Rhythm Realm

By Donnell Maylor

What is Rhythm Realm?

Rhythm Realm is a action platformer about shapes that fight and defend to rhythms and songs. On a quest to make music for their kingdom. I got this idea when I was playing the game “Just some Shapes and Beats” and “Metal Hellsinger” for the first time. Inspired by how integral music was to game play and vice versa.

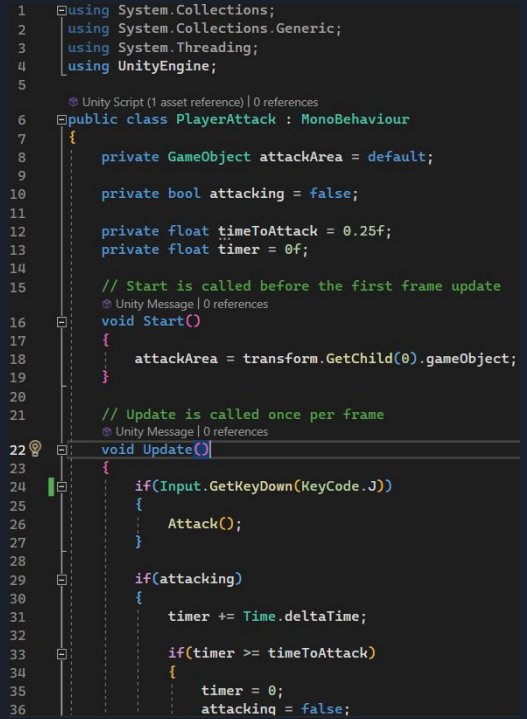
Required Resources

I used the game design program Unity and the website freesound to get my sound files

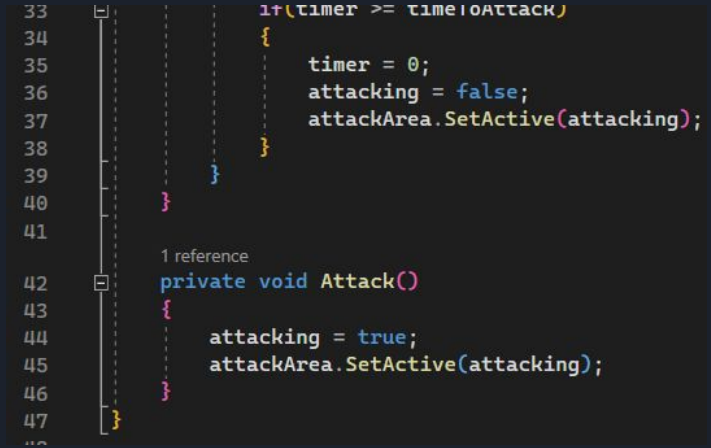
Coding - Health system

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

Unity Script (2 asset references) | 7 references
public class Health : MonoBehaviour
{
    [SerializeField] private int health = 100;

    public bool isPlayer = false;

    private int MAX_Health = 100;
    private object keyCode;


    public GameObject healthTextObj;
    private TextMeshPro healthtextMesh;
    public GameObject gameOverTextObj;
    // Update is called once per frame

    Unity Message | 0 references
    private void Start()
    {
        healthtextMesh = healthTextObj.GetComponent<TextMeshPro>();
        if (isPlayer )
        {
            gameOverTextObj.SetActive(false);

        }

    }

    Unity Message | 0 references
    void Update()
    {

        float yOffset = 3;
        healthTextObj.transform.position = transform.position + (yOffset * Vector3.up);
        healthtextMesh.text = health.ToString();
        /*

        if (Input.GetKeyDown(KeyCode.H))
        {
            Damage(10);
        }

        if(Input.GetKeyDown(KeyCode.J))
        {
            Heal(10);
        }
        */
    }


    1 reference
    public void SetHealth(int maxHealth, int health)
    {
        this.MAX_Health = maxHealth;
        this.health = health;
    }

    2 references
    private IEnumerator VisualIndicatos(Color color)
    {
        GetComponent<SpriteRenderer>().color = color;
        yield return new WaitForSeconds(0.15f);
        GetComponent<SpriteRenderer>().color = Color.white;

    }

    2 references
    public void Damage(int amount)
    {
        if(amount < 0)
        {
            throw new System.ArgumentOutOfRangeException("Cannot have negative Damage");
        }

        this.health -= amount;
        StartCoroutine(VisualIndicatos(Color.red));

        if(health <= 0)
        {
            Die();
        }
    }

    1 reference
    public void Heal(int amount)
    {
        if(amount < 0)
        {
            throw new System.ArgumentOutOfRangeException("Cannot have negative Damage");
        }

        bool wouldBeOverMaxHealth = health + amount > MAX_Health;
        StartCoroutine(VisualIndicatos(Color.green));

        if (wouldBeOverMaxHealth)
        {
            this.health = MAX_Health;
        }
        else
        {
            this.health += amount;
        }
    }

    1 reference
    private void Die()
    {
        Debug.Log("Dead");

        if (isPlayer)
        {
            gameOverTextObj.SetActive(true);
            healthTextObj.SetActive(false);


        }


        Destroy(gameObject);
    }
}
```

Coding - Player Attack

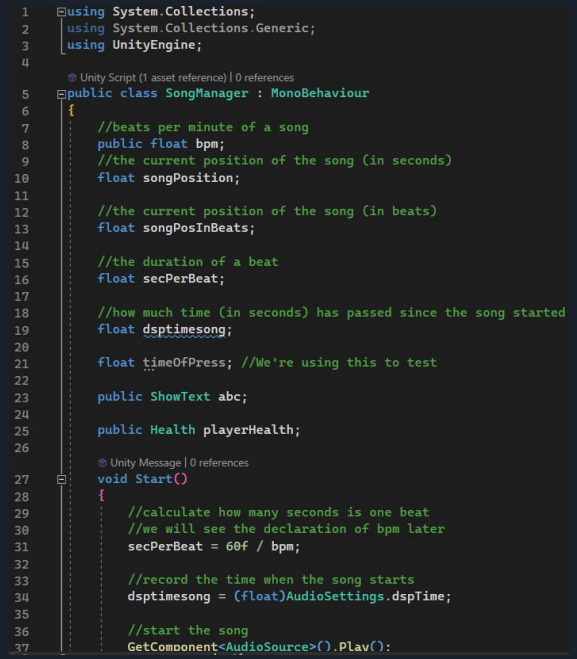


```
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class PlayerAttack : MonoBehaviour
{
    private GameObject attackArea = default;

    private bool attacking = false;

    private float timeToAttack = 0.25f;
    private float timer = 0f;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        attackArea = transform.GetChild(0).gameObject;
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.J))
        {
            Attack();
        }

        if(attacking)
        {
            timer += Time.deltaTime;

            if(timer >= timeToAttack)
            {
                timer = 0;
                attacking = false;
```

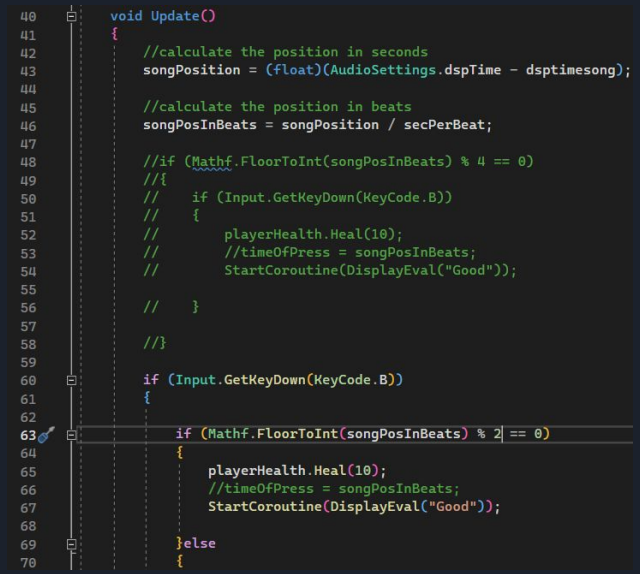


```
            if(timer >= timeToAttack)
            {
                timer = 0;
                attacking = false;
                attackArea.SetActive(attacking);
            }
        }
    }

    1 reference
    private void Attack()
    {
        attacking = true;
        attackArea.SetActive(attacking);
    }
}
```

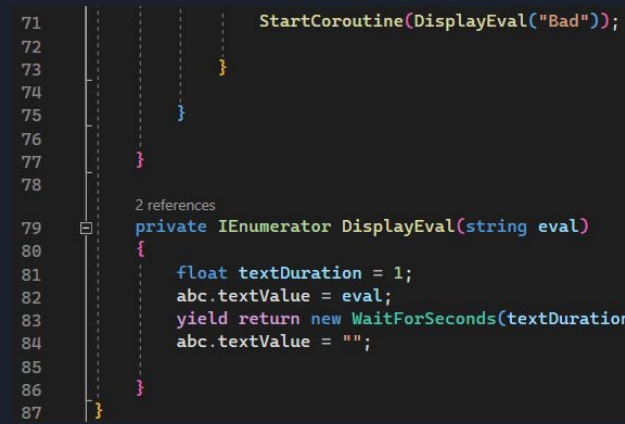
Coding - SongManager



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class SongManager : MonoBehaviour
{
    //beats per minute of a song
    public float bpm;
    //the current position of the song (in seconds)
    float songPosition;

    //the current position of the song (in beats)
    float songPosInBeats;

    //the duration of a beat
    float secPerBeat;

    //how much time (in seconds) has passed since the song started
    float dsptimesong;

    float timeOfPress; //We're using this to test

    public ShowText abc;

    public Health playerHealth;

    Unity Message | 0 references
    void Start()
    {
        //calculate how many seconds is one beat
        //we will see the declaration of bpm later
        secPerBeat = 60f / bpm;

        //record the time when the song starts
        dsptimesong = (float)AudioSettings.dspTime;

        //start the song
        GetComponent<AudioSource>().Play();
```



```
    void Update()
    {
        //calculate the position in seconds
        songPosition = (float)(AudioSettings.dspTime - dsptimesong);

        //calculate the position in beats
        songPosInBeats = songPosition / secPerBeat;

        //if (Mathf.FloorToInt(songPosInBeats) % 4 == 0)
        //{
        //    if (Input.GetKeyDown(KeyCode.B))
        //    {
        //        playerHealth.Heal(10);
        //        //timeOfPress = songPosInBeats;
        //        StartCoroutine(DisplayEval("Good"));

        //    }

        //}

        if (Input.GetKeyDown(KeyCode.B))
        {

            if (Mathf.FloorToInt(songPosInBeats) % 2 == 0)
            {
                playerHealth.Heal(10);
                //timeOfPress = songPosInBeats;
                StartCoroutine(DisplayEval("Good"));

            }else
            {
```



```
                StartCoroutine(DisplayEval("Bad"));

            }

        }

    }

    2 references
    private IEnumerator DisplayEval(string eval)
    {
        float textDuration = 1;
        abc.textValue = eval;
        yield return new WaitForSeconds(textDuration
        abc.textValue = "";

    }
}
```

Schedule

Donnel Game	Rhythem Realm	
Task	start date	end date
pick topic	02/16/23	02/20/23
craft parameters	02/20/23	02/22/23
milestones	02/20/23	02/23/23
wireframe model	02/24/23	02/27/23
select software	02/28/23	2/29/23
coding	2/30/23	03/07/23
compile game play	03/08/23	03/14/23
update	03/15/23	3/22/23
compile assets	03/23/23	03/27/23
update	03/28/23	04/04/23
test game	04/05/23	04/09/23
compile test outcomes	04/10/23	04/13/23
update	04/14/23	04/20/23
present proof of concept	4/22/23	04/26/23

Sprites

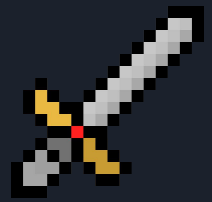
The Player

The Enemy

Obstacle

Finish line

Sword



Conclusion

What I expected from this project was a working build with a solid foundation to build off of. I want to get a better grasp of the mechanics of comdat in combination of the music to better merge music and gameplay.


The End