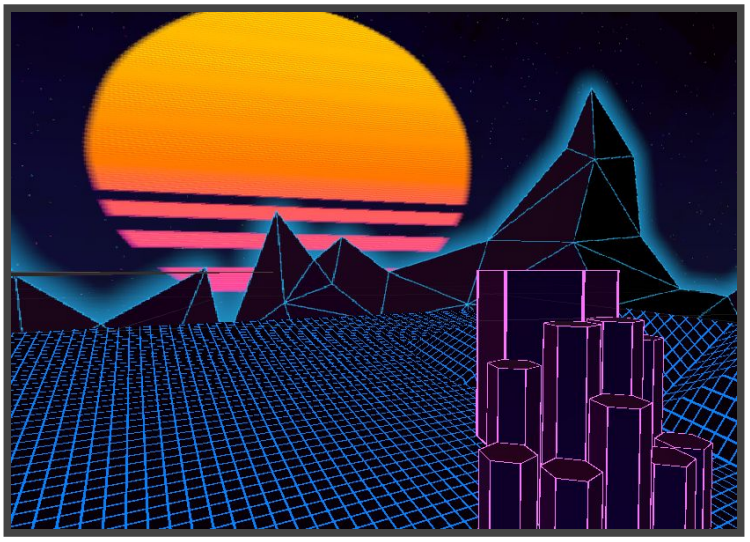
MIDTERM REPORT



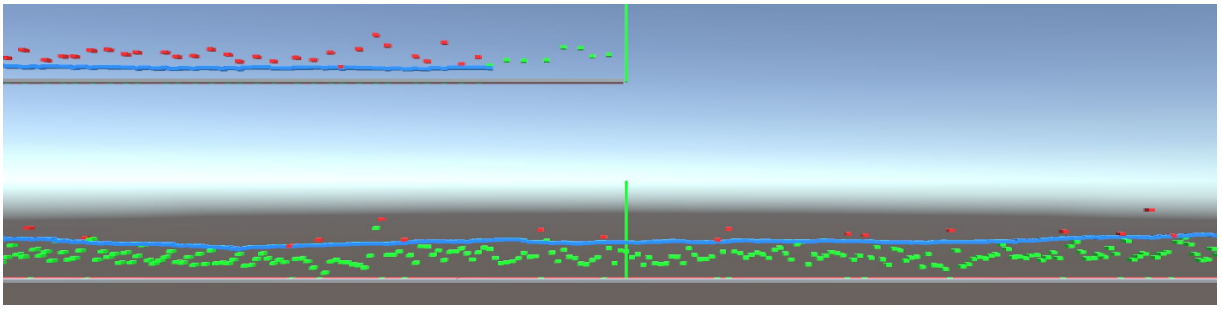
PROJECT TITLE: Procedurally generated rhythm game

GROUP MEMBERS: Mads Sejer Pedersen, Frederik Lyngsøe, Mikkel Elmelund Esbersen, Sebastian Nørager & Jonatan Heine Langer

PROBLEM FORMULATION/RESEARCH QUESTION:

Our tentative problem formulation reads as follows: “How to implement sound analysis as a seed for procedurally generating a map for a rhythm game?”

The goal for this project is to successfully build a sound analyzer that can extract the musical beats from songs. There are many ways to build such a sound analyzer e.g. using FFT. The idea is then to use the music as a seed input to a map generator that will generate a map specific to the beats of the song for the player. The map generation is done procedurally and not manually. The player will then be able to play through a map where the goal is to jump/land onto towers to the rhythm. For smoother jumping we will be implementing a “BHOP” mechanism, that lets you jump without slowing down, but instead increasing your speed. It is important to add that FFT is one of the main solutions we are using at the moment, but there are more possibilities out there.



Research topics:

Fast fourier transform, signal processing, peak detection, beat detection, perlin noise, noise, seeding, sound waves (frequency & amplitudes)

Fast fourier transform

In this project our use of the FFT is to distinguish between different frequencies of sound waves from our input audio file so we can more easily narrow down our search to for example a low frequency tone like a bass, or a kick drum.

This FFT will be the basis of our sound analysis so we can identify the beats from a given music sample since the idea of our game is to do certain things to the beat of music.

Outline:

Introduction

This section will clearly define our problem and give a brief introduction to the different research topics

Theory

Will describe the tools and algorithms used to solve our problem.

The Game Design

- **Software structure**
    - This section will cover the overall design of our game specifically the mechanics and intended gameplay.

- **Implementation**
    - This section will cover how the specific algorithms are used in our game to achieve the intended mechanics.
- **Testing**
    - This section will cover our methods of testing and what we learned by doing so.

Discussion

- What are the uses for this particular algorithm/software that we have produced?

Conclusion

- Did we satisfy our problem formulation?
- Evaluating the discussion.

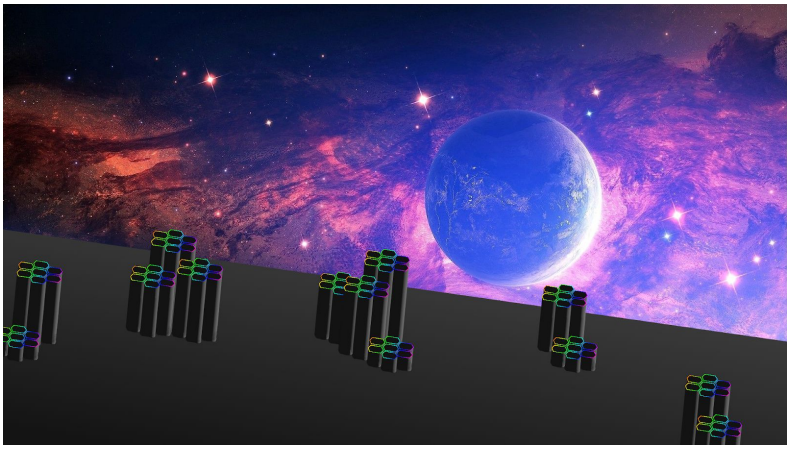
Game-Design

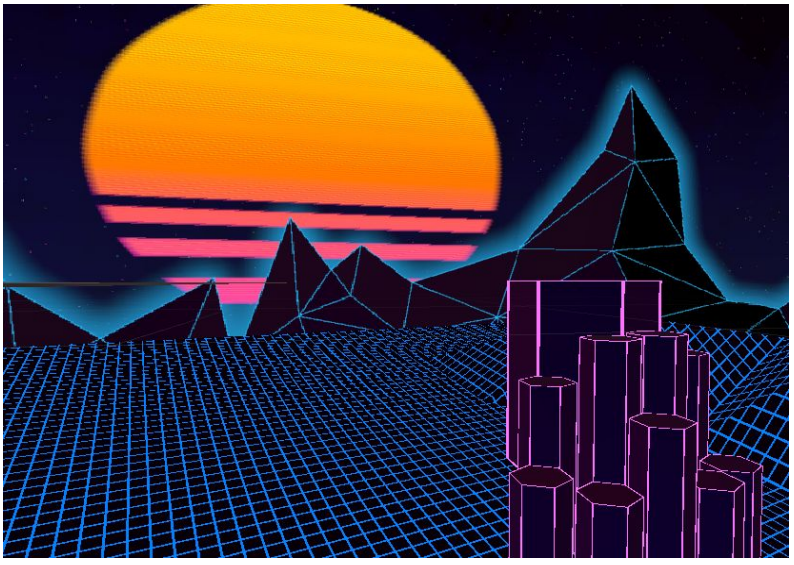
We use Unity as our game engine, it handles rendering collisions, and basic object-oriented issues for us. With Unity you write smaller “scripts” and give them to objects in the scene which then implements the script.

- **Game-engine (unity), 3d - Modelling (blender/unity) - image manipulation (gimp) & scripting (c#)**
    - player movement script (bhopping)
    - FFT script (beat analysis)
    - Making 3d -models
    - Skybox texturing

- **Game-play**
    - Difficulty
    - Is the game fun?

From Concept art to Product





“Besides all the algorithmic and software development work, we will also be working with 3d modelling, shader programming, and other visual elements of game making. Here’s a preview of some current early visual work.”