Dance 24/7

- Game design software

- C# scripting for use in Unity

- Python scripting for .wav audio sample data extraction

- 3D model design for in game instrument assets

- Audio processing for use in Unity

- Audio production

Taylor Ramsay
CSUF - Computer Science Undergraduate
Inclusive Coding Festival 2022
Team ID: GD52216

Table of contents…….

Introduction

The mission of this project is to combine several of my passions (music, video games, programming) into a creative endeavor for personal fulfillment, to produce an engaging experience to share with others, as well as provide myself with the opportunity to work on a large scale project that has deep personal meaning. The end goal is to produce a fully functional 3rd-person 3d exploration game that relies on music and music visualization to tell a story and express emotion without the use of language.

To accomplish this I decided to develop a game using Unity to produce an experience that relies heavily on music and spatial audio placement. I used python to write scripts to extract audio data from .wav audio files that I produced in Ableton Live(audio production software) to influence the visual appearance of in-game objects such as instrument/NPC(non-player character) objects which emit spatial audio that responds to player character distance and position.

Having no experience in Unity, this project also involved much learning and research in using Unity, scripting in C#, audio processing in FMOD studio, and 3D modeling Blender.

In the context of a game, this project should satisfy the general needs that a user might have; this includes coherent gameplay, entertainment, inspiration, stimulation, and relaxation.

The flow of this game is designed to provide a low-challenge audio/visual experience to the player in which they traverse small to medium scale environments where the general goal is to locate and recruit other game objects which emit music.

Planning

The requirements defined at the beginning of this project are as follows:

- Playable demo level which provides a basic representation of gameplay
- Functionality for 3rd-person input/movement controls
- Dynamic visual system that extracts audio data from a .wav file and writes data to a .csv to be read in Unity and influence the appearance of in-game objects
- Basic combat system in which friendly/non-friendly NPCs engage with each other and the player
- A system which allows interaction between the player and friendly NPCs (recruit NPC to team, enable NPCs to follow the player, revive NPCs when defeated in combat)

The goals needed in order for me to complete this project are as follows:

- Learn to use the Unity game development platform
- Familiarize myself with C# programming language
- Learn to create and export assets from Blender modeling software for use in Unity
- Learn to use FMOD studio to create spatial and stereo audio assets for use in Unity
- Write python scripts which can extract audio data from .wav files + use data to manipulate Unity assets

Roles of each team member and responsibilities:

Taylor Ramsay(self)

- Unity game development
- C# scripting

- Python scripting
- Audio production
- Audio processing
- Instrument 3d modeling

Bryan Pawlowski
- Mentor/Support

Design

Software Stack:
- Unity - game development platform
- Microsoft Visual Studio - C# programming
- Visual Studio Code - Python scripting
- Blender - 3d modeling
- Ableton Live - Audio production
- FMOD Studio - Audio processing

Prototypes of Application:
Due to the scale of game development, personal learning requirements, and timeline of the ICF event, the submitted project is a prototype/ basic representation of the final product.

Implementation

Having never developed a project of this scale, experimentation/improvisation played a big part in the code design process. In general, object oriented programming was used to implement the interactions between game objects, the game world, and the user. Each system in the game relies on a manager class to delegate instructions between game objects, such as combat, navigation, and the player character.
For audio visualization, the python scripts I wrote are relatively straightforward and can be found at the following github repository or the parent directory of the project's submission repository.
https://github.com/TaylorRamsay/.wav-sample-extractor
To achieve the desired effect of influence in game objects, I extracted data from .wav audio files using a python library called Aubio (https://pypi.org/project/aubio/), which was designed for use with numpy to process and analyze audio signals. Once extracted this data is read into Unity to influence the appearance of in-game objects, synced in time with their respective audio track. I decided to have this processing occur prior to runtime to reduce the demand of running the game, as well as ensure that the data extracted is of sufficient quality for its intended purpose. Overall this process could be improved in terms of producing data that is more representationally accurate of its parent audio file.

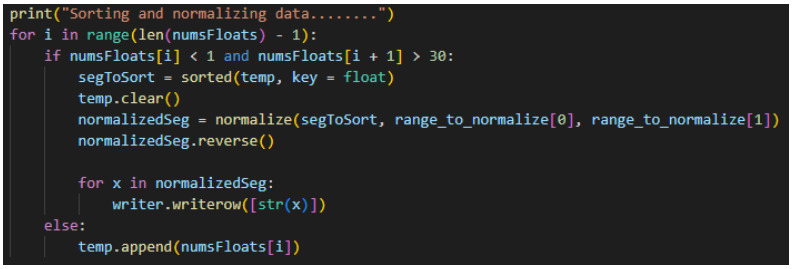
Data Structures and Algorithms

For this project the data structures I used were those build into C# and python. In C# I used lists to manage friendly/enemy NPCs, and audio data. These lists involving NPCs were used to manage their state inside/outside of combat while the audio data lists were used to access data to influence object appearances.

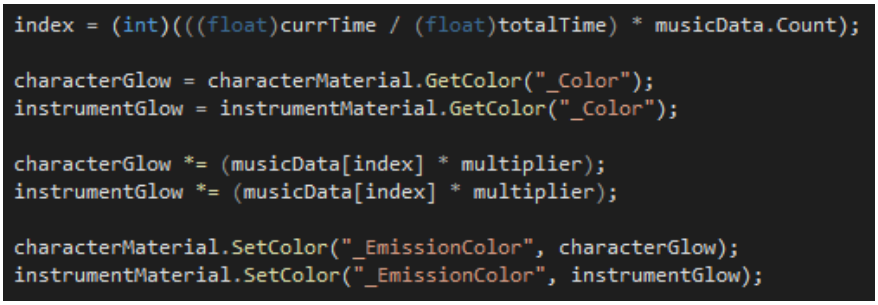
Algorithms:

- **Audio extraction**
  - This is an example of how data was extracted from a .wav file containing chords from a synthesizer. The file is stored as an aubio.source file, the sample data is then extracted and stored as a list of strings, converted to floats, then sorted and normalized according to the needs of the track. This specific example required the data to be processed in chunks before being stored into the final list and written to a .csv for use in Unity.
  - 

```
#Aubio
chiraAubio = aubio.source("C:\\Users\FSK8475\Documents\GitHub\.wav-sample-extractor\\ChiraStems\\Chords.wav")
total_read = 0
nums = []

print("Extracting data using Aubio library.......")
while True:
    samples, read = chiraAubio()
    nums.append(np.format_float_positional(samples.sum()))
    total_read += read
    if read < chiraAubio.hop_size:
        break

# Convert string list nums to float list numsFloats
numsFloats = [float(ele) for ele in nums]
```

  - 

```
print("Sorting and normalizing data.......")
for i in range(len(numsFloats) - 1):
    if numsFloats[i] < 1 and numsFloats[i + 1] > 30:
        segToSort = sorted(temp, key = float)
        temp.clear()
        normalizedSeg = normalize(segToSort, range_to_normalize[0], range_to_normalize[1])
        normalizedSeg.reverse()

        for x in normalizedSeg:
            writer.writerow([str(x)])
    else:
        temp.append(numsFloats[i])
```

- **Audio/Visual synchronization**
  - With Bryans help, I was able to successfully synchronize visual manipulation of game objects in time with respective audio tracks with the following code. To find the index, a ratio is calculated based on current vs total track time, this is then multiplied by the size of the list containing audio data to produce an int which corresponds to the index at which the current track times audio data is located.
  - 

```
index = (int)(((float)currTime / (float)totalTime) * musicData.Count);

characterGlow = characterMaterial.GetColor("_Color");
instrumentGlow = instrumentMaterial.GetColor("_Color");

characterGlow *= (musicData[index] * multiplier);
instrumentGlow *= (musicData[index] * multiplier);

characterMaterial.SetColor("_EmissionColor", characterGlow);
instrumentMaterial.SetColor("_EmissionColor", instrumentGlow);
```

- **NPC following system**
  - This relatively straightforward algorithm handles how the friendly NPCs follow the player after being recruited, it contains a helper function which sets the first

follower to follow the player object and instructs the others for follow the preceding object in the bandMembers list

```
void Follow()
{
    if (!combat.activeCombat)
    {
        for (int i = 0; i < playerManager.bandMembers.Count; i++)
        {
            if (playerManager.bandMembers[i].isFollowing)
            {
                if (i == 0)
                {
                    PlayerFollower(playerManager.bandMembers[i]);
                }
                else
                {
                    playerManager.bandMembers[i].navAgent.destination = playerManager.bandMembers[i - 1].transform.position;
                }
            }
        }
    }
}
```

- Implementing NPC combat
    - Instructing how the NPCs target each other in combat involves list traversal and decision making based on whether or not the potential combat target is already targeted by another game object.

```
public void EnemyTargeting(NPC attacker)
{
    if (combat.activeCombat)
    {
        if (attacker.isTargeting == false)
        {
            for (int i = 0; i < playerManager.agroEnemies.Count; i++)
            {
                if (playerManager.agroEnemies[i].GetComponent<EnemyNPC>().targeted == false)
                {
                    attacker.combatTarget = playerManager.agroEnemies[i];
                    attacker.isTargeting = true;
                    playerManager.agroEnemies[i].GetComponent<EnemyNPC>().targeted = true;

                    goto after;
                } else if (i == playerManager.agroEnemies.Count - 1)
                {
                    attacker.combatTarget = playerManager.agroEnemies[0];
                    attacker.isTargeting = true;
                    goto after;
                }
            }
        } else
        {
            attacker.gameObject.transform.LookAt(attacker.combatTarget.transform);
        }
    after:;
    }
    if (attacker.combatTarget.GetComponent<StatManager>().IsDefeated())
    {
        attacker.isTargeting = false;
    }
}
```

- **UI/UX Design**
    - Limited time was spent on the UI of this project due to time limitations and the relatively low priority this held in the pursuit of building a functioning game prototype/demonstration. The current UI elements present include a basic health indicator and on-screen controls.

- **Performance Enhancements:**
    - In terms of performance enhancements, due to time and learning, much of the code can be optimized functionally and organizationally since all game structures needed to be quickly built to function on a practical level.

Conclusion

- **Final Project Impressions:**
    - This project was a very fun experience that provided a very stimulating challenge in terms of learning and creating. I feel very proud of what I was able to accomplish within the relatively short timeline of the ICF event and am excited to continue working to achieve my vision for this game.

- **Obstacles and Triumphs:**
    - Learning Unity
    - Experimenting with game design
    - Learning to model 3d objects in Blender
    - Successfully extracting/formatting meaningful data from .wav file for use in Unity
    - Translating ideas in my mind into functional game mechanics with 3d models and programming

- **Potential Next Steps to Project:**
    - Continue to improve current game mechanics as well as develop new ones and work towards vision of a complete game.

- **Overall Thoughts of Project:**
    - Overall I was very grateful for the opportunity to participate in this event, I feel that it provided me an extremely valuable experience in programming, as well as a tangible creation for my portfolio.