CE318 - High Level Games Development

Student ID - 2104672

Name - Brandon Eastwell

Enigma: Haunting Labyrinth

Game Description Document for Project Enigma

Overview

“Enigma: Haunting Labyrinth” is a first person dungeon escape horror game that plunges the player into an atmospheric, nightmarish experience. Each step you take time ticks down, you explore a dark labyrinth where you will need to manage your resources. Limited flashlight batteries and camera shots are the only source of visibility.

How to play

The objective of the game is to find the exit of the enigmatic dungeon, avoiding the weeping angel-like entities at all costs while balancing resources.

Once transitioning to the next playable scene after the introductory level, ensure you follow the candles and pick up the flashlight and survival kit, you will need them!

Key bindings

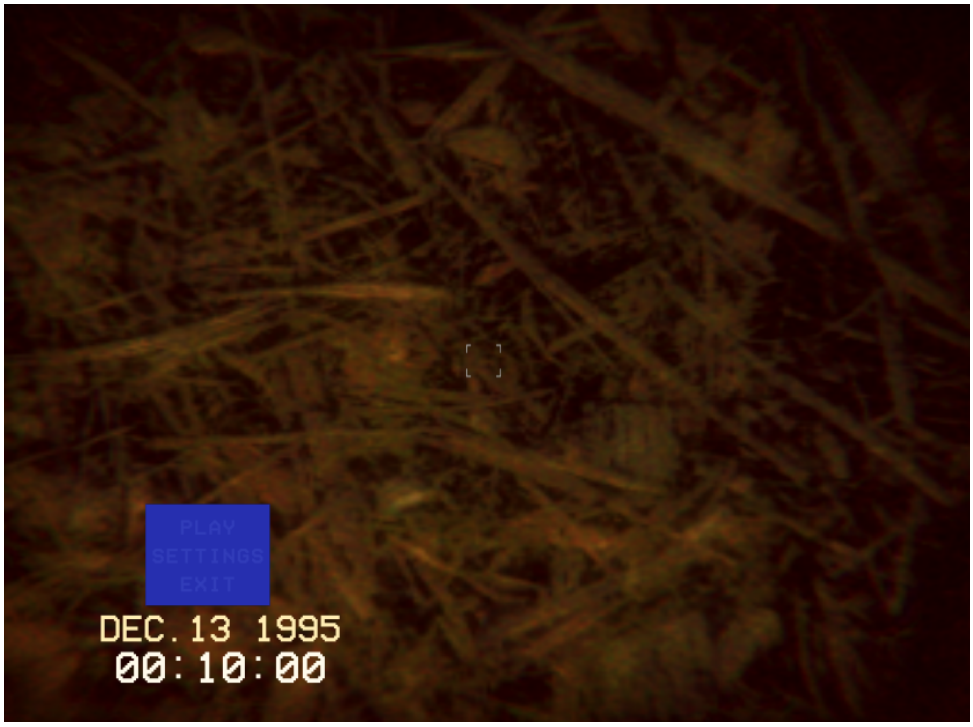
- WASD - movement
- Q - use health kit
- R - reload flashlight
- F - turn on/off flashlight
- SHIFT - sprint
- LMB - camera flash

Game Structure

Splash screen starting menu

The splash screen starting menu has an animated background using a glitch effect with dreamy background music to set the tone of the game.

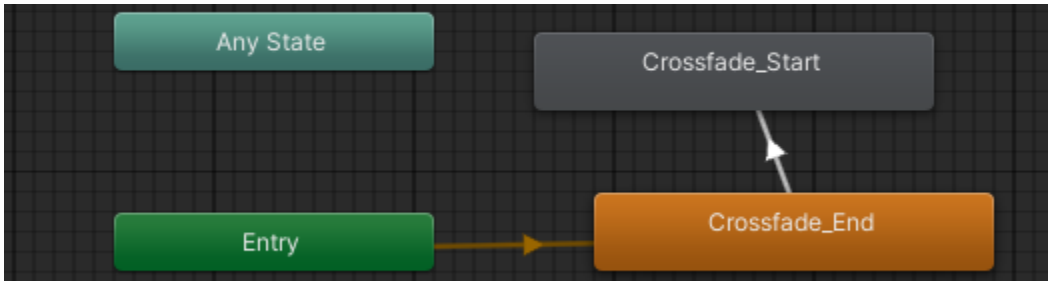
Further options menu that has a working adjustable volume slider and adjusts all every sound in the game at once by working as a master. Mode selector allows selection of an alternative difficulty which on easy mode, the time you are given to escape is 10 minutes whereas hard only gives you 5 minutes.



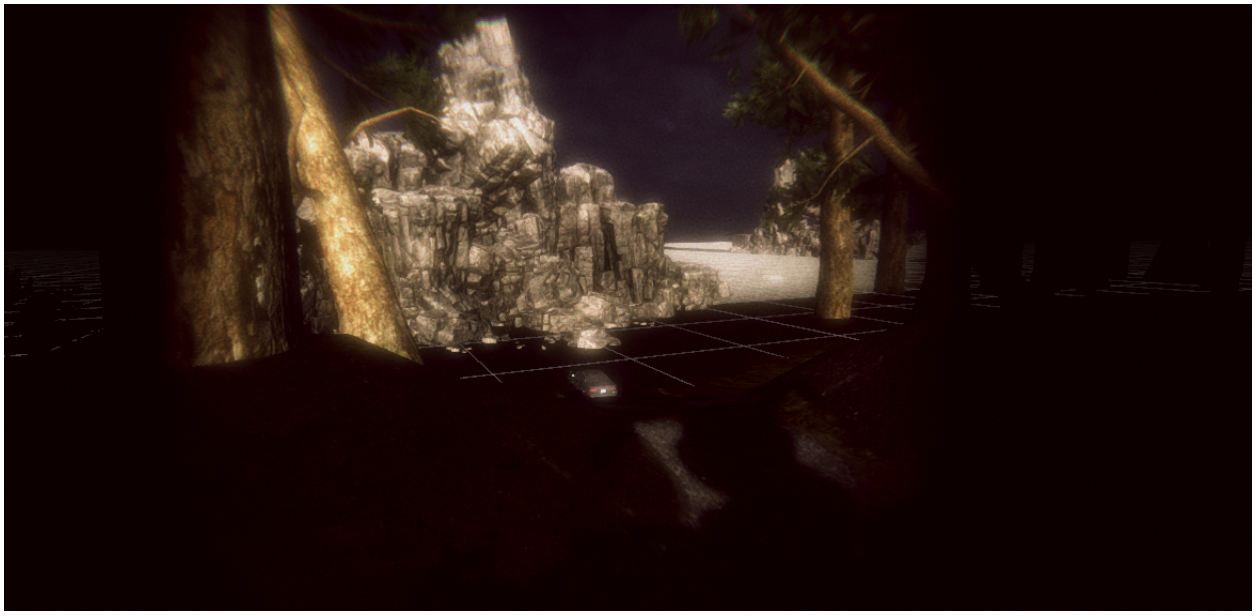
This screen shows the in-game pause menu, this is a menu that is not in a working state where the menu can not be exited out of once entered therefore menu options are not able to be selected even though it is the same build menu from the splash screen.

Due to the nature of the game, my creative decision of a timer system that will only tick down if the player is moving creates a sense of tension and urgency. Therefore I felt that a saveable game state would defeat the purpose of the level of immersion I had attempted to achieve with this game as I wanted to create a short lived experience that cannot be backed out, or in another sense enables the player to feel a level of ease due to the idea of a saveable game state at any point of the game.

Each transition between scenes feature a fade out, fade in transition as a loading screen.



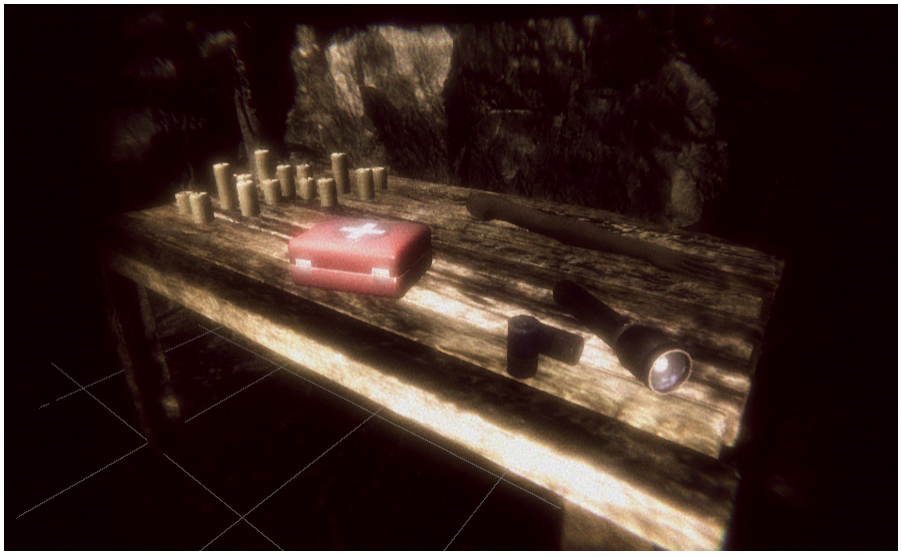
Enigma features 2 playable scenes where the introductory scene features a forest terrain as a background with rock formations. The purpose of this scene is to build a narrative of a curious explorer seeking the unknown, the sound design and visual look of this scene sets the serious yet fantastical tone of the game.



The introductory scene shows dirt tracks leading up to the vehicle that can be seen from the player perspective, giving the impression the protagonist has just driven to the location, leaving the car running as if the protagonist believes of no threat and will be back to the car shortly. If I had more time to develop the environment of this scene, I would have finished the cutscene that had to be dropped due to time constraints, as a pre cinematic for the opening to this scene where it shows a wide angle shot of the sedan driving down the dirt path with additional sound effects for exiting the vehicle. I would have also liked to place additional environmental art such as denser trees and bushes. Although I found it difficult searching for assets for this particular reason as I wanted a specific type of tree that had a far lower height and featured bushes and branches at a player level to enhance density.

Leading into the second scene, the transition begins within the cave as you explore deeper, a mannequin is found at the start of the cave to lead the player through it, reaching a trigger deeper into the cave that starts the next scene.

Enigma features a second scene called the dungeon, where the player traverses through the other end of the cave in this scene and is led to a table of survival items such as a flashlight, health kit and batteries.



Within the dungeon scene, there can be survival items scattered across the labyrinth, particularly the first set of items encountered in the starting room of the scene where the first set of items can be found, especially the flashlight which is a key component of the gameplay.

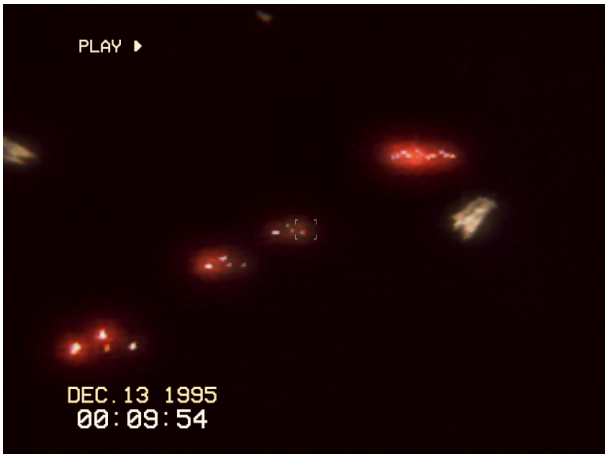
As part of a tutorial system for the game, instead of a dedicated tutorial level that would break the immersion and take the player out of the narrative, I wanted to achieve a design where forced interactions take place as part of a way to ease the player into learning the key elements of the game. By emerging the player into risk free experiences they remain immersed and gain a level of satisfaction by discovering the rules of the game in a natural way where it does not feel like the game is trying too hard to teach.

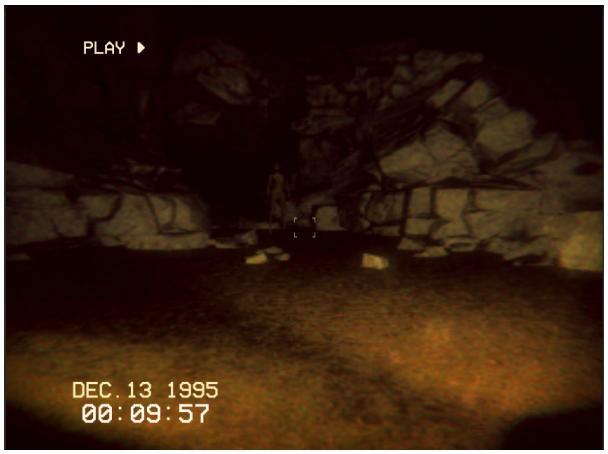
I do this by causing situations where the player would need to use a particular resource they had gathered in order to survive, a forced encounter at the very entrance into the labyrinth hurts the player and prompts the player to heal with the corresponding key press in a fluent way.


```
0 Unity Message | 0 references
void FixedUpdate()
{
    Ray ray = mainCamera.ScreenPointToRay(Input.mousePosition);
    RaycastHit hit;
    if (Physics.Raycast(ray.origin, mainCamera.transform.forward, out hit, Mathf.Infinity, layerMask)) //shoot
    {
        Debug.DrawLine(ray.origin, hit.point, Color.yellow);
        GameObject hitObj = hit.collider.gameObject;
        Debug.Log("distance from object " + hit.distance);
        if (hit.distance < 3)
        {
            Vector3 pos = getScreenSpaceTransform(hitObj.transform.position);
            ObjectText.text = hitObj.tag;
            if (firstTime)
            {
                UI.pointersText.text = "E to pickup";
            }
            ObjectText.rectTransform.anchoredPosition = pos;
            if (Input.GetKey(KeyCode.E)) {
                if (hitObj.tag == "Flashlight" && !flashlight.flashlightPickedUp)
                {
                    flashlight.flashlightPickedUp = true;
                }
                if (hitObj.tag == "Battery")
                {
                    inventory.addBattery();
                }
                if (hitObj.tag == "Firstaid")
                {
                    inventory.addHealthKit();
                }
                if (hitObj.tag == "Pill")
                {
                    inventory.addFilm();
                }
                firstTime = false;
                PickupSound.Play();
                UI.pointersText.text = "";
                hitObj.gameObject.SetActive(false);
            }
        }

    } else
    {
        Debug.DrawLine(ray.origin, hit.point, Color.yellow);
        ObjectText.text = "";
    }
}
```

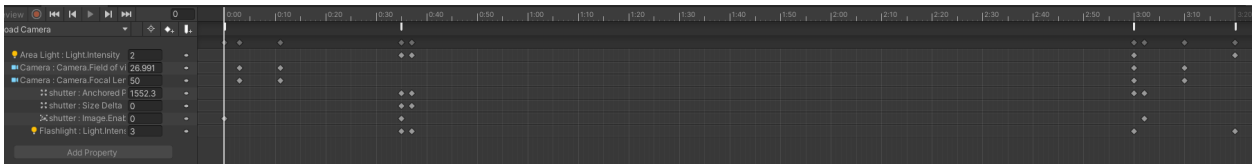
This display shows another technical feat that aids in the intuitivity and teaching of the player, as shown below, I have created a raycasting interaction system that shoots rays from the center of the screen and hits objects on the interactables layer mask. I then take the screen space location of the point where the ray hits and apply an offset to the tooltip indicator that tells the player what the object is, and that it is able to be picked up. An information indicator text area fixed to the bottom right corner tells the user what the keybinds are depending on whether or not it is their first time encountering that system.





Enigma features many sources of light, as a atmospheric horror, light is one of the most powerful elements for creating an atmosphere, and it is done here with minimal but resourceful lighting, lighting has become a resource you need to balance as not only does lighting offer visibility and environmental purpose, it is also envisioned to be a mechanic for the AI, where AI are disrupted by light and will only move in the darkness. In the displays above I show dynamic lighting sources like the players flashlight. The most interesting source of lighting is the camera shutter effect that produces a very quick glimpse of light from the cameras flash which may be the only source of lighting with a flashlight on low or no battery, this short burst of light in combination with unsettling movement of mannequins where they may have moved between camera flashes but is only seen for a flash moment.

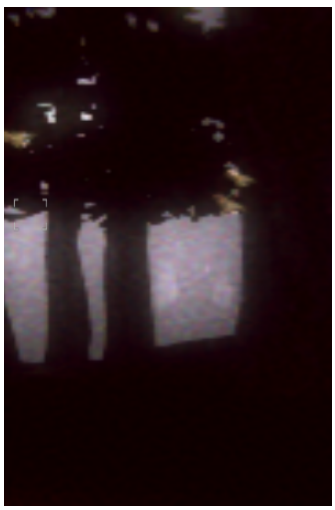
Audio is another element of a game I focused a lot of development time towards, to compliment the eerie tone and the satisfaction of the camera shutter mechanic, analog vintage camera sounds remind the player of the time period, as well as a sense of isolation with the majority being quick sharp sounds to break up the eerie drone of the dungeon and rebounding throughout the dungeon walls.



The animation timeline shows the camera reload animation for which is used for both reloading the camera with film and applying health kits, the use of sound such as the brushing of hands across the microphone, camera flick off switch and bleep alongside lighting keyframes has helped produce an animation that feels appropriate for the game world. Further extensive sounds are included for shutter effects, picking up of items, background ambience, reloading the flashlight with batteries and turning the flashlight on or off.

The use of multiple cameras was being implemented while I was developing the cutscene opening for the introductory scene, however due to incompletion I was not able to feature it into the game.

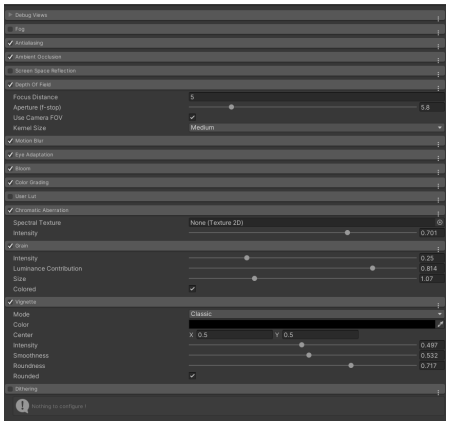
Enigma showcases uses of particle effects that follow the player and acts as a falling dust effect that enhances the look and feel of the game. I had also used particle emitters and a flame material for the creation of fire emitters that are used for the fire torch light sources.

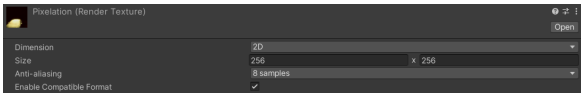


Enigma demonstrates an extensive use of shader effects, post processing and glitch effects to construct the visual style of the retro PS1 era, achieving a look and feel that gives you a slight uncanny feeling, an

essence of nostalgia for classic technology and the vintage styled camera interfaces and lenses that had a more analog feeling experience as opposed to digital. Post processing was used to achieve the vintage, retro-like color palette of deeper oranges that relate to the characteristics of color processing on analog film cameras of the 1990s.

To achieve the intentionally pixelated, PS1 styled graphic effects I had used a 2D pixelation filter that is simply a low resolution render texture applied to the camera. Alongside other post processing effects such as motion blur, eye adaptation (acts as auto focusing on a camera), color grading, vignetting, chromatic aberration, noise grain and bloom.





To achieve the level of detail in the rock formations and across the dungeon scene, I had used asset packs from the unity store such as Rocks and Boulders and Dungeon Lite for high quality materials and models to construct the labyrinth. To create clusters of rocks I applied rigidbody components to rock meshes and dropped them from a height to create a natural cluster of rock piles across the ground. This very randomisation enabled me to build varied looking environments that look carefully placed extremely quickly.

```
Unity Message | 0 references
void Update()
{
    //Look at the player
    if (IsInView(player, gameObject)) {
        seen = true;
    }
    if (seen && !IsInView(player, gameObject)) {
        agent.SetDestination(player.transform.position);
        if (start)
        {
            StartCoroutine(playRunningFootsteps());
        }
        inMotion = true;
    } else
    {
        inMotion = false;
        agent.SetDestination(transform.position);
    }


    if (Vector3.Distance(transform.position, player.transform.position) < enemyDistance)
    {
        gameObject.GetComponent<NavMeshAgent>().velocity = Vector3.zero;
        player.GetComponent<PlayerManager>().health -= 20;
        gameObject.SetActive(false);
    }
}
1 reference
IEnumerator playRunningFootsteps()
{
    //play sound
    start = false;
    footsteps.Play();
    yield return new WaitForSeconds(1f);
    start = true;
}
0 references
void MoveTowardsPlayer()
{
    if (player == null)
        return;

    transform.position = player.transform.position - player.transform.forward;
    Vector3 lookPos = player.transform.position - transform.position;
    lookPos.y = 0;
    transform.rotation = Quaternion.LookRotation(lookPos);
}

2 references
bool IsInView(GameObject origin, GameObject toCheck)
{
    Vector3 pointOnScreen = mainCam.WorldToScreenPoint(toCheck.GetComponent<Renderer>().bounds.center);

    //Is in front
    if (pointOnScreen.z < 0)
    {
        Debug.Log("Behind: " + toCheck.name);
        return false;
    }

    //Is in FOV
    if ((pointOnScreen.x < 0) || (pointOnScreen.x > Screen.width) ||
            (pointOnScreen.y < 0) || (pointOnScreen.y > Screen.height))
    {
        Debug.Log("OutOfBounds: " + toCheck.name);
        return false;
    }

    RaycastHit hit;
    Vector3 heading = toCheck.transform.position - origin.transform.position;
    Vector3 direction = heading.normalized;// / heading.magnitude;

    if (Physics.Linecast(mainCam.transform.position, toCheck.GetComponent<Renderer>().bounds.center, out hit))
    {
        if (hit.transform.name != toCheck.name)
        {
            /* -->
            Debug.DrawLine(cam.transform.position, toCheck.GetComponentInChildren<Renderer>().bounds.center, Color.red);
            Debug.LogError(toCheck.name + " occluded by " + hit.transform.name);
            */
            Debug.Log(toCheck.name + " occluded by " + hit.transform.name);
            return false;
        }
    }
    return true;
}
```

In the code snippet above, I show my AI Manager for the mannequins enemy where the behavior is that they do not do anything until they are seen, or rendered by the camera in which they have technically been seen. I then shoot rays from the origin (player) to the AI to detect objects disrupting the path between them. If there is a clear line of sight, the AI is set to seen, which in the update method if the AI is seen and (using the same look detection method) the camera is not rendering the enemy, the enemy will move towards the player. However this behavior is currently not working as intended and could not figure out the issue with the in view detection as even when there is a clear line of sight, each ray is occluded by some object and therefore is never returning true.

As mentioned in my interaction system section, I feel I have contributed to a tutorial level through intuitive level design that slowly introduces new players to mechanics as the player progresses through the level in order to maintain immersion. I have made a conscious effort to ensure the level is intuitive, and the game mechanics are introduced in a way that they are straightforward to pick up by building my interaction system around making it easier for new players to learn.

Asset Reference List

[Unity Asset Store Pack by IL.ranch]

https://assetstore.unity.com/packages/3d/vegetation/trees/dream-forest-tree-105297

[Unity Asset Store Pack by Unity Technologies]

https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808

[Sedan car by art georg]

https://assetstore.unity.com/packages/3d/sedan-car-6751

[Mannequins by VIS games]

https://assetstore.unity.com/packages/3d/props/mannequins-56448

[Rocks and Boulders]

[Survival Items]