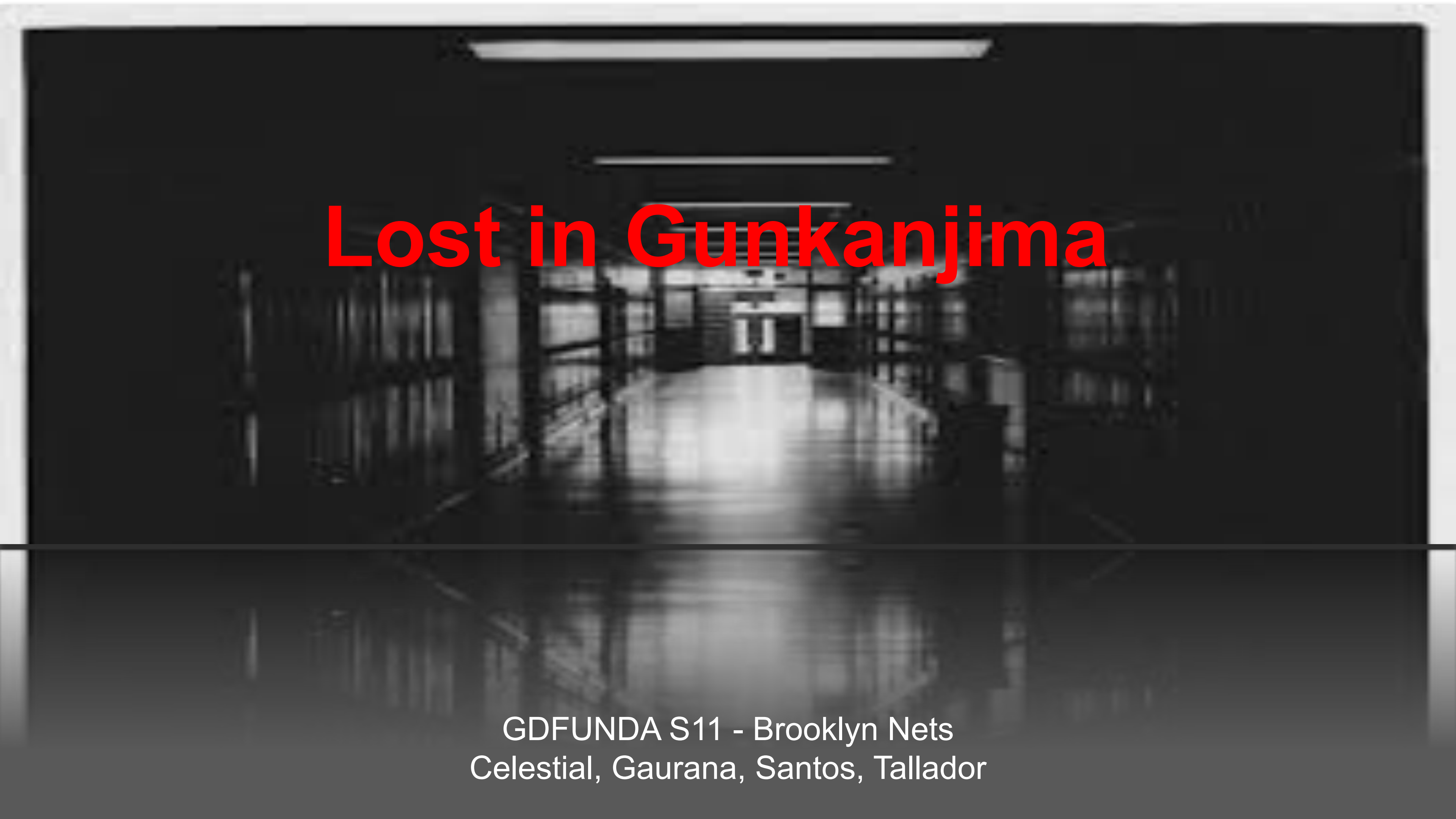


Lost in Gunkanjima

GDFUNDA S11 - Brooklyn Nets
Celestial, Gaurana, Santos, Tallador

About the Game

Setting: Abandoned school

Description: You are at an abandoned old school building with no recollection of how you got there. As you arrive, an unknown entity takes over your head and compels you to fulfill a task of finding an object by riddles. You can’t escape the school building if you can’t finish the task and you will be lost forever in the school. However, you will be set free by the unknown entity if you fulfill the task well.

Armed with just a flashlight and your knowledge in riddles, it is a race to finish the task before the time runs out.

Are you prepared for the task?

Game Controls

- **W** - move forward
- **A** - move left
- **S** - move backward
- **D**- move right
- **Left Click (hold)** - interact/hold object
- **Left Click (release)** - drop object
- **Spacebar** - Jump
- **Left Shift** - Run
- **F -** flashlight

Relevance to Theme

- When we learned that the theme for this Game Jam was Lost and Found, we thought of making the player to search for items in the dark. Similar to games such as Slender and the feel of J-Horror games, we wanted to put the player into a situation wherein time and how you solve tasks become a factor to success or failure.
- Our initial idea was that the player will find a lost object from the school in order to clear the game. In order to find the lost object however, the player will have to traverse a complex maze. We originally planned it to be a 3rd person game, but we wanted the player immersion to be effective so we switched to 1st person. This concept was based on mystery games which revolves around making the player find things to clear the game such as the aforementioned “Slender”. We also believe that the theme “Lost and Found” can fit to the mystery/horror genre of games.

Core Features Required

Core Feature 1 - Physics and Rigidbodies

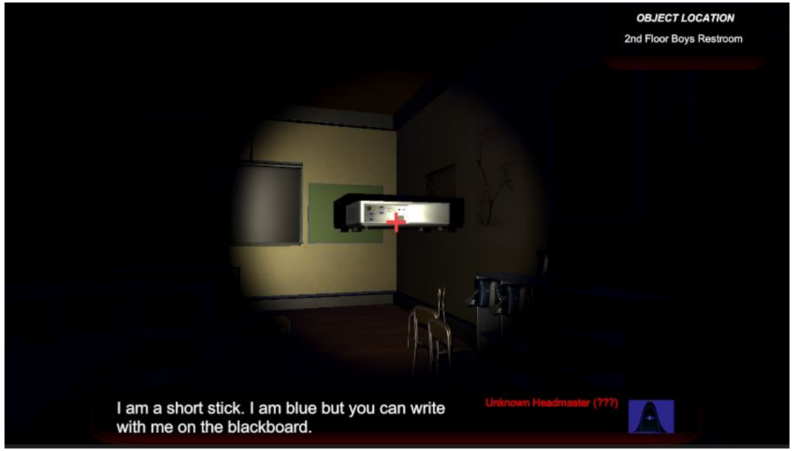
Rigidbodies and Physics was used all throughout the game. It was implemented as a core feature of our interactable game objects as well as the interaction of characters in the game world.

Features applied:

- Player can drag and drop some objects in the game. The input here is the left mouse button.
- Player can choose to walk (W) or run (hold LSHIFT with W).
- Player can choose to also jump (Spacebar).
- The collision detection was also made possible by implementing interactable objects as rigidbodies (which can be seen in the event broadcasting slides)

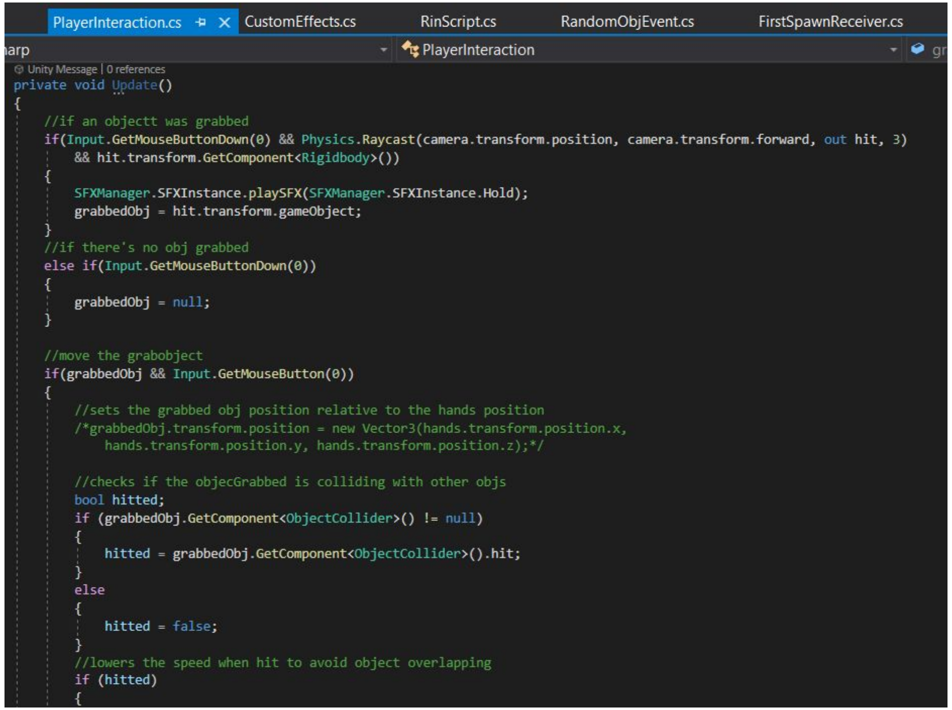
Features with Physics and Rigidbodies

Player can drag and drop some objects in the game. The input here is the left mouse button.



I am a short stick. I am blue but you can write with me on the blackboard.

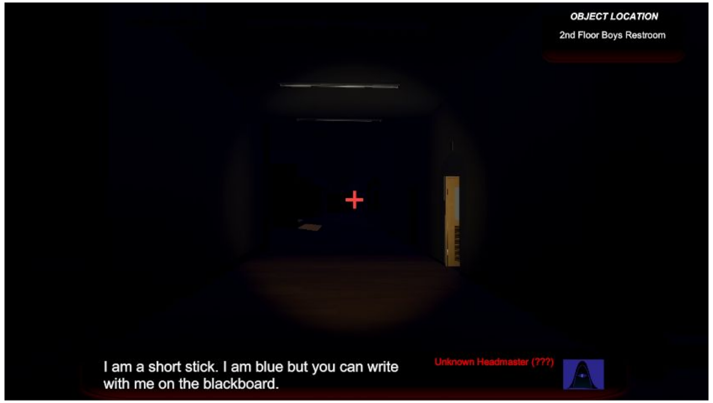
Player is dragging the projector; item can be dropped



```
private void Update()
{
    //if an objectt was grabbed
    if(Input.GetMouseButtonDown(0) && Physics.Raycast(camera.transform.position, camera.transform.forward, out hit, 3)
        && hit.transform.GetComponent<Rigidbody>())
    {
        SFXManager.SFXInstance.playSFX(SFXManager.SFXInstance.Hold);
        grabbedObj = hit.transform.gameObject;
    }
    //if there's no obj grabbed
    else if(Input.GetMouseButtonDown(0))
    {
        grabbedObj = null;
    }

    //move the grabobject
    if(grabbedObj && Input.GetMouseButton(0))
    {
        //sets the grabbed obj position relative to the hands position
        /*grabbedObj.transform.position = new Vector3(hands.transform.position.x,
            hands.transform.position.y, hands.transform.position.z);*/

        //checks if the objecGrabbed is colliding with other objs
        bool hitted;
        if (grabbedObj.GetComponent<ObjectCollider>() != null)
        {
            hitted = grabbedObj.GetComponent<ObjectCollider>().hit;
        }
        else
        {
            hitted = false;
        }
        //lowers the speed when hit to avoid object overlapping
        if (hitted)
        {
```

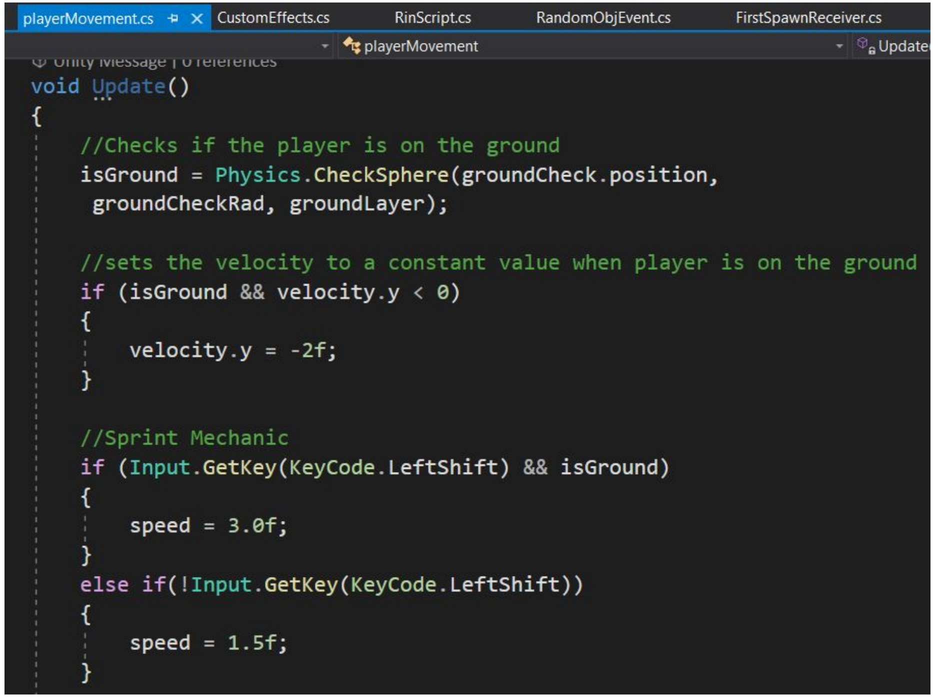
Features with Physics and Rigidbodies

Player can choose to walk or run.



I am a short stick. I am blue but you can write with me on the blackboard.

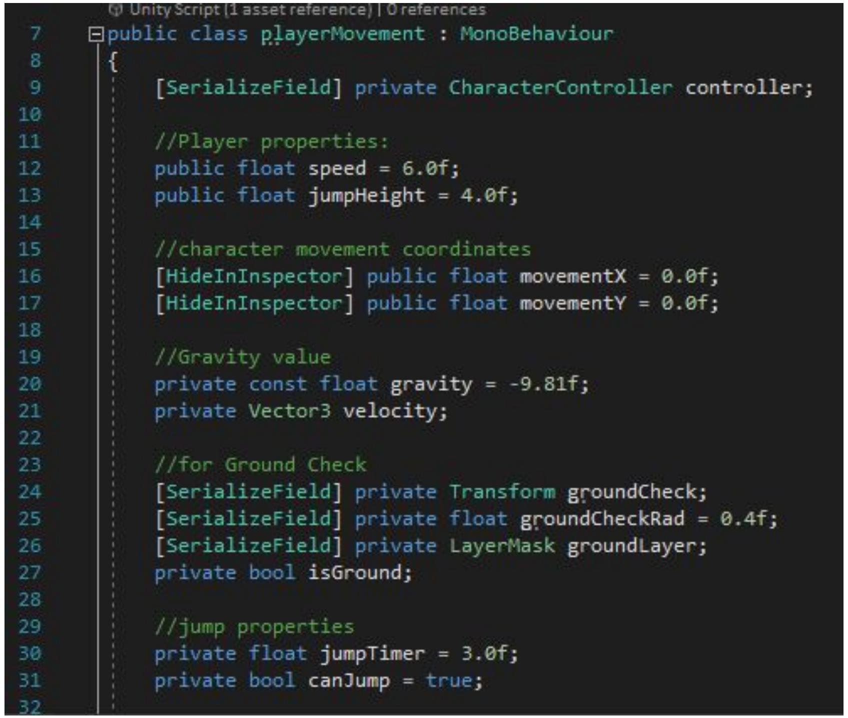
Player is Running on the corridor



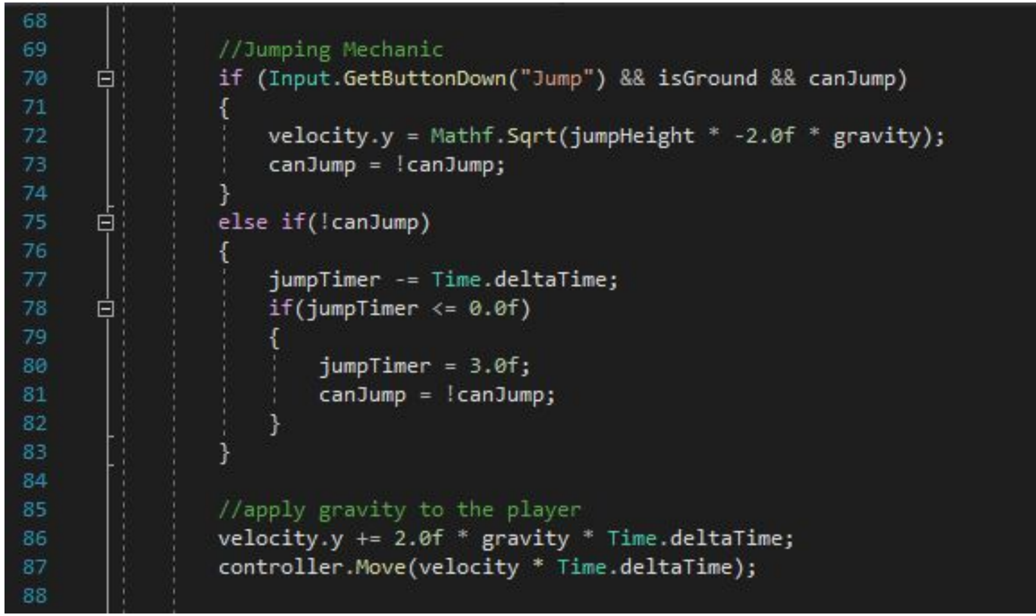
```
void Update()
{
    //Checks if the player is on the ground
    isGround = Physics.CheckSphere(groundCheck.position,
     groundCheckRad, groundLayer);

    //sets the velocity to a constant value when player is on the ground
    if (isGround && velocity.y < 0)
    {
        velocity.y = -2f;
    }

    //Sprint Mechanic
    if (Input.GetKey(KeyCode.LeftShift) && isGround)
    {
        speed = 3.0f;
    }
    else if(!Input.GetKey(KeyCode.LeftShift))
    {
        speed = 1.5f;
    }
```

Features with Physics and Rigidbodies

Players can also choose to jump



```
Unity Script (1 asset reference) | 0 references
public class playerMovement : MonoBehaviour
{
    [SerializeField] private CharacterController controller;

    //Player properties:
    public float speed = 6.0f;
    public float jumpHeight = 4.0f;

    //character movement coordinates
    [HideInInspector] public float movementX = 0.0f;
    [HideInInspector] public float movementY = 0.0f;

    //Gravity value
    private const float gravity = -9.81f;
    private Vector3 velocity;

    //for Ground Check
    [SerializeField] private Transform groundCheck;
    [SerializeField] private float groundCheckRad = 0.4f;
    [SerializeField] private LayerMask groundLayer;
    private bool isGround;

    //jump properties
    private float jumpTimer = 3.0f;
    private bool canJump = true;

```



```

            //Jumping Mechanic
            if (Input.GetButtonDown("Jump") && isGround && canJump)
            {
                velocity.y = Mathf.Sqrt(jumpHeight * -2.0f * gravity);
                canJump = !canJump;
            }
            else if(!canJump)
            {
                jumpTimer -= Time.deltaTime;
                if(jumpTimer <= 0.0f)
                {
                    jumpTimer = 3.0f;
                    canJump = !canJump;
                }
            }

            //apply gravity to the player
            velocity.y += 2.0f * gravity * Time.deltaTime;
            controller.Move(velocity * Time.deltaTime);

```

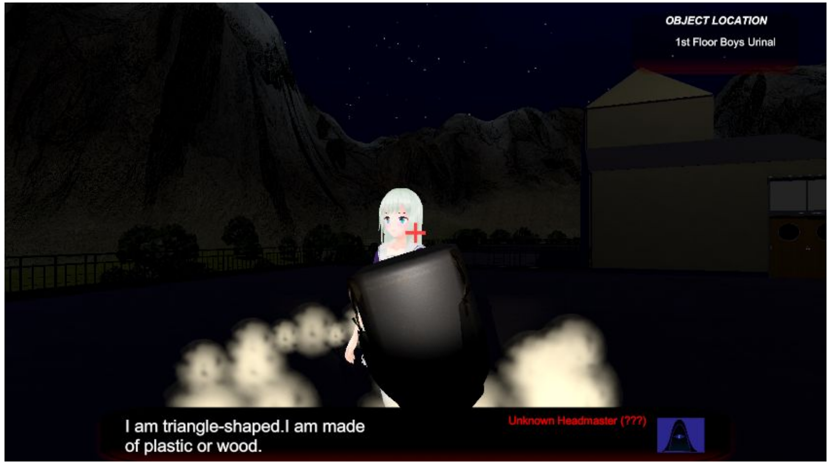
Core Feature 2 - Event Broadcasting

Our Event Broadcasting core feature was responsible for handling a big task in our game and it enabled us to divide the work into small tasks. This core feature acted as our events management system when it came to the following game features:

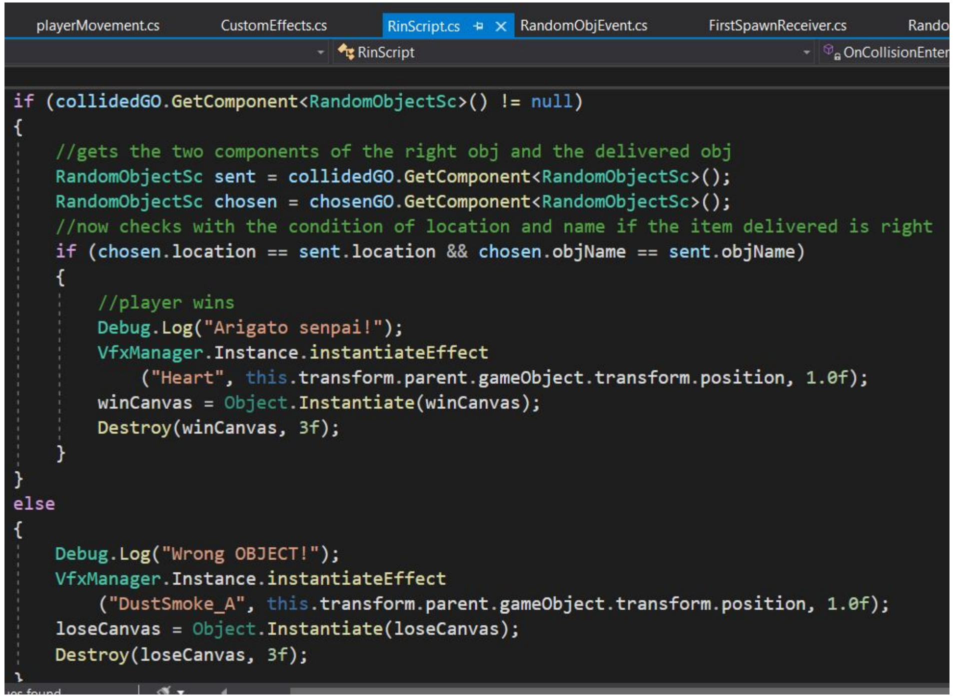
- Object Collision Detection
- Random Object Generation and Placement
- Random Placement of Player and Headmaster prefabs
- SFX and VFX

Features with Event Broadcasting

Object Collision Detection



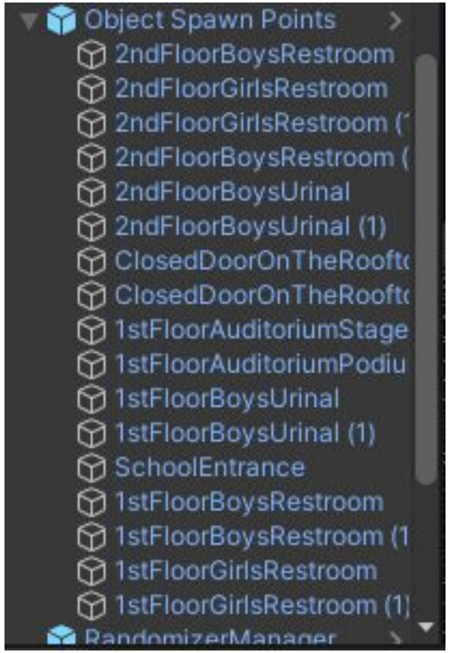
Smoke effect will show when wrong object was given to Rin



```
if (collidedGO.GetComponent<RandomObjectSc>() != null)
{
    //gets the two components of the right obj and the delivered obj
    RandomObjectSc sent = collidedGO.GetComponent<RandomObjectSc>();
    RandomObjectSc chosen = chosenGO.GetComponent<RandomObjectSc>();
    //now checks with the condition of location and name if the item delivered is right
    if (chosen.location == sent.location && chosen.objName == sent.objName)
    {
        //player wins
        Debug.Log("Arigato senpai!");
        VfxManager.Instance.instantiateEffect
            ("Heart", this.transform.parent.gameObject.transform.position, 1.0f);
        winCanvas = Object.Instantiate(winCanvas);
        Destroy(winCanvas, 3f);
    }
}
else
{
    Debug.Log("Wrong OBJECT!");
    VfxManager.Instance.instantiateEffect
        ("DustSmoke_A", this.transform.parent.gameObject.transform.position, 1.0f);
    loseCanvas = Object.Instantiate(loseCanvas);
    Destroy(loseCanvas, 3f);
}
```

Features with Event Broadcasting

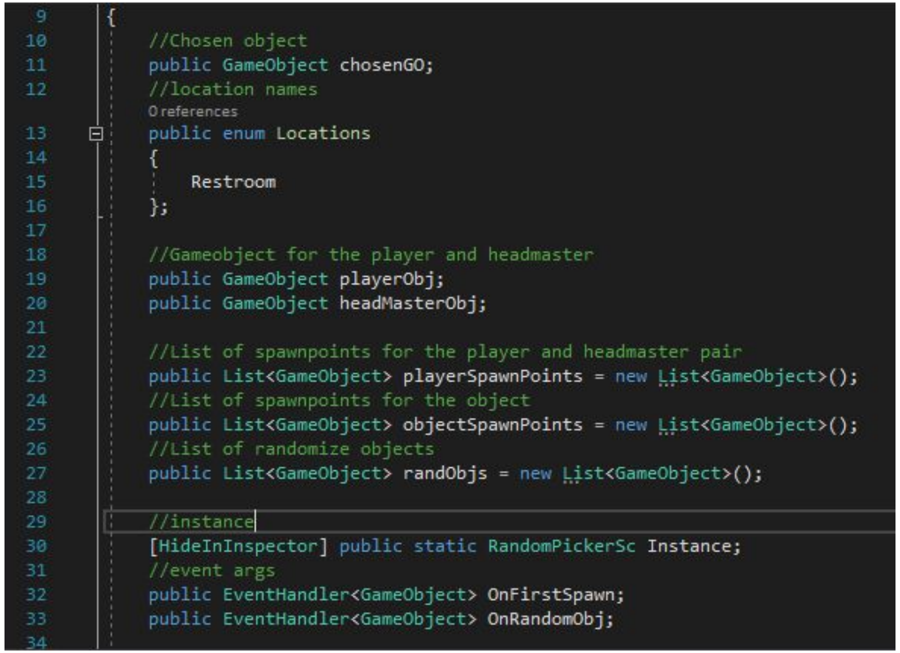
Random Object Generation and Placement



Object Spawn Points
- 2ndFloorBoysRestroom
- 2ndFloorGirlsRestroom
- 2ndFloorGirlsRestroom (
- 2ndFloorBoysRestroom (
- 2ndFloorBoysUrinal
- 2ndFloorBoysUrinal (1)
- ClosedDoorOnTheRooft
- ClosedDoorOnTheRooft
- 1stFloorAuditoriumStage
- 1stFloorAuditoriumPodiu
- 1stFloorBoysUrinal
- 1stFloorBoysUrinal (1)
- SchoolEntrance
- 1stFloorBoysRestroom
- 1stFloorBoysRestroom (1
- 1stFloorGirlsRestroom
- 1stFloorGirlsRestroom (1)

RandomizerManager

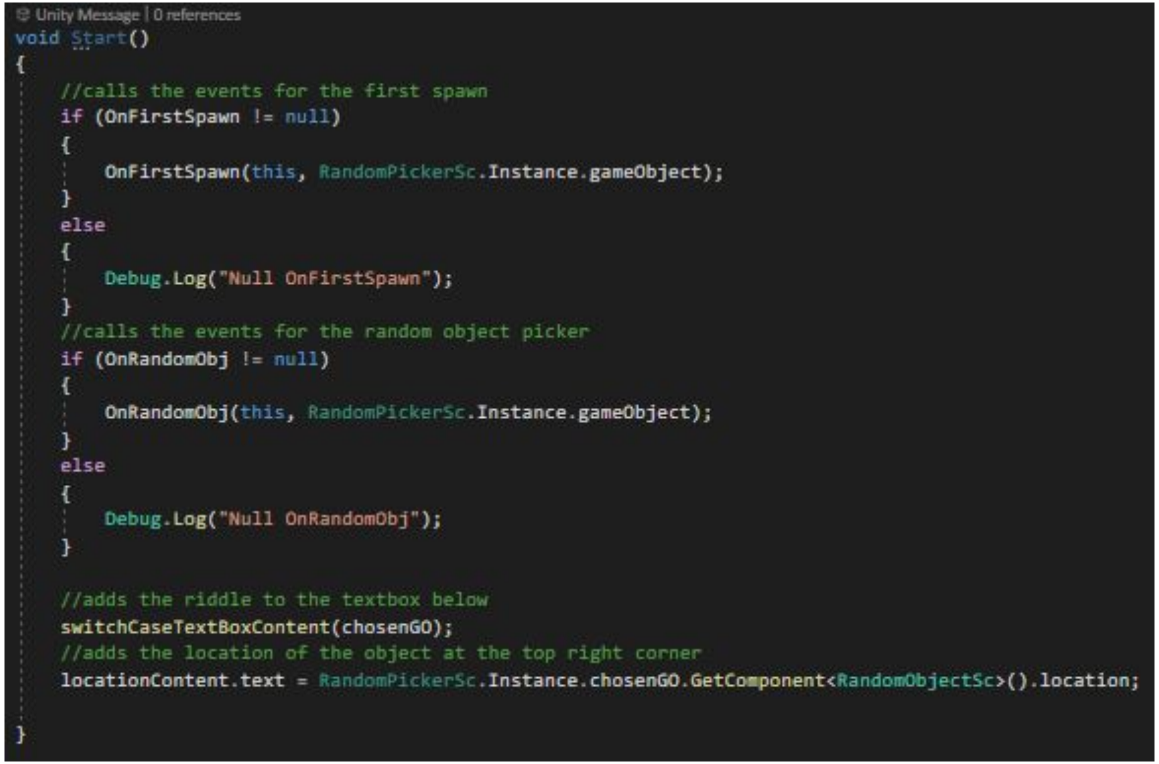
As seen in the hierarchy, this is a prefab of all the object spawn points represented as empty game objects



```
{
    //Chosen object
    public GameObject chosenGO;
    //location names
    0 references
    public enum Locations
    {
        Restroom
    };

    //Gameobject for the player and headmaster
    public GameObject playerObj;
    public GameObject headMasterObj;

    //List of spawnpoints for the player and headmaster pair
    public List<GameObject> playerSpawnPoints = new List<GameObject>();
    //List of spawnpoints for the object
    public List<GameObject> objectSpawnPoints = new List<GameObject>();
    //List of randomize objects
    public List<GameObject> randObjs = new List<GameObject>();

    //instance
    [HideInInspector] public static RandomPickerSc Instance;
    //event args
    public EventHandler<GameObject> OnFirstSpawn;
    public EventHandler<GameObject> OnRandomObj;

```

Features with Event Broadcasting

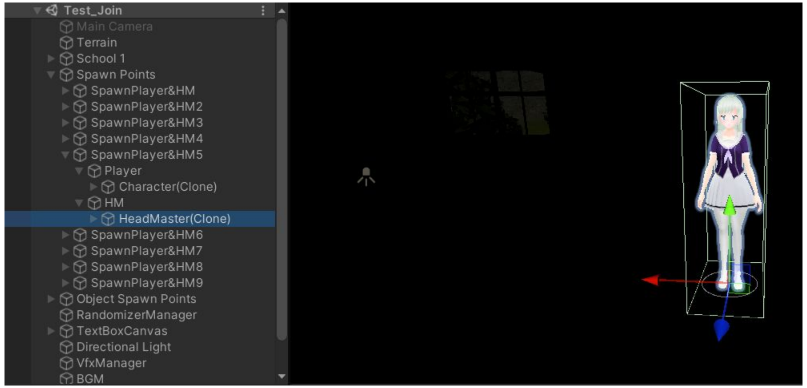
Random Object Generation and Placement



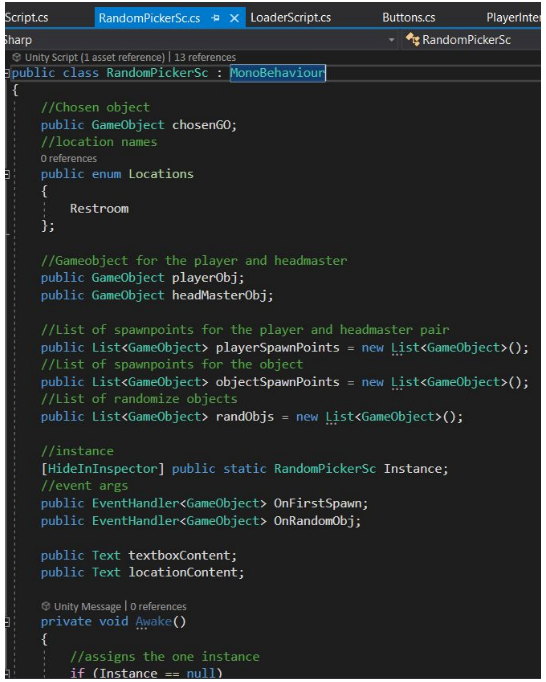
```
Unity Message | 0 references
void Start()
{
    //calls the events for the first spawn
    if (OnFirstSpawn != null)
    {
        OnFirstSpawn(this, RandomPickerSc.Instance.gameObject);
    }
    else
    {
        Debug.Log("Null OnFirstSpawn");
    }
    //calls the events for the random object picker
    if (OnRandomObj != null)
    {
        OnRandomObj(this, RandomPickerSc.Instance.gameObject);
    }
    else
    {
        Debug.Log("Null OnRandomObj");
    }

    //adds the riddle to the textbox below
    switchCaseTextBoxContent(chosenGO);
    //adds the location of the object at the top right corner
    locationContent.text = RandomPickerSc.Instance.chosenGO.GetComponent<RandomObjectSc>().location;

}
```

Features with Event Broadcasting

Random Placement of Player and Headmaster prefabs



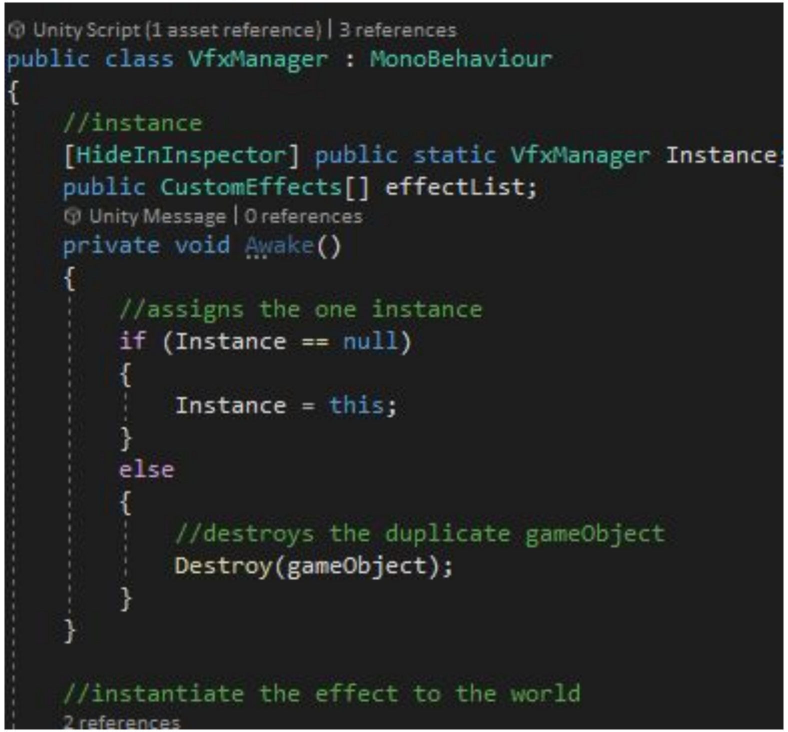
Transform of the possible spawnPoints for the player and headmaster



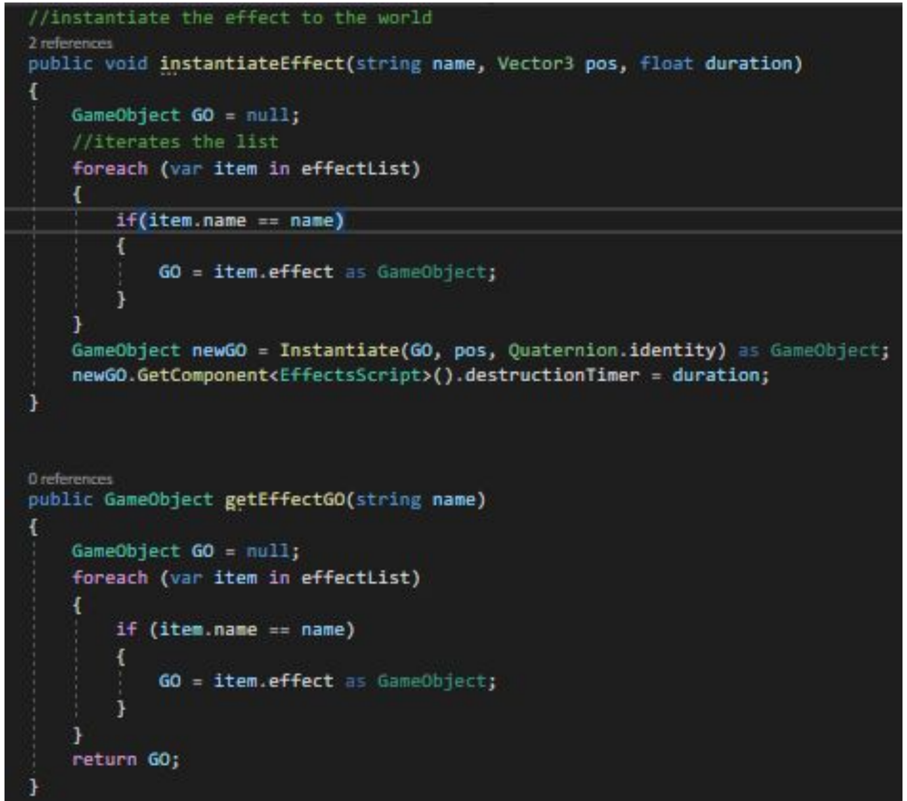
```
Unity Script (1 asset reference) | 13 references
public class RandomPickerSc : MonoBehaviour
{
    //Chosen object
    public GameObject chosenGO;
    //location names
    0 references
    public enum Locations
    {
        Restroom
    };

    //Gameobject for the player and headmaster
    public GameObject playerObj;
    public GameObject headMasterObj;

    //List of spawnpoints for the player and headmaster pair
    public List<GameObject> playerSpawnPoints = new List<GameObject>();
    //List of spawnpoints for the object
    public List<GameObject> objectSpawnPoints = new List<GameObject>();
    //List of randomize objects
    public List<GameObject> randObjs = new List<GameObject>();

    //instance
    [HideInInspector] public static RandomPickerSc Instance;
    //event args
    public EventHandler<GameObject> OnFirstSpawn;
    public EventHandler<GameObject> OnRandomObj;

    public Text textboxContent;
    public Text locationContent;

    Unity Message | 0 references
    private void Awake()
    {
        //assigns the one instance
        if (Instance == null)
```

Features with Event Broadcasting

SFX and VFX



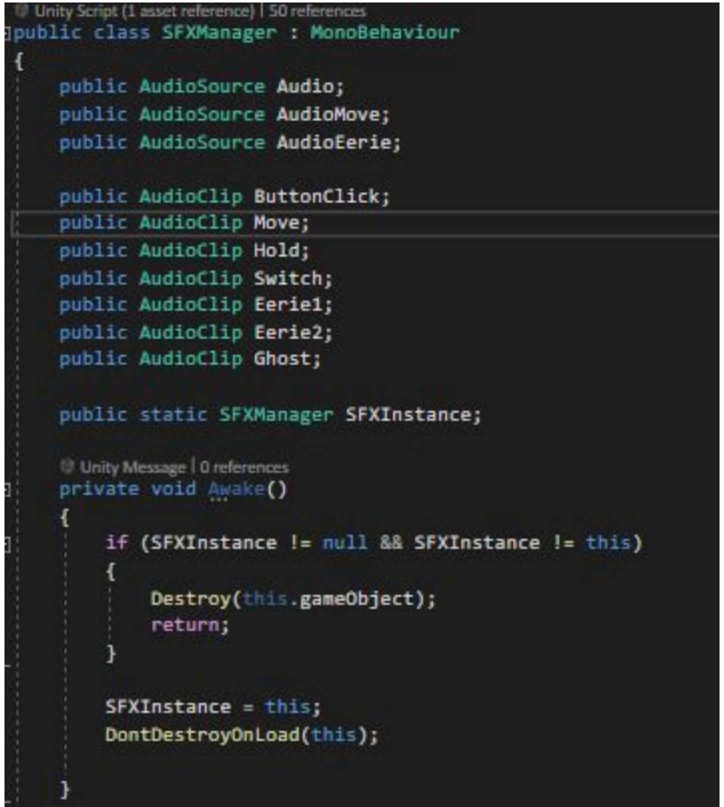
```
Unity Script (1 asset reference) | 3 references
public class VfxManager : MonoBehaviour
{
    //instance
    [HideInInspector] public static VfxManager Instance
    public CustomEffects[] effectList;
    Unity Message | 0 references
    private void Awake()
    {
        //assigns the one instance
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            //destroys the duplicate gameObject
            Destroy(gameObject);
        }
    }

    //instantiate the effect to the world
    2 references
```



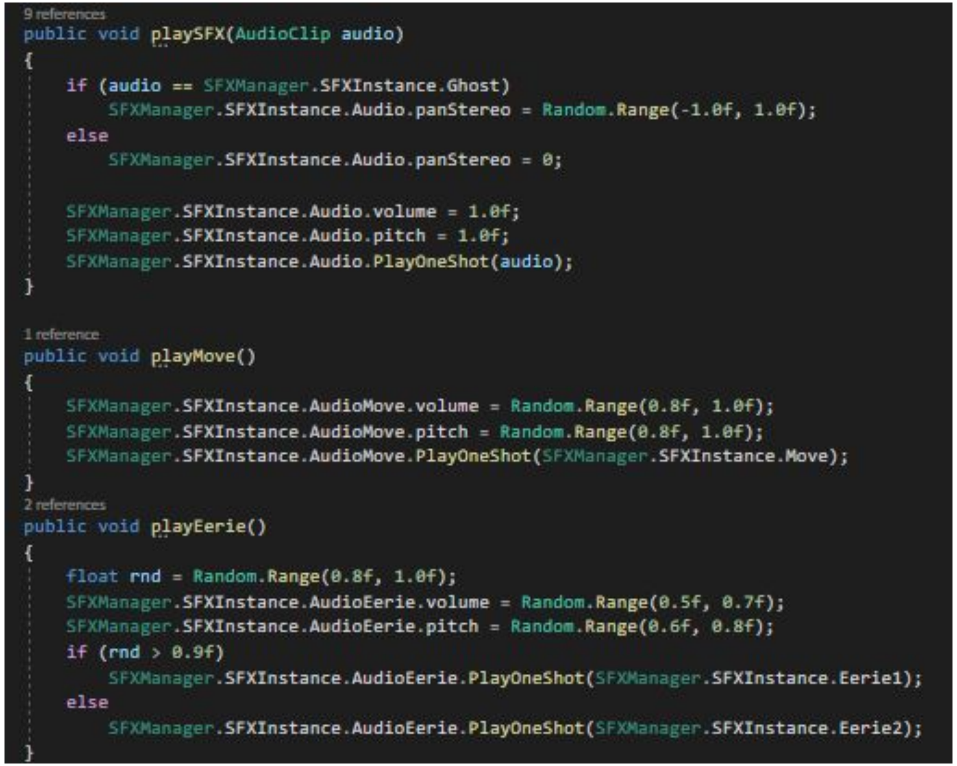
```
//instantiate the effect to the world
2 references
public void instantiateEffect(string name, Vector3 pos, float duration)
{
    GameObject GO = null;
    //iterates the list
    foreach (var item in effectList)
    {
        if(item.name == name)
        {
            GO = item.effect as GameObject;
        }
    }
    GameObject newGO = Instantiate(GO, pos, Quaternion.identity) as GameObject;
    newGO.GetComponent<EffectsScript>().destructionTimer = duration;
}


0 references
public GameObject getEffectGO(string name)
{
    GameObject GO = null;
    foreach (var item in effectList)
    {
        if (item.name == name)
        {
            GO = item.effect as GameObject;
        }
    }
    return GO;
}
```

Features with Event Broadcasting

SFX and VFX



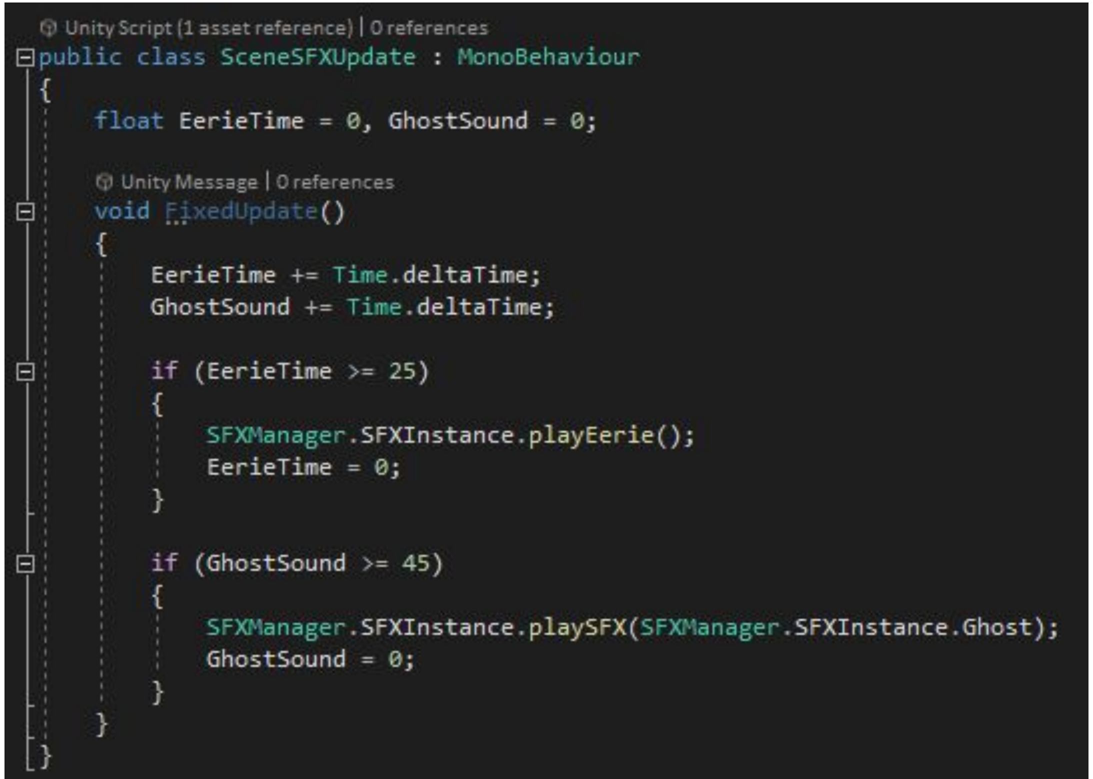
```
Unity Script (1 asset reference) | 50 references
public class SFXManager : MonoBehaviour
{
    public AudioSource Audio;
    public AudioSource AudioMove;
    public AudioSource AudioEerie;

    public AudioClip ButtonClick;
    public AudioClip Move;
    public AudioClip Hold;
    public AudioClip Switch;
    public AudioClip Eerie1;
    public AudioClip Eerie2;
    public AudioClip Ghost;

    public static SFXManager SFXInstance;

    Unity Message | 0 references
    private void Awake()
    {
        if (SFXInstance != null && SFXInstance != this)
        {
            Destroy(this.gameObject);
            return;
        }

        SFXInstance = this;
        DontDestroyOnLoad(this);

    }
```



```
9 references
public void playSFX(AudioClip audio)
{
    if (audio == SFXManager.SFXInstance.Ghost)
        SFXManager.SFXInstance.Audio.panStereo = Random.Range(-1.0f, 1.0f);
    else
        SFXManager.SFXInstance.Audio.panStereo = 0;

    SFXManager.SFXInstance.Audio.volume = 1.0f;
    SFXManager.SFXInstance.Audio.pitch = 1.0f;
    SFXManager.SFXInstance.Audio.PlayOneShot(audio);
}

1 reference
public void playMove()
{
    SFXManager.SFXInstance.AudioMove.volume = Random.Range(0.8f, 1.0f);
    SFXManager.SFXInstance.AudioMove.pitch = Random.Range(0.8f, 1.0f);
    SFXManager.SFXInstance.AudioMove.PlayOneShot(SFXManager.SFXInstance.Move);
}
2 references
public void playEerie()
{
    float rnd = Random.Range(0.8f, 1.0f);
    SFXManager.SFXInstance.AudioEerie.volume = Random.Range(0.5f, 0.7f);
    SFXManager.SFXInstance.AudioEerie.pitch = Random.Range(0.6f, 0.8f);
    if (rnd > 0.9f)
        SFXManager.SFXInstance.AudioEerie.PlayOneShot(SFXManager.SFXInstance.Eerie1);
    else
        SFXManager.SFXInstance.AudioEerie.PlayOneShot(SFXManager.SFXInstance.Eerie2);
}
```

Features with Event Broadcasting

SFX and VFX



```
Unity Script (1 asset reference) | 0 references
public class SceneSFXUpdate : MonoBehaviour
{
    float EerieTime = 0, GhostSound = 0;

    Unity Message | 0 references
    void FixedUpdate()
    {
        EerieTime += Time.deltaTime;
        GhostSound += Time.deltaTime;

        if (EerieTime >= 25)
        {
            SFXManager.SFXInstance.playEerie();
            EerieTime = 0;
        }

        if (GhostSound >= 45)
        {
            SFXManager.SFXInstance.playSFX(SFXManager.SFXInstance.Ghost);
            GhostSound = 0;
        }
    }
}
```

Contribution

-EMERSON PAUL P CELESTIAL

- Level Design, Scripting

-ALDREY D GAURANA

- Level Design, Scripting

-ERYN GABRIEL C. TALLADOR

- SFX and Environment Design , Scripting

-JOSEPH CHRISTOPHER C. SANTOS

- UI Design, Scripting