AKDENIZ UNIVERSITY
ENGINEERING FACULTY
DEPARTMENT OF COMPUTER ENGINEERING

SENIOR PROJECT FINAL REPORT

Students’ Name Surname :	Mert KARABABA/Kaan KANATLI /Enes Salim ERDEM
Students’ ID Number :	20160807017/20160808062/20160808033/
Supervisor Name(s) :	MUSTAFA BERKAY YILMAZ
Date:	12 /01 / 2020

Contents

List of Group Members and Their Roles in the Project

To complete this project, we first need character physics, then we need a map design that this character can play interactively, and finally, we need music-integrated algorithms to play this game in a rhythm-dependent manner, which is the main goal of the project.

So we divided that task to 3 main categories as we can see in below;

First category-) Mert Karababa

All physics of the game.

Second Category-) Kaan KANATLI

Music and rhythm adjustment algorithms.

Third Category-) Enes Salim ERDEM

Design of the game; Game interface, graphic design, map design, character design.

Let's see what those categories include in detail.

First Category -) All Physics of the Game

The primary purpose of these physics in the game is to make the character make movements that are related to the reality, for example; we want to jump, and when we press the jump button, the character takes off and lands correctly.To do this, we need to define the following physics rules.

1- **Character main physics**: basic movement physics like ,going left,right,up and down sides.

These physics will be active throughout the game regardless of the music.

2- **Character play physics**: Jump, hit, slash etc. physics.

These physics will be active when **Music-Rhythm Algorithm** kicks in.

3- **Object physics**: The speed of arrival of objects and their interaction with the character while playing the game

Second Category-) Music and Rhythm Adjustment Algorithms

Keeping certain rhythms of the music with the help of algorithms, activating the objects at these points and activating the **character play physics** accordingly.

1- **Music adjustment**: Music adjustments of the game like music start point, change point and end points.
2- **Music-Rhythm Algorithm:** The progress part according to the rhythm, which is the main purpose of the game, will be activated.This part will primarily divide the music playing into its rhythms. Selections from these rhythms will be assigned to the parts where **Character play physics** will be activated.
3- **Music-Object Adjustment:** Gather informations from **Music-Rhythm Algorithm** and according to that informations it selects object sender times.With the help of that adjustment now we can have rhythm based sended objects while playing.

Third Category-) Design of the Game

The part where the game is visualized. Such as Character design, map design, objects design, user interface. Basicly all visual things in the game.

1- **Game interface:**Game login screen, with menu includes start the game, logout, options etc.
2- **Graphic Design:** Character , object and map design and drawings.
3- **Map Design:** The part where the designs of the area where the game is playing.such as, map difficulties, map backgrounds, environment of map, colour and the shape of the objects.

Definition of the Problem and Difficulties

Most of the difficulties we are going to experience and problems that will occur will be in the rhythm part of the game. Since the base game going to have it’s own tracks for original designed maps, it will be easier to balance the game together with the rhythm it is going to have for the specific map.

Some difficulties may occur during the part where we set the perfect timing for the player to act based on rhythm of the track. Other than synchronizing the game with the current track being played, the most difficult part will be applying the player’s chosen track to our endless run since there must be lots of handling with the track like key signature and time signature changes in the middle of the song or some songs having intros with unauditable time signatures etc.

a) Player’s Action Timings

While this is not being the actual problem of the project, it is something that must be worked on since it is what makes the game fun. We know that no human can time any button action perfectly since that includes both human reflexes and input lag delay. So, delayed actions must be accepted at a certain point while still making the player feel they are being part of the rhythm that is going on.

We can get around it by leaving player enough time to act or giving them some kind of indicator to follow the rhythm of the track. The most appropriate way will be found by trying some different action times and indicators or maybe different approaches to make the player involved to the rhythm as much as possible.

b) Player Chosen Tracks

This is the part where we are going to have lots of difficulties from playing the player chosen track in game to synchronizing the game with any kind of genre. There are so many things that we need to be careful about implementing of this system. Even it seems like players do not have any limitations about this part, there must be some limitations of player chosen track to make it still playable.

First of all, the algorithm we are going to use must check the whole track since there can be more than one time signature in a song. If the player wants to use a song that starts with 120 bpm but later on the song continues with 150 bpm, action time must change in game at that part of the song as well. Even if we can implement this as it is, it may occur some other problems like player not noticing the timing difference or the game might get too fast to keep the pace with or just too slow.

Other than that, there are songs that has long intros with unauditable rhythms we should take care of to not lose the pace of the game. Not for only intros but this includes for some music genres as well. Some limitation for the bpm (beats per minute) might be needed since there are lots of fast or too slow songs. There are couple ways to work around with it by including the 8th notes to the player actions to make it playable even the song is too slow (doubling the bpm) or doing the opposite which is using half notes to keep up with the pace of the song if it is too fast.

c) Character Interaction

The game physics are another part of the project where we will experience some difficulties. Game object and character boxes must be well taken care of otherwise, there will be lots of issues about interactions between character and game objects or while a game object colliding with other game object etc.

Comprehensive Survey

The main goal of the design of the game yields the player to explore the world in a fun and efficient way and in an effective mechanic-driven form.

As a consequence of this principle, the priority of the level design is a crucial part of the game.

This paper expounds, how to acquire this principle by analyzing the existing games of the genre.

Examples such as Sonic the Hedgehog, Super Mario, Super Meat Boy, Celeste, Dead Cells… on the 2D platformer side. On the other hand, in the Rhythm-based gameplay-side: BPM (Bullets Per Minute), Crypt of The Necrodancer, Rhythm Doctor, etc.

That much of gameplay elements led us to focus on level design physics constraints, object structures.

There are many books that we benefit from, concentrated on basis of the level design: *Rules of Play* by Eric Zimmerman and Katie Salen, *Beginning Game Level Design* by Marc Scattergood and John Fiel, *Level Design for Games: Creating Compelling Game* by Phil Co, etc.

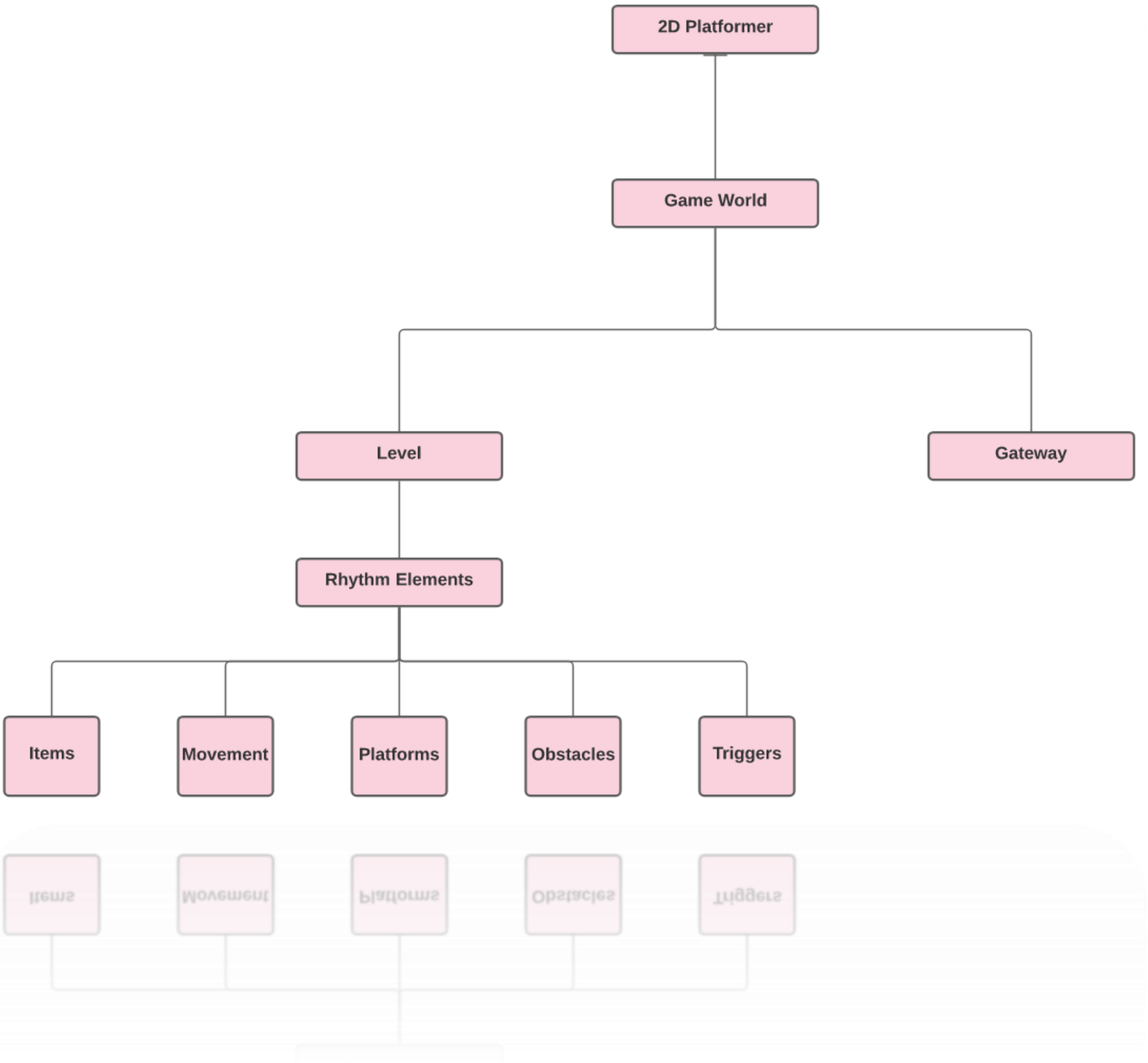


Fig. Conceptual Model of the Game Components.

Level Components and Rhythm Elements

There are certain components of the game categorized by their functions as shown in the figure.

a) ***Items*:** In the game as in the many existing examples of the genre, we'll have items that provide diverse benefits when player acquired. Not just particularly weapons but bonuses like gold, health, powerups or of course weapons which have various fun features according to rhythm system.

b) ***Movement:*** Besides the game character's core movement system, there will be alternative features that bring diversity in character movement-vise. Such as, ladders as in Super Mario or Donkey Kong, ropes, bouncy surfaces which gives moderate jump boosts, etc.

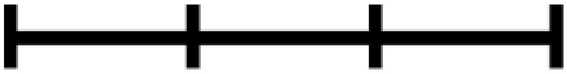
c) ***Platform:*** Maybe the main element of the game, Platform contains any ground surface that the game character can move on it and it moves interactively with other objects.

d) ***Obstacles:*** Those are the objects that can affect the player's movement in a good or bad way (most likely the second one). The player should avoid them by dodging, destroying, or jumping over them. Of course according to the rhythm system and also some obstacles are capable of move and they move in a most rhythmic way.

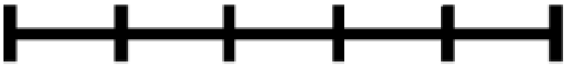
e) ***Triggers:*** Some of the objects that were previously mentioned in the Movement section are an example of the Triggers. For instance, Buttons, surfaces that give jump boost, doors that'd be opened by keys, pressure plates, etc. Those triggers bring more interactivity to the game and it changes the level's current state actively. Those triggers also can be functional elements for puzzle-focused levels.

Rhythm

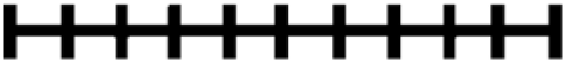
In the game player actions like jump or shoot are corresponding to the rhythm. There will be spesific time intervals for those events to start end end. To determine those time intervals criterias are can be the length, density and the type of the beat.



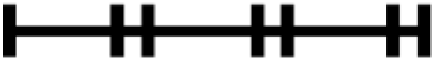
40s, low density, regular type



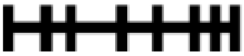
40s, mid density, regular type



40s, high density, regular type



30s, mid density, swing type



20s, high density, random type

Design of the Game

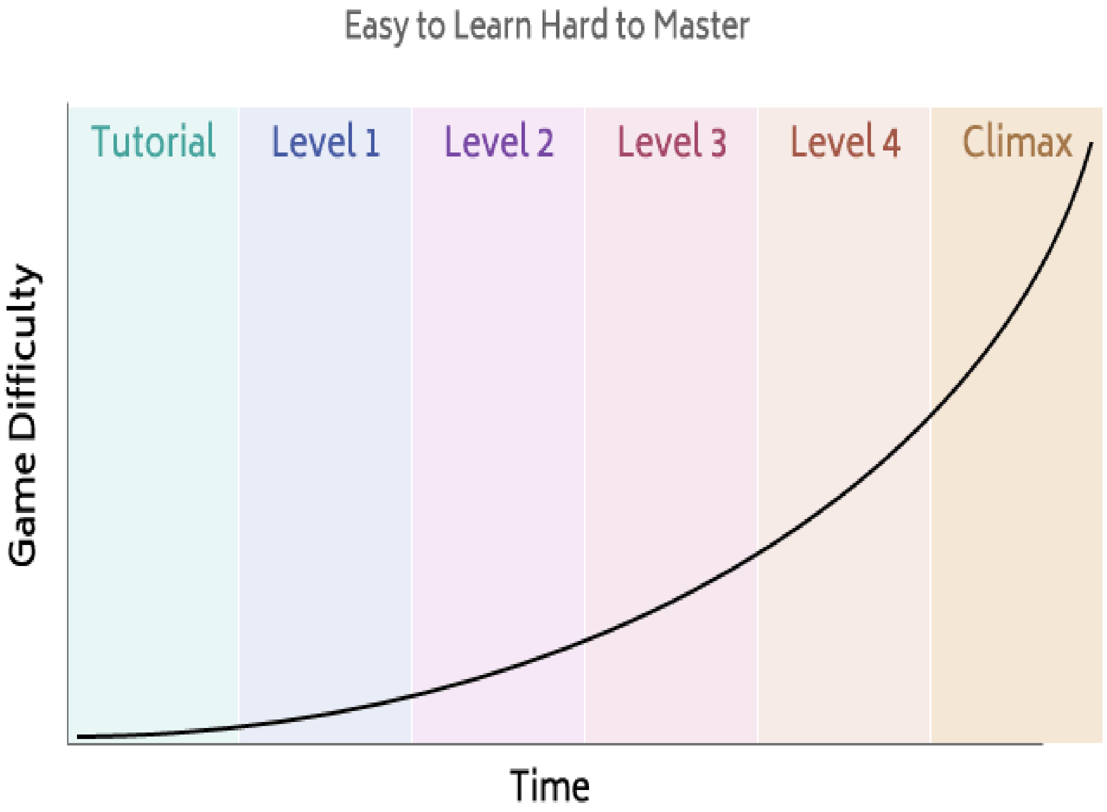
In general, on the design of our game we can speak of;

- **Level Design (world of the game)**
- **Characters Design (the main motive of the character)**
- **In terms of the motive of the character, we can include a background story. So, Storyline Design.**
- **One of the main parts of the game, Design of the Musics.**
- **Visual, Art Design.**

Level Design

In terms of the design of the game levels, we have a high opinion of creating a world that is, intriguing, immersive, fun to play, and also has the right difficulty curve that ensures most types of players getting progress. At this point, we are aiming for the "Easy to learn, hard to master" model.

Furthermore, we are planning to have different types of level sets. Meaning by that, as the game progresses, the elements of the game world will show variety in design factors.



Character Design

The main character has it's own; art design, animations, backstory-motive… There is a character that we work on within this scope. But until the designing process ends, we use a temporary character in our demo.

Storyline Design

In fact, in this project, we keep the gameplay over the storyline as prementioned in other reports. However, we are aware of, providing a motivation and even a basic background story to the main character is essential.

Musical Design

One of the most crucial components that are forming the atmosphere in a game is undoubtedly the music. Especially, in our game, the music has its own specific role on the gameplay.

Moreover, music in our game is one of the factors that can affect the level design.

We are still working on implementing the rhythm system as we desire.

Art Design

This part is completely dependent on aesthetic perception.

We will not be aiming to have high-end graphics but a minimalist, suitable for the main art theme, sharp graphics.

As in the character part, we are temporarily using an environmental asset to work on the demo version of the game.

Prototype of The Game

In this short-term, we have created a demo version of the game which has the main features of a 2D platformer game to work on the details that we want to implement in the game, such as the rhythm system.

So, in this version of the demo, besides the visual design part such as sprites, artworks, environmental assets which are temporary and going to be changed, we have those main basics to build our game on them:

Fig. Conceptual Model of the Game Components. (from previously report)

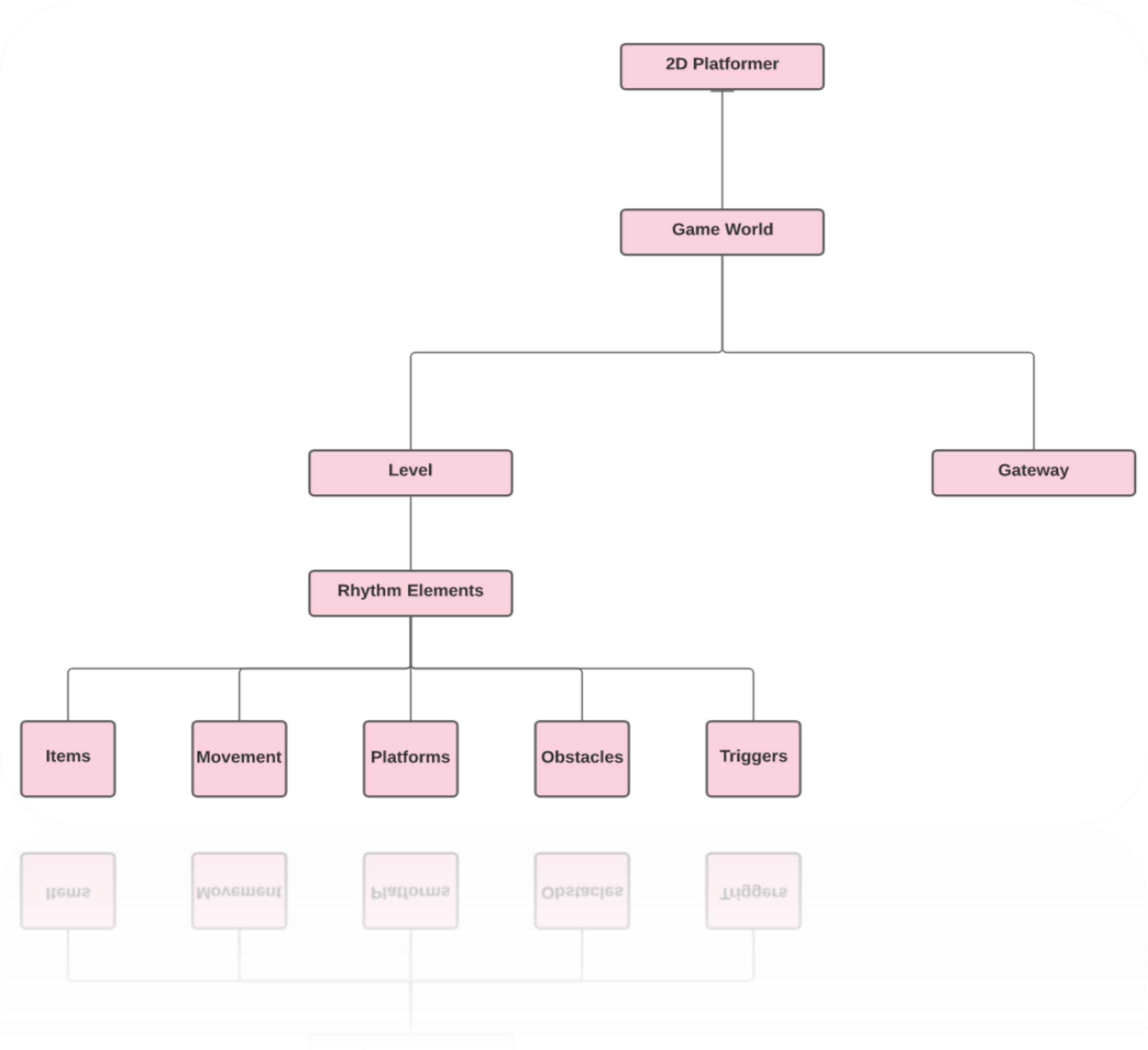
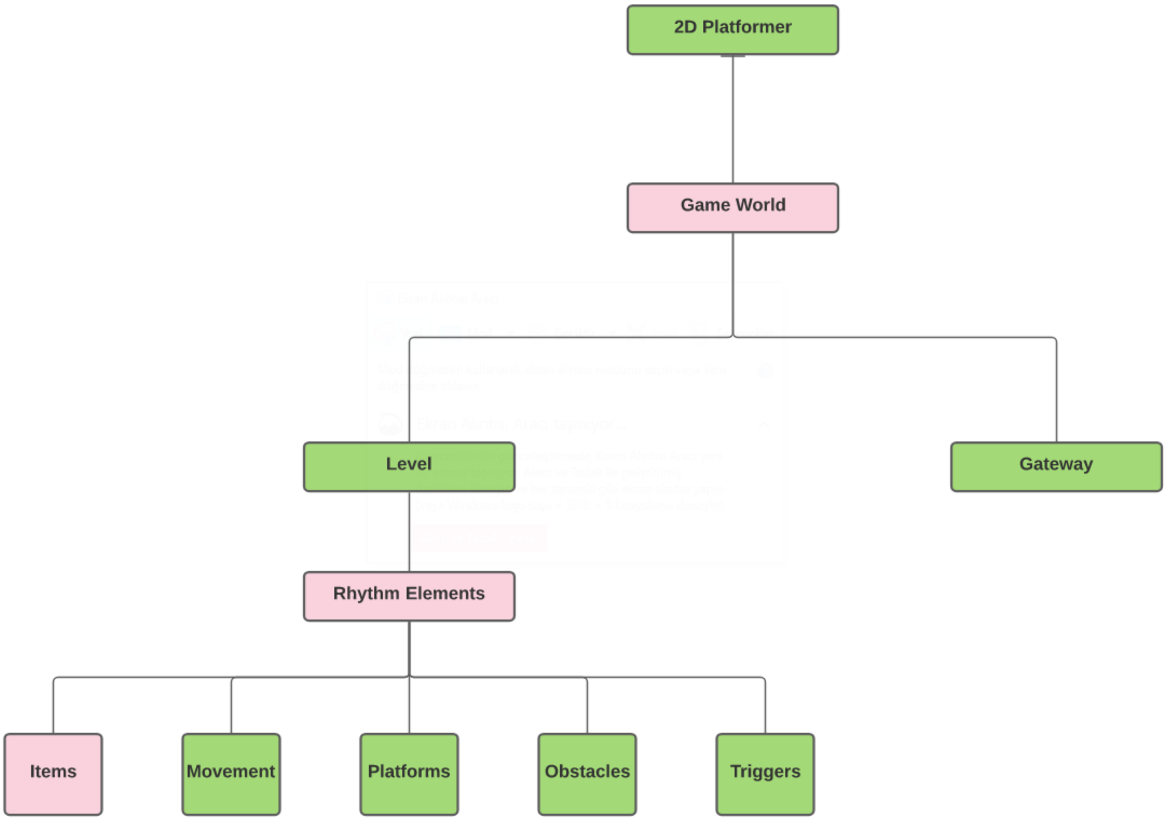
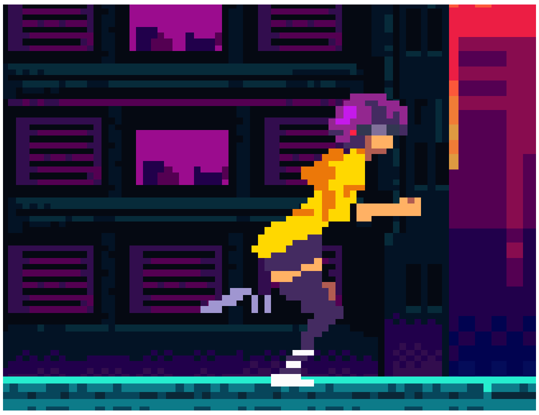


Fig. Basics of the Game Components that implemented into the demo.

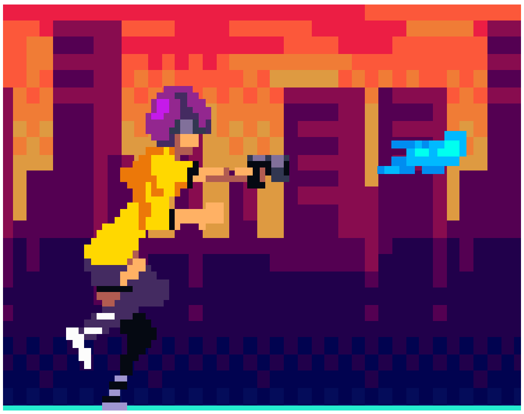


a) ***Items:*** We didn't go that deep in the demo. Items will be implemented in the next versions.

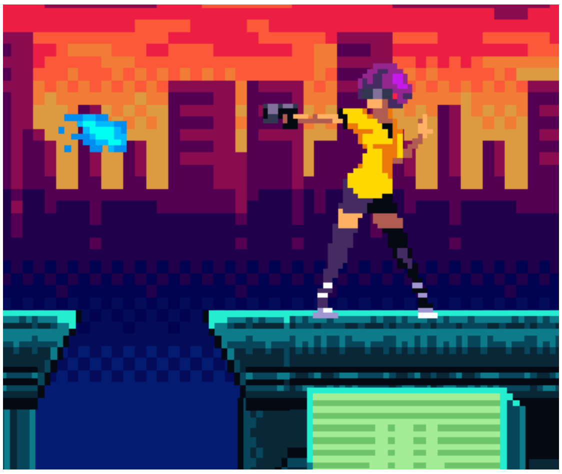
b) ***Movement:*** The main parts of the character's movements have been implemented into the demo. The main character can run, jump, and shoot with specific animations.



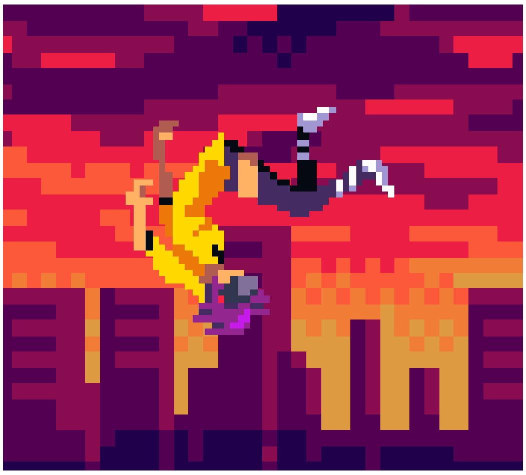
Running



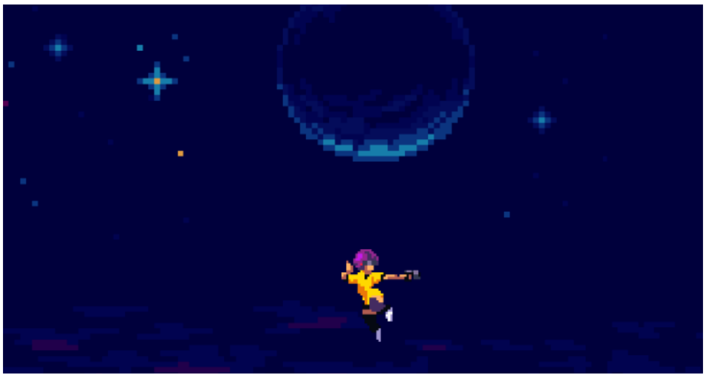
Running while shooting



Shooting while standing



Jumping

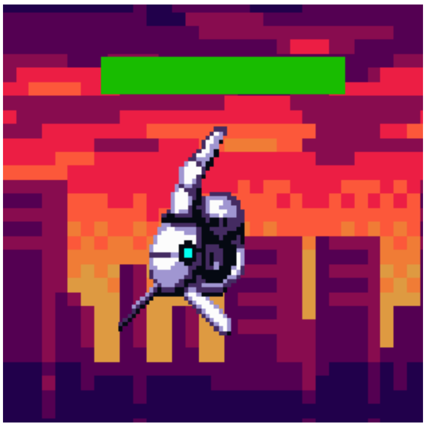


Gliding

a) ***Platform:*** Fundamental parts of the game, platforms designed and implemented to the game. Surface diversity will be increased in the next versions.

b) ***Obstacles:*** As an obstacle, firstly an enemy type is designed and added into the game. "The Drone" can damage the player. It has it's own movement pattern and can be destroyed by shooting.

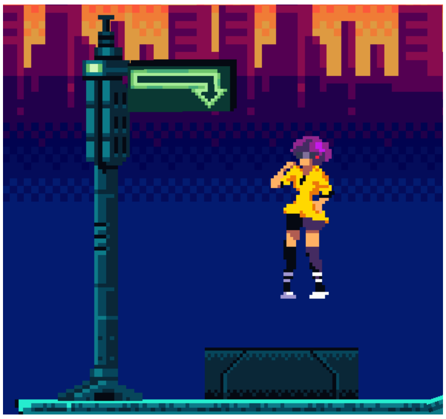
In the next versions, "The Turret" will be also implemented in the game. Other than that, there will be kind of obstacles that player would fall in it or have to be dodged it.



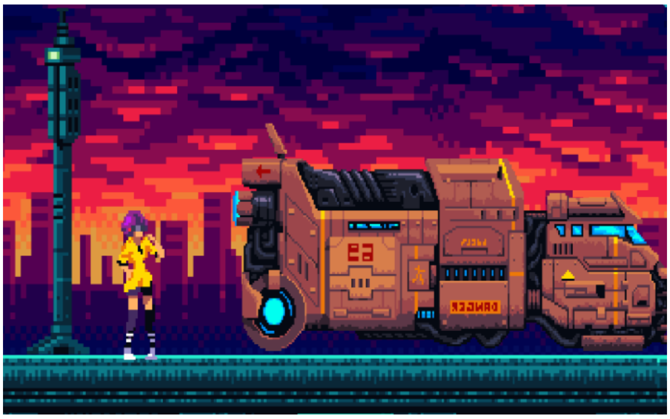
The Drone

a) ***Triggers:*** As prementioned in the previous reports, an example of the surface that gives jump boost has been implemented to the Demo Level. Also, buttons, doors, pressure plates etc. will be in the next versions.

The Player can go to the next level by reaching the vehicle that waits at the end of the level.



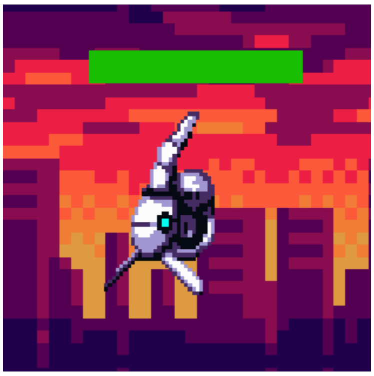
Jump Boost



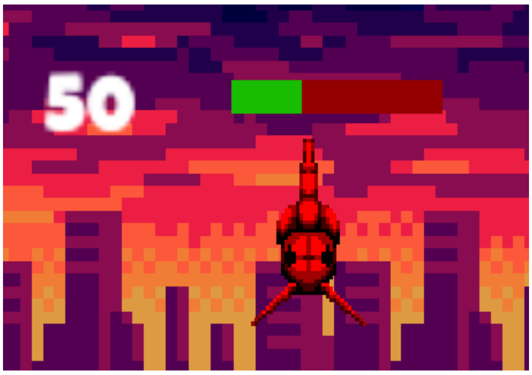
The Vehicle

c) ***UI Elements:*** A basic main menu has been implemented into the game. The player can check the current control buttons of the game from that menu before playing the game.

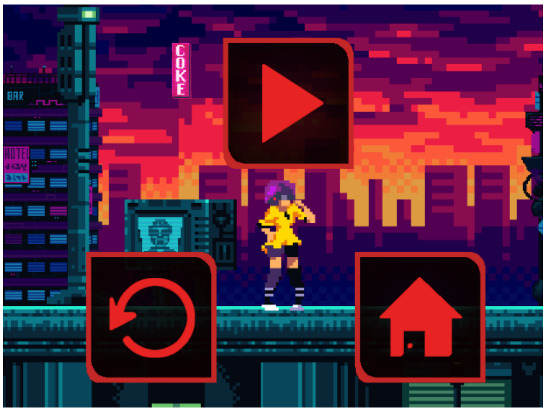
- An In-Game pause menu also implemented which includes a Restart Button, Home Button, Unpause Button.
- End-Level and Dead-Menu also implemented into the game.
- Player has its own health-bar on the middle-bottom of the screen.
- Enemies have their healthbars, attachted to the top of them.



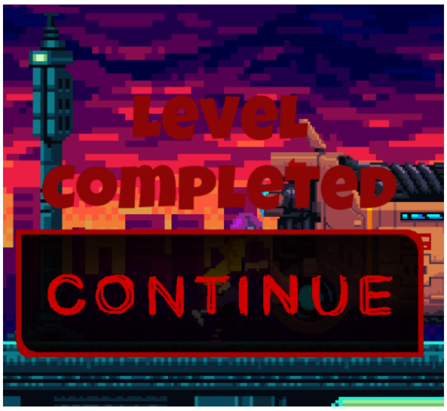
Enemy: Full hp bar



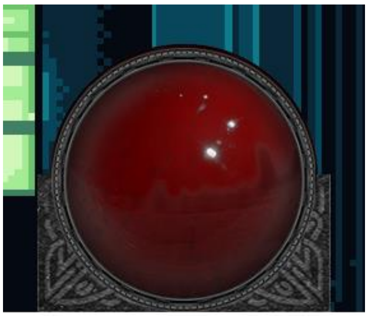
Enemy: Damage taken



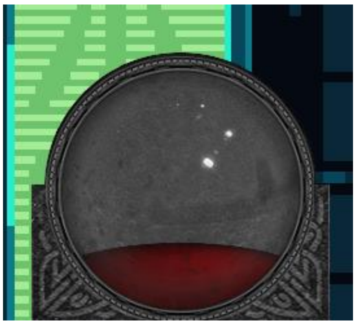
In-Game Pause Menu



End-Level Menu



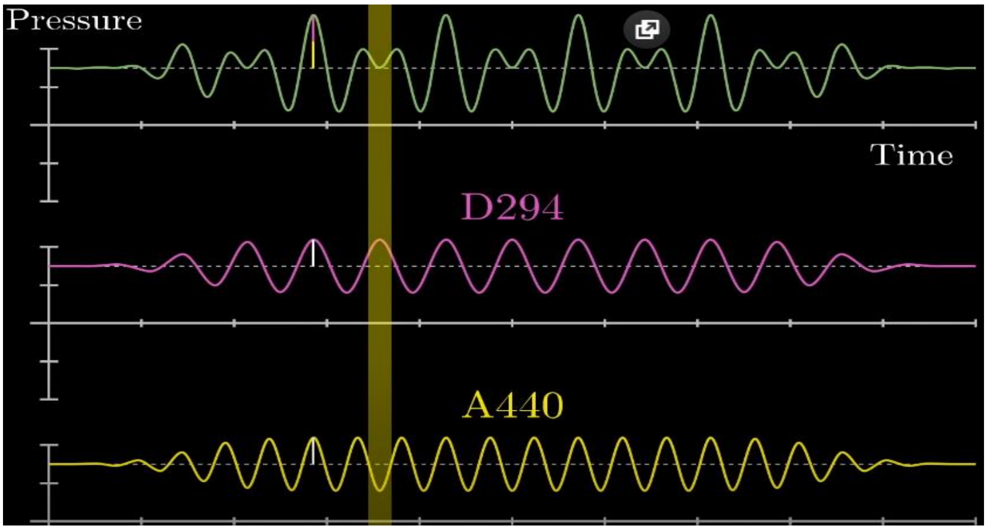
Player: Full Hp Bar



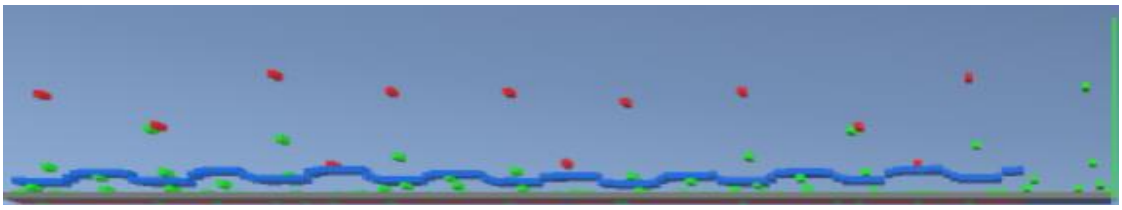
Player: Hp Bar

Beat Mapping Algorithm and Audio Analysis

In most of the rhythm-based games, beat mapping is being used. It is being used for several reasons. By applying the beat mapping to an audio file, you can basicly seperate the overlapping frequencies to many frequencies. The information like seperated instrument frequencies, musical notes that are overlapping, and audio's tempo can be gathered through beat mapping.



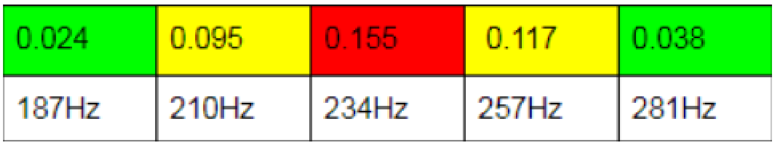
Above we can see, how combinated high and low frequencies lowers the average frequency in the exact time and creates repetitive frequencies. So if we consider the music has a lot more than two combined frequencies that comes from different type of instruments. Seperating them gives us good rhythmic information about the audio that we are working on as well.



This is an example for beat mapping that is done on an audio. We can see the evenly spaced peaks that occur which we can use to find the average tempo of the audio. In the giving example peaks occur every 0.414 seconds which is equivalent of 145 BPM.

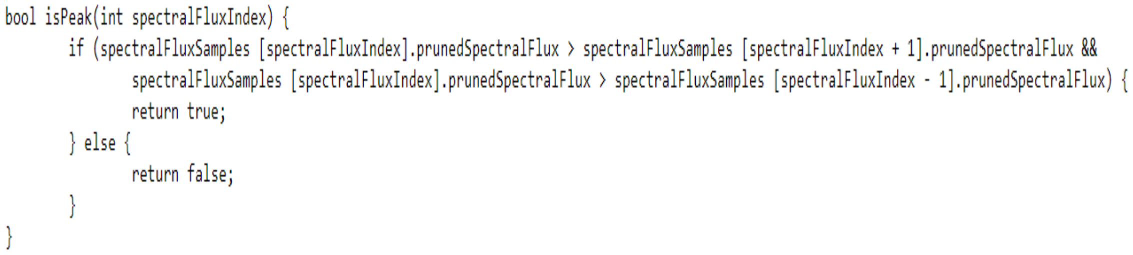
Beat Mapping Algorithm and Example Methods

Like most of the other parts of our project, we will be working on Unity for the audio part as well. Before getting into, we should know that all the audio analysis, tempo tracking, and beat mapping should not be done real-time since it will lag out because we are working over 20,000,000 samples average 4 minute of an audio. To make the process faster, we convert the stereo audio file to mono as left and right panned audios and combine them together where it directly halves the 20,000,000 samples. To get the spectrum data from these samples, Unity's own script can be used which is called "AudioSource.GetSpectrumData" and perform the Fast Fourier Transform over this data with it. By doing that, we get information about the audio's amplitude for specific frequencies where peaks can be already seen between frequencies through amplitude



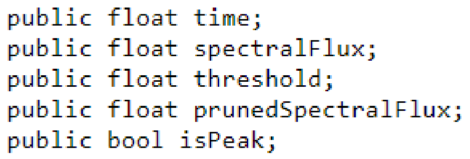
0.024	0.095	0.155	0.117	0.038
187Hz	210Hz	234Hz	257Hz	281Hz

After collecting these spectral data through audio, rest is about continuously getting spectral data and checking the spectral flux to see the significant frequency changes. Spectral flux changes can be collected by comparing every amplitude of the spectrum with previous and next frame's amplitude of spectrum.



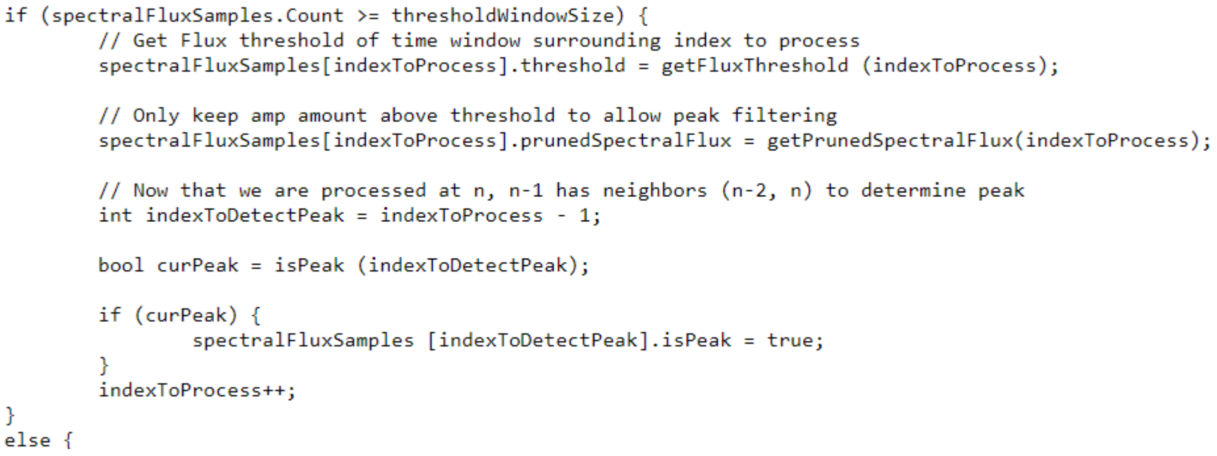
```
bool isPeak(int spectralFluxIndex) {
        if (spectralFluxSamples [spectralFluxIndex].prunedSpectralFlux > spectralFluxSamples [spectralFluxIndex + 1].prunedSpectralFlux &&
                spectralFluxSamples [spectralFluxIndex].prunedSpectralFlux > spectralFluxSamples [spectralFluxIndex - 1].prunedSpectralFlux) {
                return true;
        } else {
                return false;
        }
}
```

So, what informations will we get at the end by doing beat mapping?



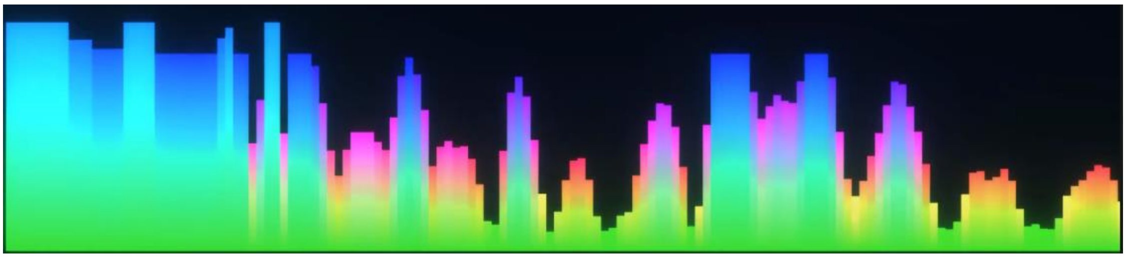
```
public float time;
public float spectralFlux;
public float threshold;
public float prunedSpectralFlux;
public bool isPeak;
```

Most of the information to define the tempo of the audio comes from the spectral flux. Frequencies that are compressed through limit level of the threshold gives us the peaks which appears when the frequency level gets above the threshold frequencies but will be limited at where threshold level is set. These limited peak index distances will give us the beats per minute at the end.



```
if (spectralFluxSamples.Count >= thresholdWindowSize) {
        // Get Flux threshold of time window surrounding index to process
        spectralFluxSamples[indexToProcess].threshold = getFluxThreshold (indexToProcess);

        // Only keep amp amount above threshold to allow peak filtering
        spectralFluxSamples[indexToProcess].prunedSpectralFlux = getPrunedSpectralFlux(indexToProcess);

        // Now that we are processed at n, n-1 has neighbors (n-2, n) to determine peak
        int indexToDetectPeak = indexToProcess - 1;

        bool curPeak = isPeak (indexToDetectPeak);

        if (curPeak) {
                spectralFluxSamples [indexToDetectPeak].isPeak = true;
        }
        indexToProcess++;
}
else {
```

While we are implementing the game’s most important mechanic with this way, we can add some visuals for the player to understand or feel the tempo of the music way better. After the beat detection, we will basicly have an audio spectrum data in our hand. By visualizing this data, we can put audio spectrum to the game as an indicator as well.



There won’t be need for a dataset for audio part since we will be getting our own data through audios that will be playing and we will be composing our own music for specific themes and maps.

References

https://mad-gfx.com/news/2018/7/15/10-principles-of-good-level-design

https://www.gamasutra.com/blogs/DanTaylor/20130929/196791/Ten_Principles_of_Good_Level_Design_Part_1.php

https://users.soe.ucsc.edu/~ejw/papers/Smith-Launchpad-TCIAIG-2011.pdf

https://dl.acm.org/doi/abs/10.1145/1536513.1536548?download=true

https://www.bulletsperminute.com/presskit

https://www.reddit.com/r/gamedev/comments/13y26t/how_do_rhythm_games_stay_in_sync_with_the_music/

Beginning Game Level Design, John Feil, Marc Scattergood, 1592004342, 9781592004348, 2005.

https://www.researchgate.net/publication/337109314_Procedural_Content_Generation_of_Rhythm_Games_Using_Deep_Learning_Methods

https://medium.com/giant-scam/algorithmic-beat-mapping-in-unity-preprocessed-audio-analysis-d41c339c135a

https://medium.com/giant-scam/algorithmic-beat-mapping-in-unity-intro-d4c2c25d2f27

https://www.researchgate.net/publication/337109314_Procedural_Content_Generation_of_Rhythm_Games_Using_Deep_Learning_Methods

https://medium.com/giant-scam/algorithmic-beat-mapping-in-unity-real-time-audio-analysis-using-the-unity-api-6e9595823ce4

https://prajwalsouza.github.io/Experiments/Fourier-Transform-Visualization.html

https://www.mono-project.com/archived/threadsbeginnersguide/

https://www.badlogicgames.com/wordpress/?p=99