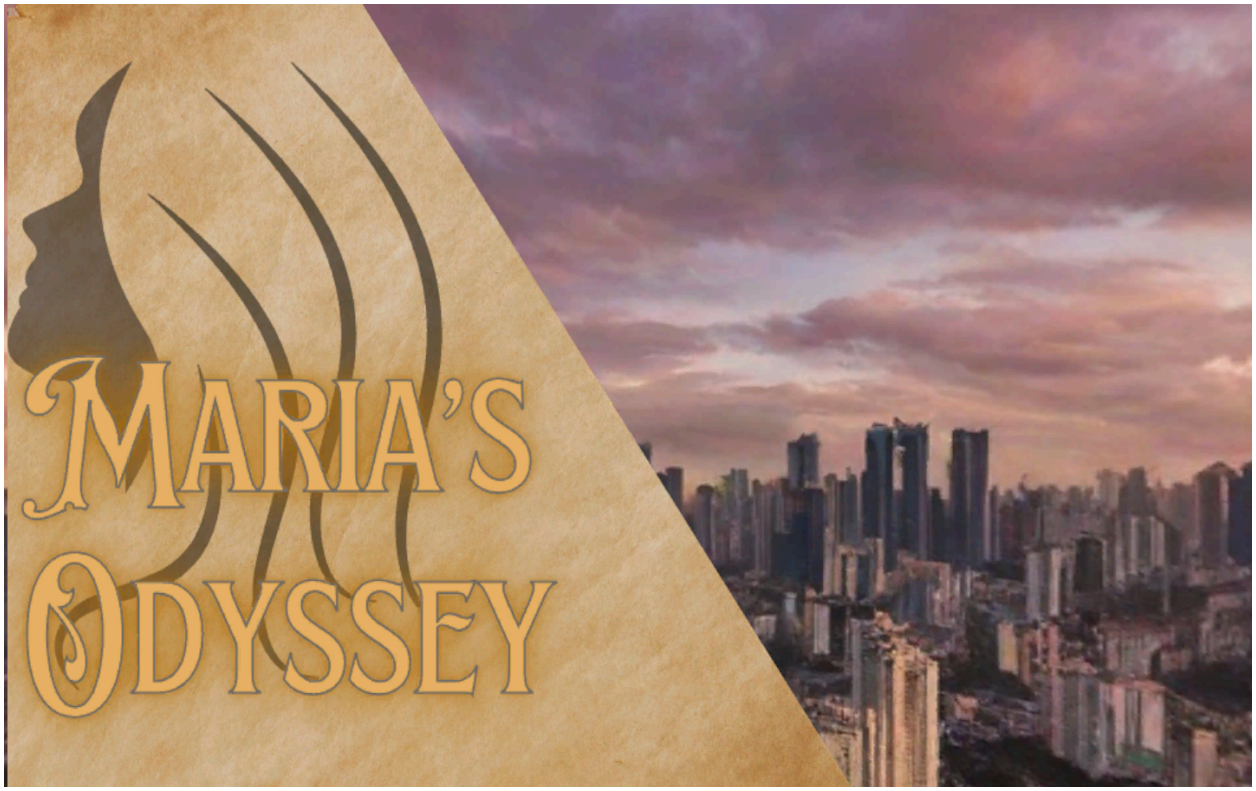
Portfolio Breakdown

Gameplay Demo videos:

https://drive.google.com/drive/folders/1OjthNafSYIsa01ZX3B32komYnaIbk6pO?usp=drive_link

Thesis Project (Maria's Odyssey)

Maria's odyssey is a first-person educational adventure travel game utilizing player speech mechanics to create an engaging, educational and immersive adventure into 3 Philippine dialects (Tagalog, Cebuano & Hiligaynon) and regions (Manila, Cebu & Bacolod)

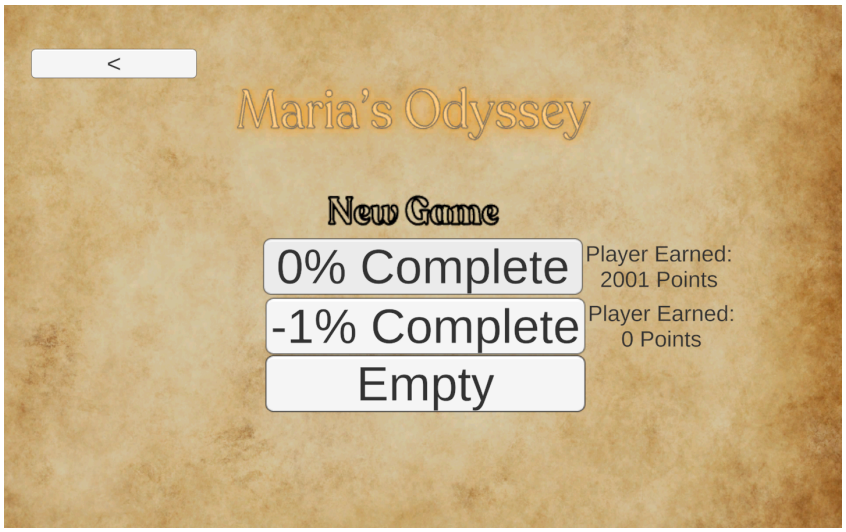


Opening Scene

After pressing the Opening screen, the Main Menu allows 3 options switched through 3 different screens utilizing onbuttonpress to switch between screens. The 3 different screens have their own buttons in order to return the main hub, allowing for proper navigation of options.

```
public class IntroPage1 : MonoBehaviour
{
    public CanvasGroup cGroup;
    public SaveSlotsMenu SaveSlotsMenu;

    public void DownButtonClicked()
    {
        if (Server.instance.isTransitioning)
            return;

        StartCoroutine(TransitionToNewPanel());
        Debug.Log("Working:");
    }

    IEnumerator TransitionToNewPanel()
    {

        Server.instance.isTransitioning = true;
        Sequence _mySequence = DOTween.Sequence()
        .Append(cGroup.DOFade(0f, 1f).SetEase(Ease.Linear))
        .AppendCallback(delegate()
        {
            Server.instance.m_Panels[1].GetComponent<IntroPage1>().cGroup.alpha = 0f;
            Server.instance.m_Panels[1].SetActive(true);
            Server.instance.m_Panels[2].GetComponent<IntroPage1>().cGroup.alpha = 0f;

                    })
        .Append(Server.instance.m_Panels[1].GetComponent<IntroPage1>().cGroup.DOFade(1f, 0f).SetEase(Ease.Linear));

        yield return _mySequence.WaitForCompletion();
        Server.instance.isTransitioning = false;
        StartCoroutine(TransitionToOldPanel());


    }
```

This code snippet presents a code version of the code utilized for menu exploration, turning different game Objects off and changing the opacity of others for better visuals of the game



New Game Script utilizes save slots with multiple variables unique to each profile that will allow the player to enter and override the current save file, meanwhile the Load Game screen will allow the player to load different variables as is, with the added part of being able to spawn at the location last loaded in.

Opening Tutorial Scene after Newgame button is pressed teaches the player the needed basic mechanics of the game, such as walking interaction with NPCs and objects, also serves as the beginning of the plot of the game

Talking with NPC and choosing dialogue option triggers the Speech Minigame mechanic that tasks players with speaking words on screen into their microphone if the pronunciation of the word is acceptable, the system will award the player with 100

points seen in the top right of the screen and load the next word in the sequence. If there are no more words within the sequence, the system will then exit the player from the minigame and add the awarded points into the player's permanent Point counter. Other mechanics within this minigame are the timer inwhich when it runs out reveals a button that will allow the player to skip the word if they deem it too difficult, if they do choose this option they will not be awarded with the associated 100 point they would have gotten if they did choose to say the word. The X button on the top right exits the minigame and returns player to state before interaction with associated NPC, no points are given if this avenue is taken. The words on the left show the words that the system hears at that moment allowing the player to adjust what they are saying in order to align with the system. The words on the bottom show words that have been said already.

```
public class VoiceLoader : MonoBehaviour
{
    [SerializeField] private Whisper.Samples.VoiceMechanic voice;
    public TextAsset textAssetdata;
    public string version1,version2,version3;
    public bool isLong=false;
    public DictionarywordsScript DictionarywordsScript;

    public void Search(string nextword){
        DictionarywordsScript.NewAdd(nextword);
        string[] data = textAssetdata.text.Split(new string[] {",","\n"}, System.StringSplitOptions.None);
        for (int i = 0; i < data.Length; i++){

        if(nextword.ToLower()==data[i].ToLower()){

            voice.words.Add(data[i]);
            if (data[i].Length>5){
                version3 = data[i].Remove(data[i].Length-1);
                version3 = version3.Substring(1);
                voice.words.Add(version3);
            }
            else if (data[i].Length>3){
                isLong=true;
                version1 = data[i].Remove(data[i].Length-1);
                version2 = data[i].Substring(1);
                voice.words.Add(version1);
                voice.words.Add(version2);
            }

            voice.words.Add("");
            version1 = "";
            version2 = "";
            version3 = "";
            isLong =false;
        }
        }
    }
    }
```

Code that checks the word to be loaded into the voice mechanic, in which the word loaded will first be checked whether it is seen within the given Excel file, and after which it will create multiple versions of the word to make the listening part of the game easier

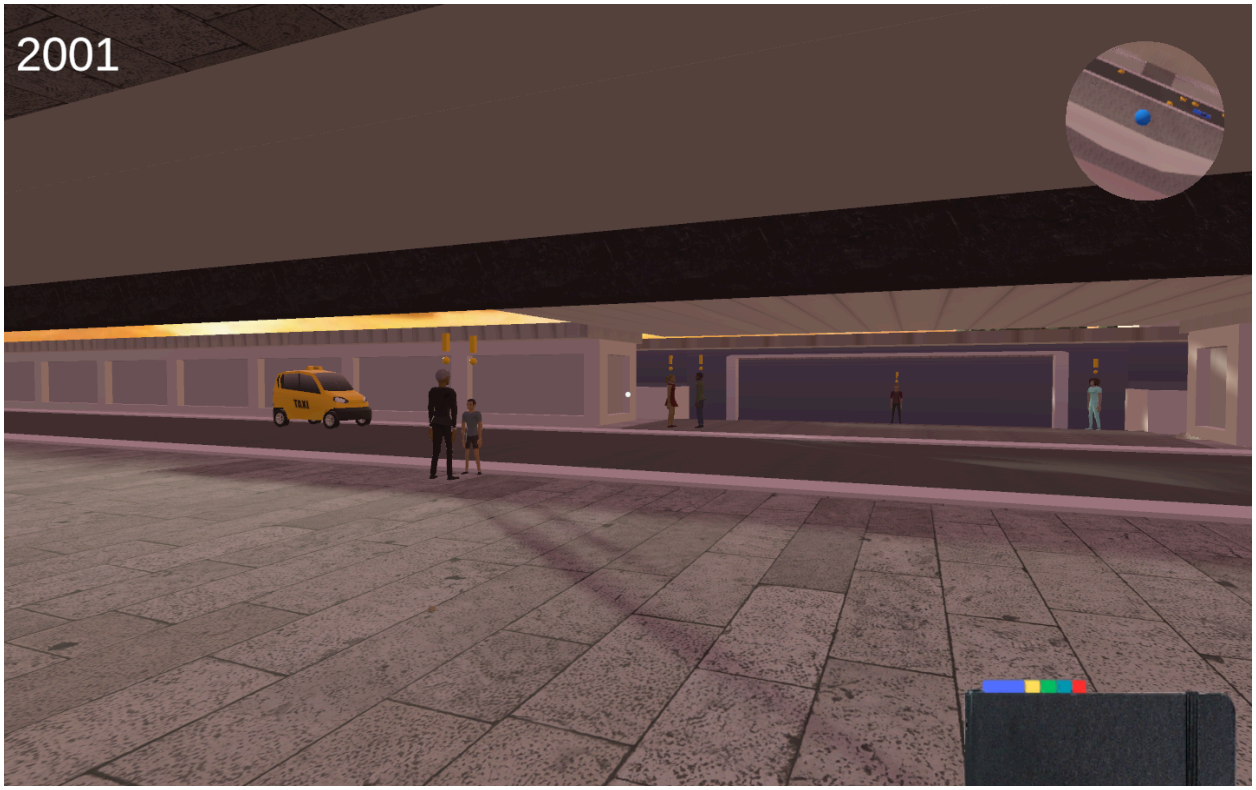
for the player. Length of words will change the amount of strings removed or amount of version the words that is used.

```
    public void VoiceBegin(){
        DialogueManager.instance.dialoguePanel.SetActive(false);
        pastwords.GetComponent<TMPro.TextMeshProUGUI>().text = "";
        playerwords.GetComponent<TMPro.TextMeshProUGUI>().text = "";
        pointsearned.GetComponent<TMPro.TextMeshProUGUI>().text = "+" + accumulatedAmount.ToString();
        Minigame.SetActive(true);
        screenwords.GetComponent<TMPro.TextMeshProUGUI>().text = words[0];
        actions.Add(words[0],Do);
        if(words[0].Length>5){
            actions.Add(words[1],Do);
        }
        else if(words[0].Length>3){
            actions.Add(words[1],Do);
            actions.Add(words[2],Do);
        }
        whisper.initialPrompt = words[0];
        _stream.StartStream();
        microphoneRecord.StartRecord();
        x = 0;
        y = 0;
            timeron=true;
            yes.Play();
        accumulatedAmount = 0;
        }

public void Update(){
            if(timeron==true){
        targetTime -= Time.deltaTime;
        Timer.fillAmount = targetTime /10;
        if (targetTime <= 0.0f)
    {
skipButton.SetActive(true);
yes.Stop();
    }
}
```

This area of the code is where the voice mechanic would begin, clearing needed variables and entering words given from the voice loader and putting them both on screen and into the mechanic that would listen for the required word.

```
92      private void OnSegmentUpdated(WhisperResult segment)
93          {
94              string nospaces = segment.Result.Replace(" ", string.Empty);
95              var sb = new StringBuilder();
96              foreach (char c in nospaces)
97              {
98              if (!char.IsPunctuation(c))
99                  sb.Append(c);
100             }
101             nospaces = sb.ToString();
102             found = nospaces.Contains(words[x], StringComparison.OrdinalIgnoreCase);
103             if(words[x].Length >5){
104                 found4 = nospaces.Contains(words[x+1], StringComparison.OrdinalIgnoreCase);
105             }
106             else if(words[x].Length >3){
107             found2 = nospaces.Contains(words[x+1], StringComparison.OrdinalIgnoreCase);
108             found3 = nospaces.Contains(words[x+2], StringComparison.OrdinalIgnoreCase);
109             }
110             Debug.Log(nospaces);
111             playerwords.GetComponent<TMPro.TextMeshProUGUI>().text = nospaces;
112             print($"Segment updated:{segment.Result}");
113             if(found){
114                 accumulatedAmount += 100;
115                             pointsearned.GetComponent<TMPro.TextMeshProUGUI>().text = "+" + accumulatedAmount.ToString();
116                             yes.PlayOneShot(no);
117                             yes.Play();
118                             found = false;
119                             actions[words[x]].Invoke();
120
121
122             }
123             else if(words[0].Length>5){
124             if(found4){
125                                 accumulatedAmount += 100;
126                             pointsearned.GetComponent<TMPro.TextMeshProUGUI>().text = "+" + accumulatedAmount.ToString();
127                             yes.PlayOneShot(no);
```

This code is the area of the code that cleans and turns the speech to text output into text that we can check for proper pronunciation, if it deems the string output close enough, this function will then invoke the function related to these words. The function related to these words is to load in the next word in the list, and to add 100+ points into the score box and in-game variables.


2001
TAXI


2001
CHICKEN STAR
2GO
2GO
Jollibee

NPCs you can talk to are exposed through the yellow exclamation point above their head and red circle on the map showing the location of intractable individuals.

When NPC is interacted with, the game will then take the text file within NPC game object and load it into the dialogue manager. Dialogue manager will then load INK json file, the player will then be given text on screen and will be able to proceed with text if they press on the screen. NPC will sometimes give dialogue options on screen that will load the Speech minigame onto screen. Skip button allows the player to skip any dialogue encounter, if the mission or speech minigame npc is skipped, the player will not be able to access the NPC dialogue afterwards.

```
64      public void EnterDialogueMode(TextAsset inkJSON){
65          Server.instance.MenuPause  = true;
66          Server.instance.RestrictPlayer();
67          currentStory = new Story(inkJSON.text);
68          makingDecision = false;
69          dialogueIsPlaying = true;
70          dialoguePanel.SetActive(true);
71          ContinueStory();
72      }
73      public void ExitDialogueMode(){
74          Server.instance.MenuPause  = false;
75          dialogueIsPlaying = false;
76          dialoguePanel.SetActive(false);
77          dialogueText.text = "";
78          Server.instance.StartPlayer();
79          choiceLoad = 5;
80          voice.skipped = false;
81          if(people != null){
82          if(people.hasNextActivity){
83          people.hasNextA();
84          }
85          if(people.SecondLink){
86              people.SecondL();
87          }
88          else
89          {
90              if(people.secondTimeSpeaking){
91                  people.secondTimeS();
92              }
93              if(people.hasQuest){
94                  people.hasQ();
95              }
96          }
97          if(people.onetimedialouge){
98              if(!voice.earlyExit){
99              people.allowed = false;
```

This Code Snippet presents the Enter Dialogue function that is run when an NPC is interacted with wherein the INK Json file is loaded into the dialogue manager and needed parameters are initialized and begun for gameplay. It also shows parameters run when exiting dialogue, clearing all needed text boxes and choice options for smooth

loading of processes.

```
107     public void ContinueStory(){
108         if(currentStory.canContinue){
109             dialogueText.text = currentStory.Continue();
110             DisplayChoices();
111         }
112         else{
113             ExitDialogueMode();
114         }
115     }
116     private void DisplayChoices(){
117         List<Choice> currentChoices = currentStory.currentChoices;
118         if (currentChoices.Count > choices.Length){
119             Debug.LogError("More Choices than able to be supported: " + currentChoices.Count);
120         }
121         int index = 0;
122         foreach(Choice choice in currentChoices){
123             choices[index].gameObject.SetActive(true);
124             choicesText[index].text = choice.text;
125             index++;
126             makingDecision = true;
127         }
128         for(int i = index; i < choices.Length; i++){
129             choices[i].gameObject.SetActive(false);
130         }
131     }
132     public void MakeChoice(int choiceIndex){
133         currentStory.ChooseChoiceIndex(choiceIndex);
134         choiceLoad = choiceIndex;
135         if(people.active){
136             people.loading();
137         }
138     }
139     private IEnumerator SelectFirstChoice(){
140         EventSystem.current.SetSelectedGameObject(null);
141         yield return new WaitForEndOfFrame();
142
```

This code snippet presents the continue story function that, when used, allows the player to continue to the next line of text within the INK json file, pushing dialogue forward. After that, we can see the Display options section that loads the given options of players in the needed text boxes and buttons. It also shows the function initialized when a choice is made, moving the player into the speech minigame, freezing all player actions till they complete or quit this task. It also loads in the words associated with the

given encounter, allowing for less stress upon the system.

After Dialogue or objects related to word learning are found, they are then loaded into the dictionary section of the game, in which players can then press words that they would like to know the meaning of, and read about the translation and meaning. Players are also allowed to explore different words learned in different dialects such as Tagalog, Cebuano & Hiligaynon.

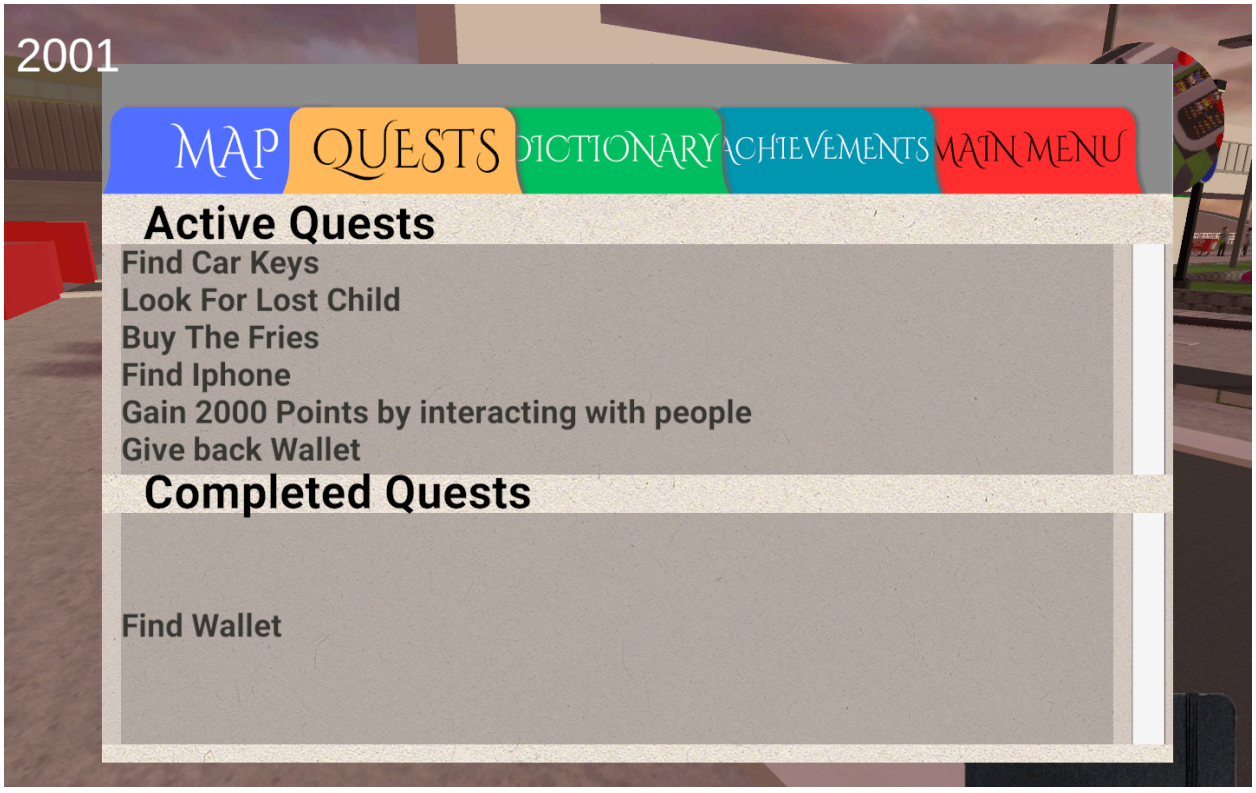
```
    public void Search(TextMeshProUGUI yes){
        string[] data = textAssetdata.text.Split(new string[] {",","\n"}, System.StringSplitOptions.None);
        for (int i = 0; i < data.Length; i++){
            if(yes.text == data[i]){
                if(panel1.TagalogActive){
                ManilausedWord.text = data[i];
                ManilaTranslation.text = data[i+1];
                ManilaDefinition.text = data[i+2];
                }
                else if(panel1.CebuanoActive){
                CebuusedWord.text = data[i];
                CebuTranslation.text = data[i+1];
                CebuDefinition.text = data[i+2];
                }
                else if(panel1.HiligaynonActive){
                BacolodusedWord.text = data[i];
                BacolodTranslation.text = data[i+1];
                BacolodDefinition.text = data[i+2];
                }
```

```
public void NewAdd(string added){
    thisnewWord = null;
    wordused = false;
    string[] data = usedWords.text.Split(new string[] {",","\n"}, System.StringSplitOptions.None);
    for (int i = 0; i < data.Length; i++){
    a = i;
    if(data[i] == added){
        wordused = true;
    }
    }
    if(!wordused){
    if("Tagalog" == data[a] || Manila){
    thisnewWord =  Instantiate(newWord, ManilaparentObject.transform);
    thisnewWord.GetComponent<TextMeshProUGUI>().text = added;
    }
    else if("Cebuano" == data[a] || Cebu){
    thisnewWord =  Instantiate(newWord, CebuparentObject.transform);
    thisnewWord.GetComponent<TextMeshProUGUI>().text = added;
    }
    else if("Hiligaynon" == data[a] || Bacolod){
    thisnewWord =  Instantiate(newWord, BacolodparentObject.transform);
    thisnewWord.GetComponent<TextMeshProUGUI>().text = added;
    }
    }
}
```

These code snippets present the addition of words into dictionary first checking if word has already been loaded into used words CSV and if clear will load word into dictionary and will then add the word into CSV for proper use.

Game is comprised of 9 different scenes, 2 tutorial scenes, and 6 explorable real life Locations within the Philippines filled with objects and NPCs for the player to interact and learn Filipino, Cebuano & Hiligaynon words from, Location such as the rizal

-monument and ruins in bacolod present a realistic exploration ground for immersion and study.




```
public void FinishedQuest(){
        if(CompletedquestTextBox != null){
        if(QuestManager.questManager.completedNames.Count == 0){
            if(noQuestsText != null){
                int randomNumber = (Random.Range(0, noQuestsText.Length));
                CompletedquestTextBox.text = noQuestsText[randomNumber];
            }
        }
        else{
                StringBuilder stringsBuilder = new();
                foreach (string quest in QuestManager.questManager.completedNames){
                stringsBuilder.AppendLine(quest);
                }
                CompletedquestTextBox.text = stringsBuilder.ToString();
        }
        }
                CompletedquestTextBox.rectTransform.sizeDelta = new Vector2(questTextBox.rectTransform.sizeDelta.x,questTextBox.preferredHeight);
}
public void WriteQuest(){
    if(questTextBox != null){
        if(QuestManager.questManager.questNames.Count == 0){
            if(noQuestsText != null){
                int randomNumber = (Random.Range(0, noQuestsText.Length));
                questTextBox.text = noQuestsText[randomNumber];
            }
        }

        else{
            StringBuilder stringBuilder = new();
            foreach (string quest in QuestManager.questManager.questNames){
                stringBuilder.AppendLine(quest);
            }
                questTextBox.text = stringBuilder.ToString();

            }
        }
        questTextBox.rectTransform.sizeDelta = new Vector2(questTextBox.rectTransform.sizeDelta.x,questTextBox.preferredHeight);
```

Quest system Loads active quests into Active quests list after talking with concerned NPCs allowing. After completing each quest the code will then remove the quest string from the active quest and load it into the completed quest tab.

Map transportation utilizes switching scene asynchronous for smooth transition between scenes. Switching regions within the philippines will always load you into airports of specific regions such as NAIA for manila. After which the player will be tasked with earning X amount of points to proceed to the larger explorable area within the region which in manila would be intramuros.

Adventures of George

Adventures of George is a 2.5D shooter game in which you play as George and are tasked with traversing the dungeon filled with monsters, the player can defend themselves through the press of the mouse button, instantiating a bullet projectile that damages opponents caught in its flight path. They will have to manage resources such as bullets that are infinite but have a reload time to manage as well as a health bar that shows how close George is from dying and spawning back at the closest intitiated

respawn point.

Aiming within the game utilizes raycasts and an invisible plane in the background that the George head will look at, then look at the raycast hit location.

OnColliderstay is utilized for appearing Button prompts and Invisible boxes for this purpose

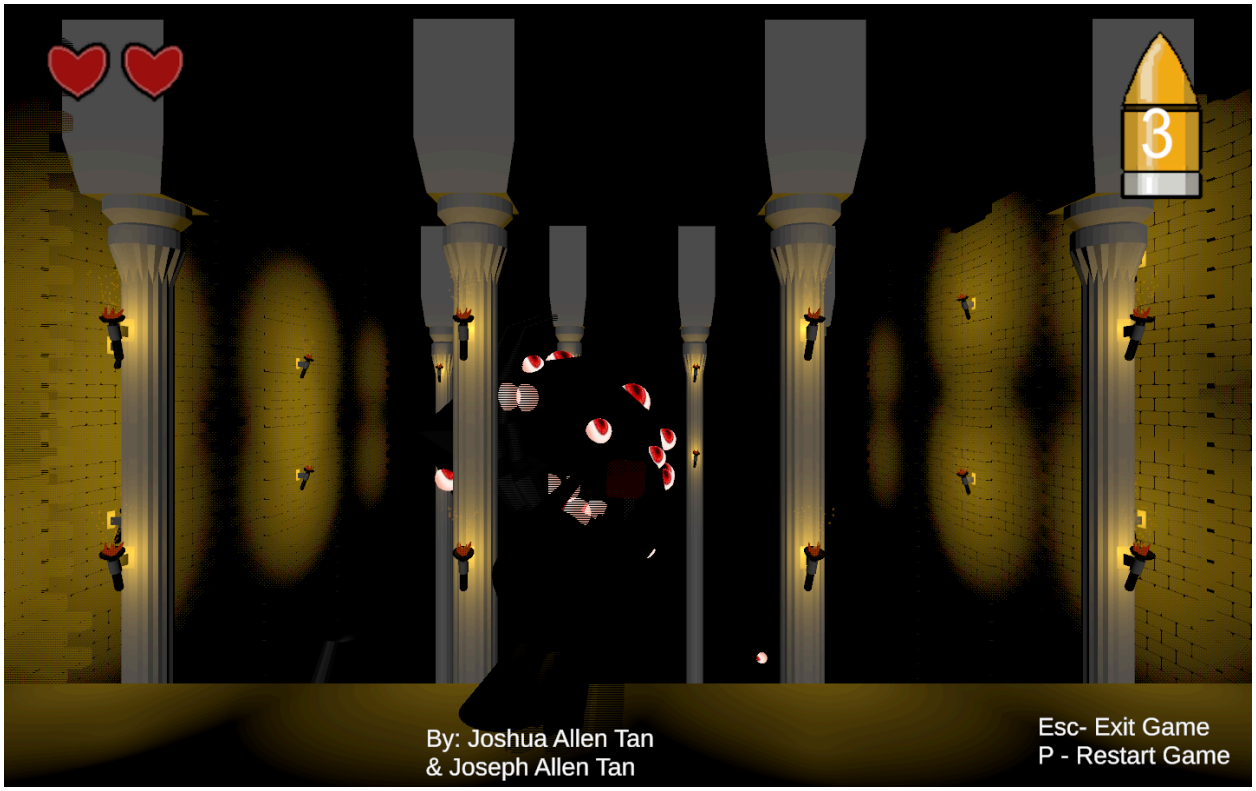
Respawn beacons are intialized after players walk past them allowing player to respawn at that location if they happen to die afterwards.

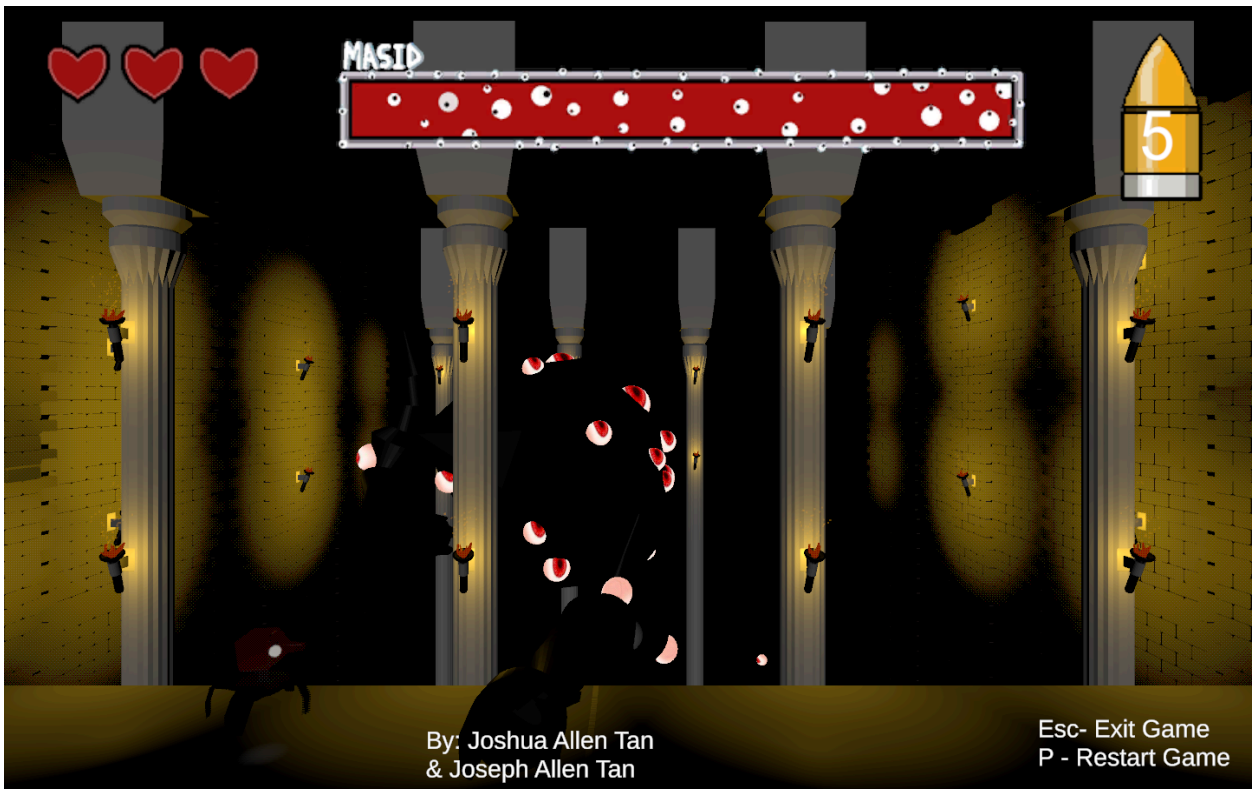
Shadows and PSX filter are utilized in the game in order to give an older look to the game and boost interest and immersion into the setting,

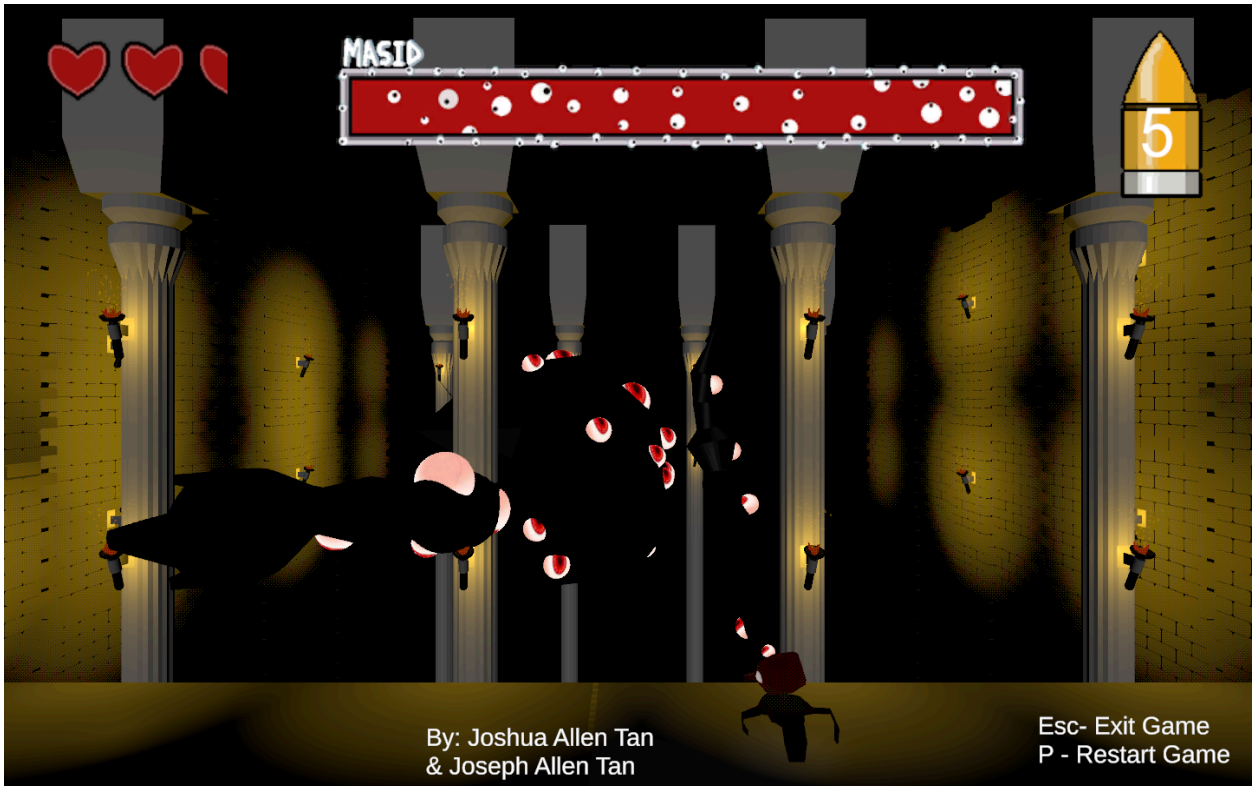
Floating hearts are presented to the player for them to heal after combat encounters, giving them a heart if they manage to take damage beforehand.

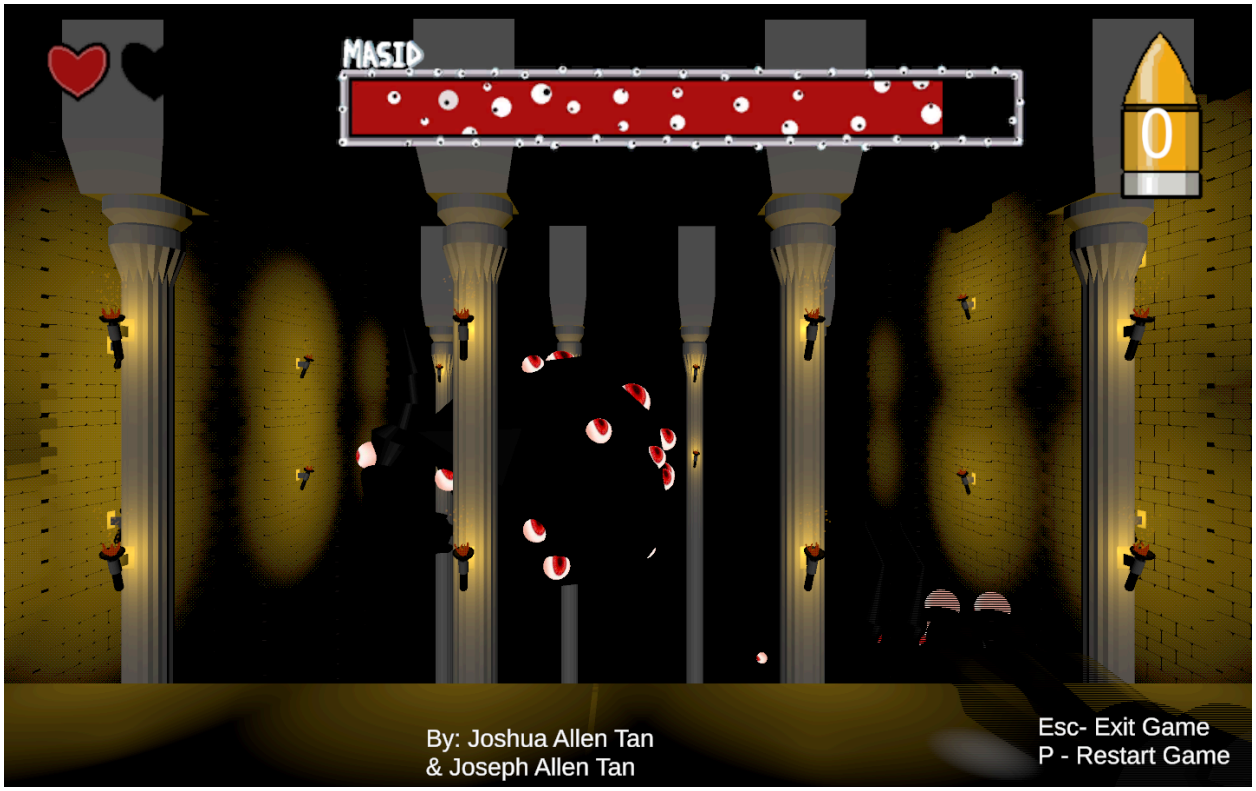
Different instances of opponents shoot projectiles which damage the player entering them into a blinking damage immunity stage allowing them to get their footing before engagin in combat once again. Players can shoot given projectiles or opponents,

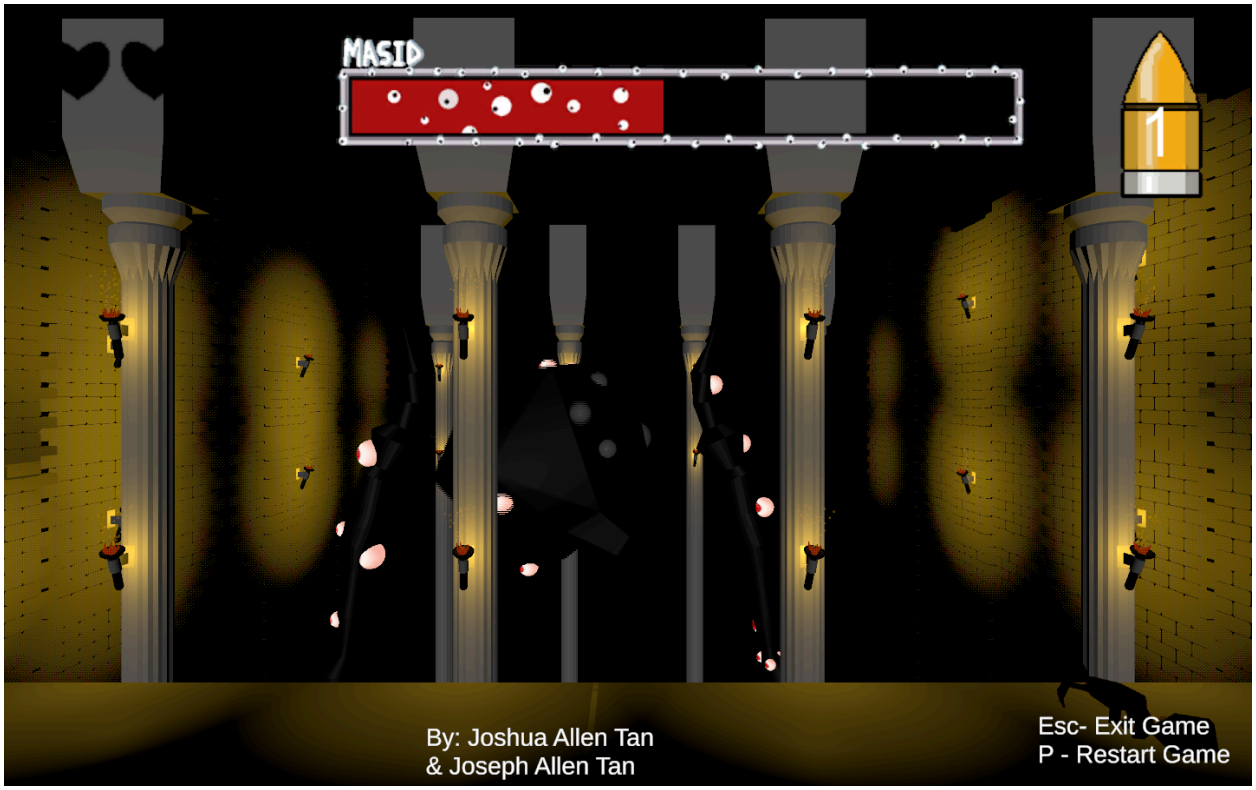
deleting them if they have crossed a certain threshold in their health.






MASIP
5
By: Joshua Allen Tan
& Joseph Allen Tan
Esc- Exit Game
P - Restart Game


MASIP
0
By: Joshua Allen Tan
& Joseph Allen Tan
Esc- Exit Game
P - Restart Game

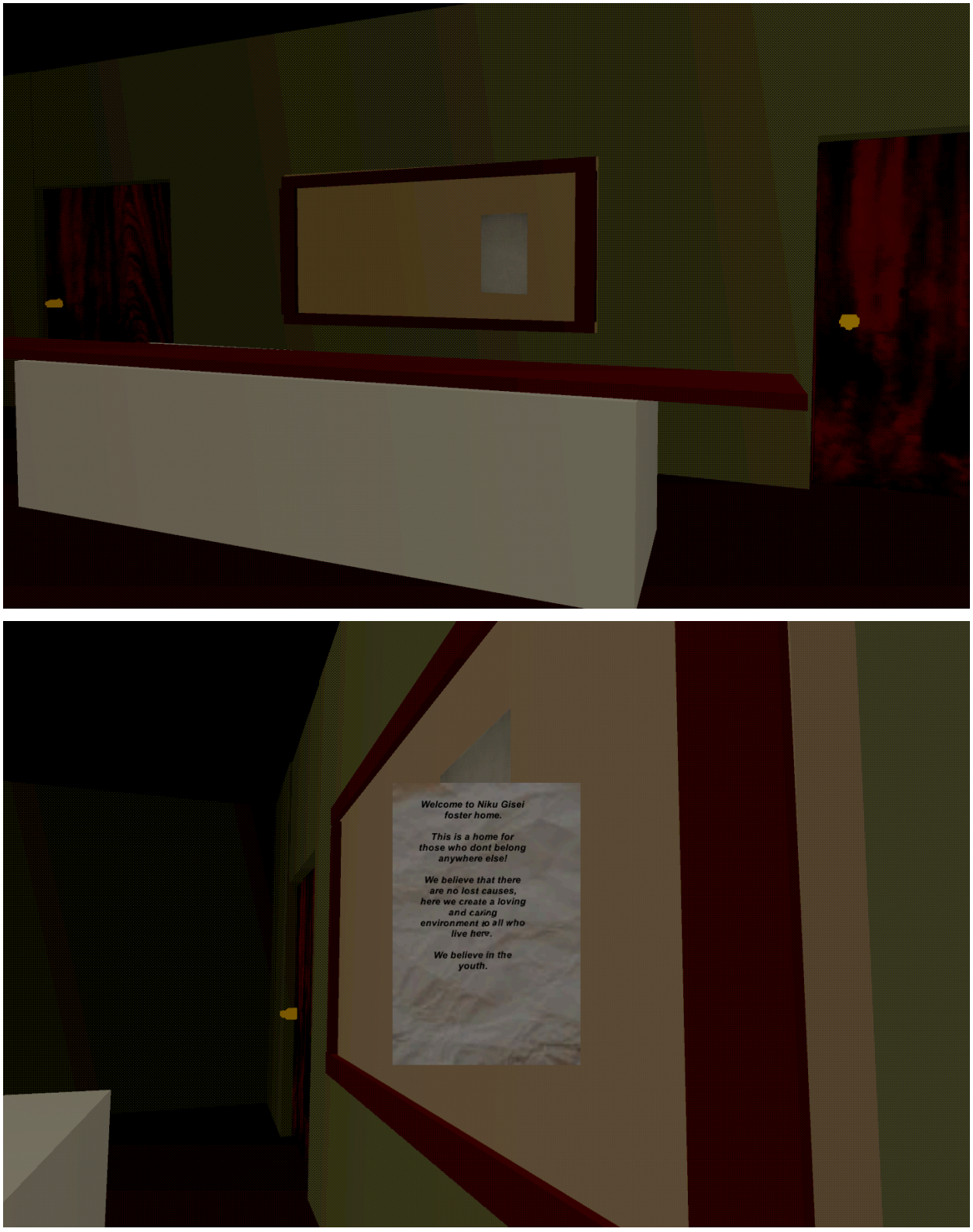


Boss Fight section in which the player has to shoot the boss or limbs in the middle. Mechanics such as animations are used in order to allow the boss to attack accordingly. Before entering combat short cutscene is loaded in which boss is introduced.

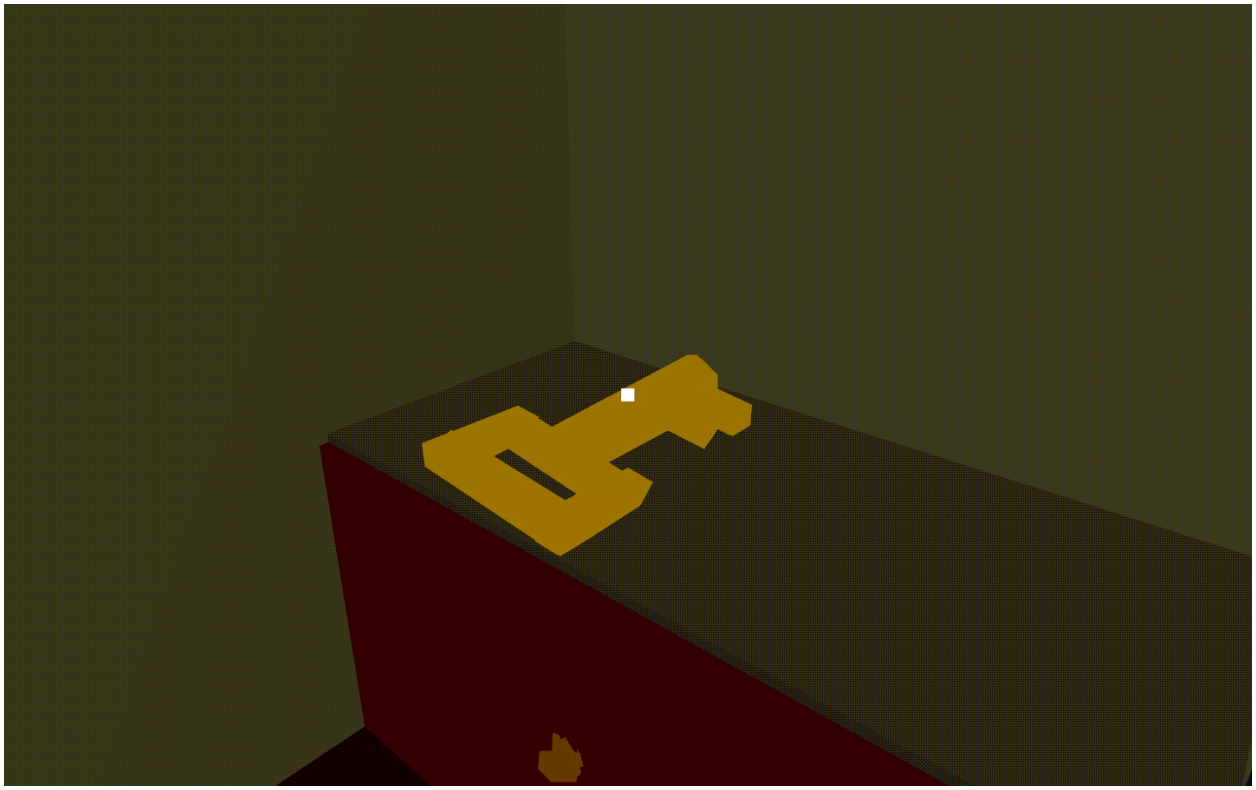
If a player if killed in the encounter, they are then spawned in the closest spawn beacon initiated and are given all health back for a boss fight as it waits for the player to return for conflict.

Niku Gisei

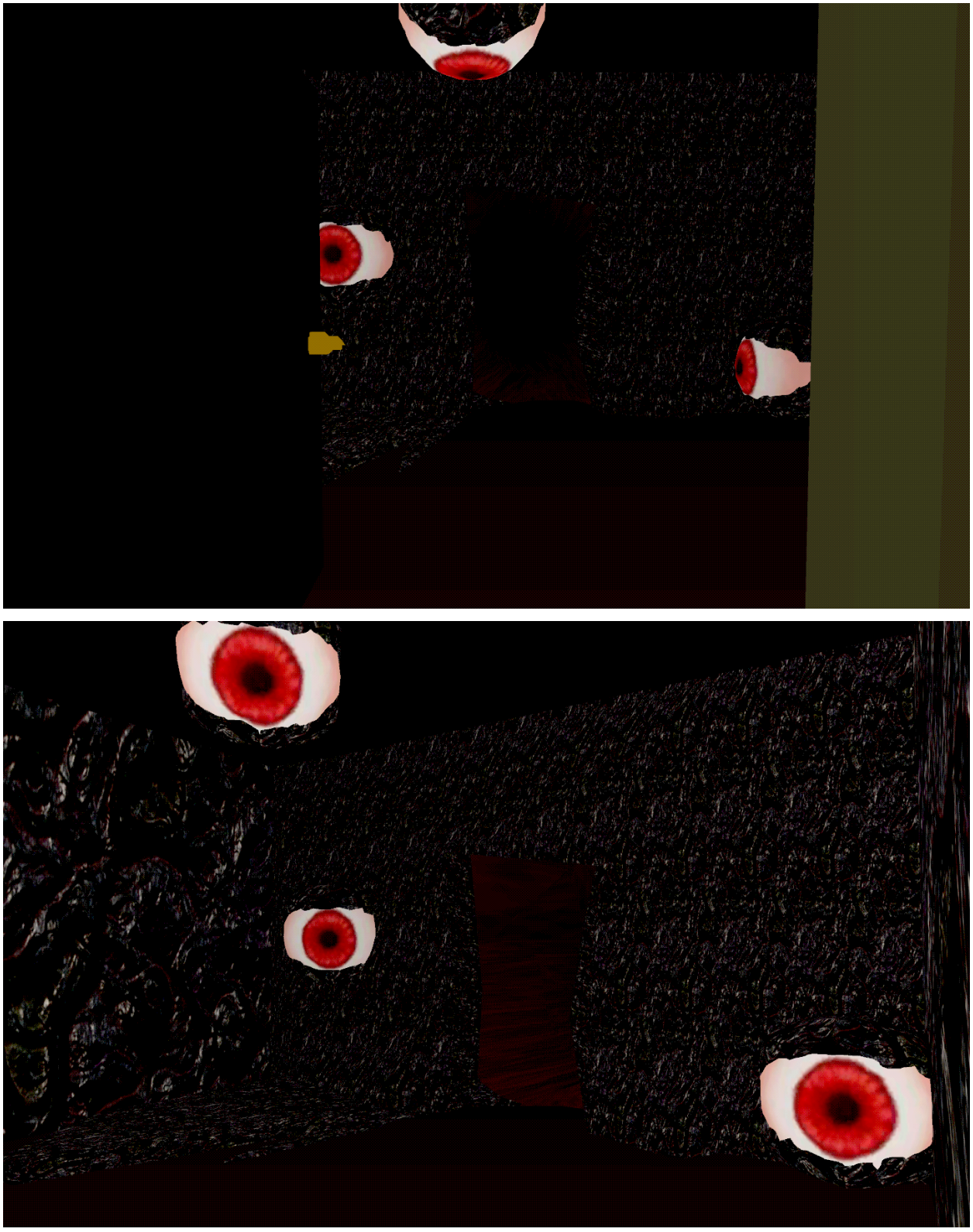
Niku Gisei is a simple horror game utilizing a basic movement script, item interaction scripts, UI elements, scene transitioning, and Game fog to deliver a brief but scary horror experience



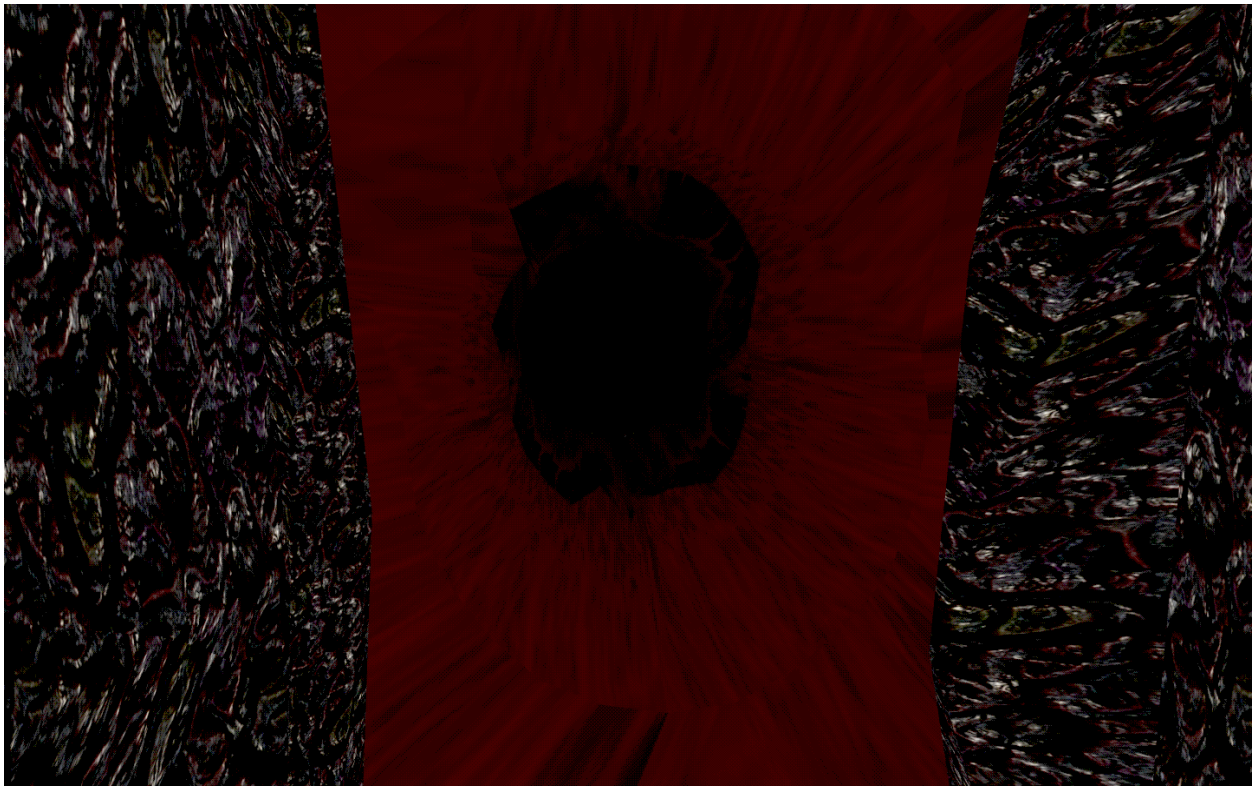
UI elements are used for purposes of presentation to players.

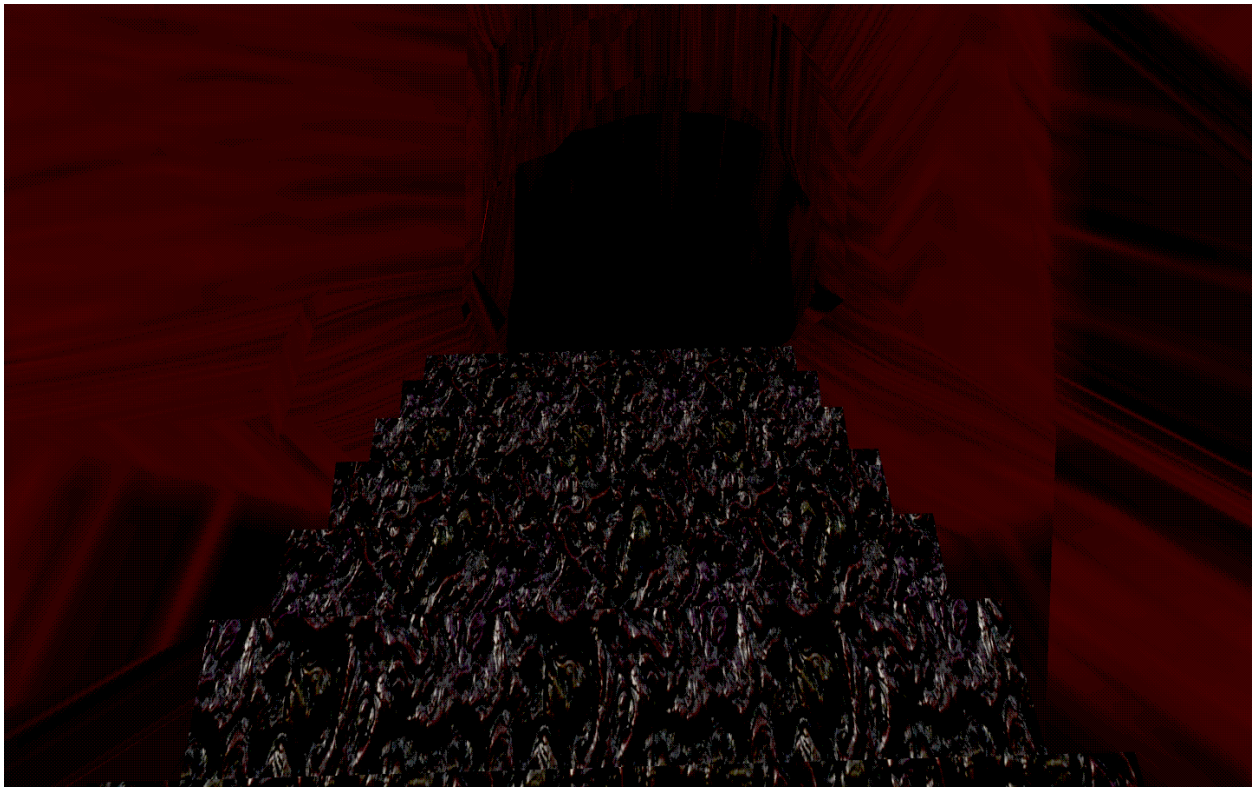


White dot appears on screen whenever interactable object appears on screen, Keys are given to the player to unlock doors important to exploration within the game.

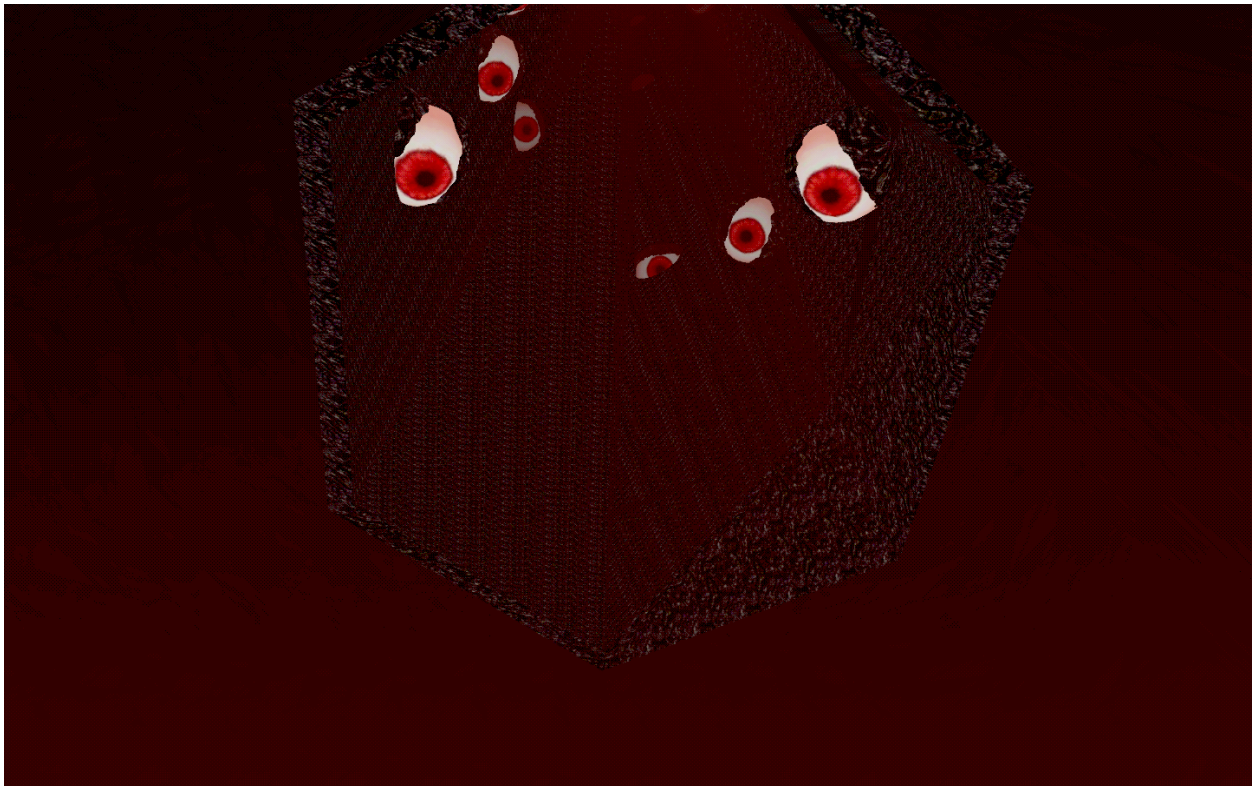


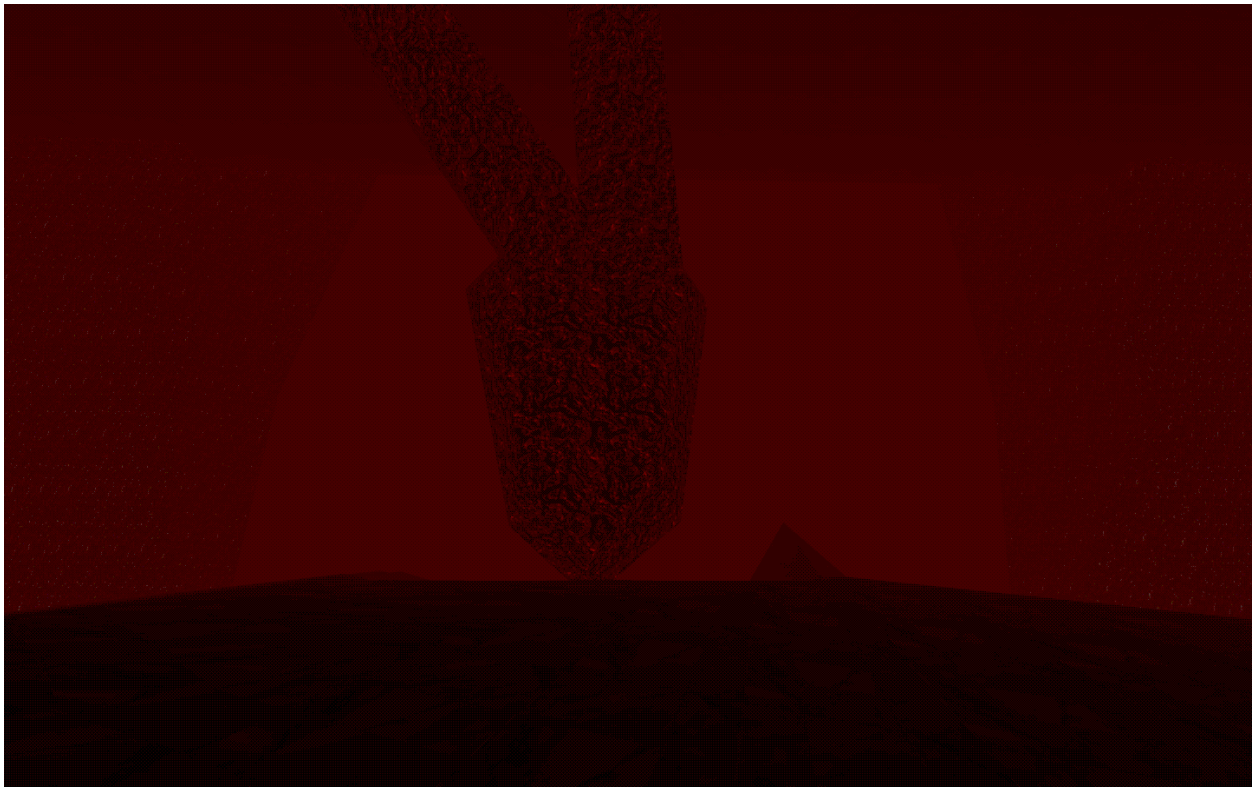
Eyes utilize box collider and looks at the player if they enter the zone of their vision

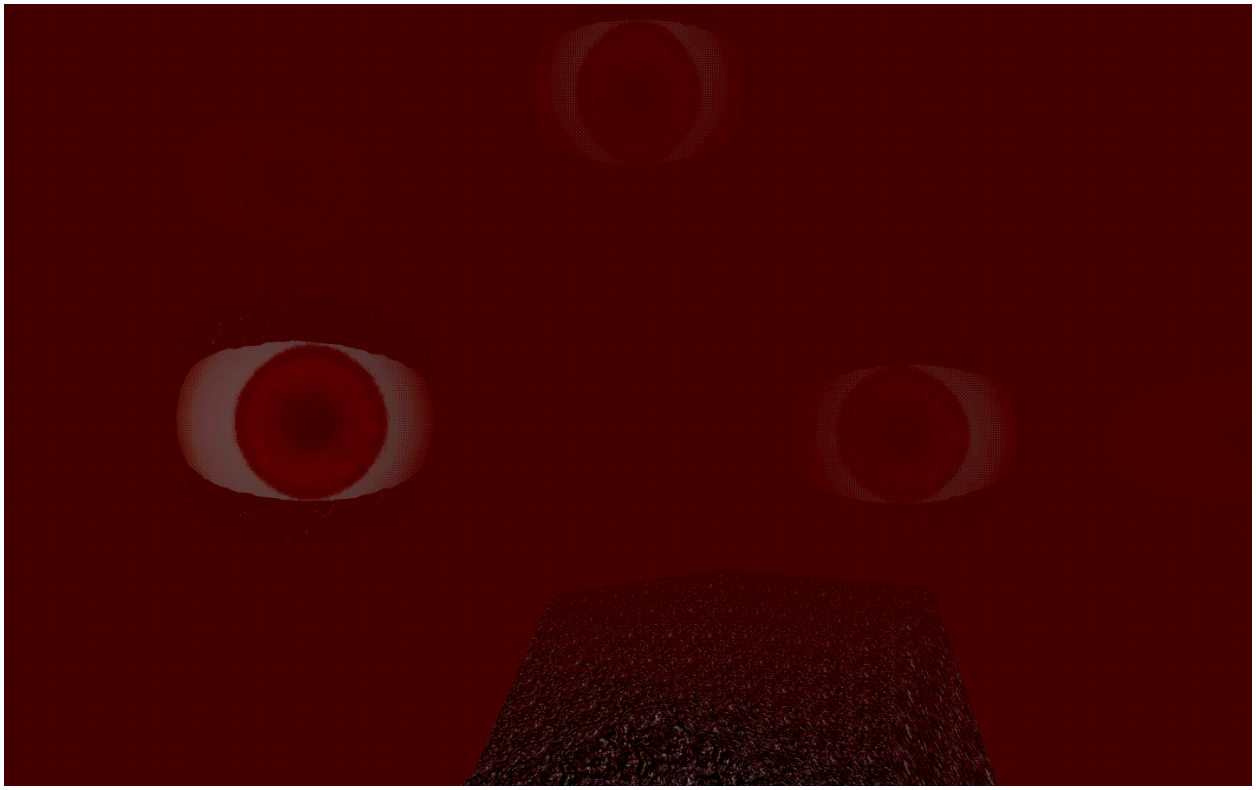


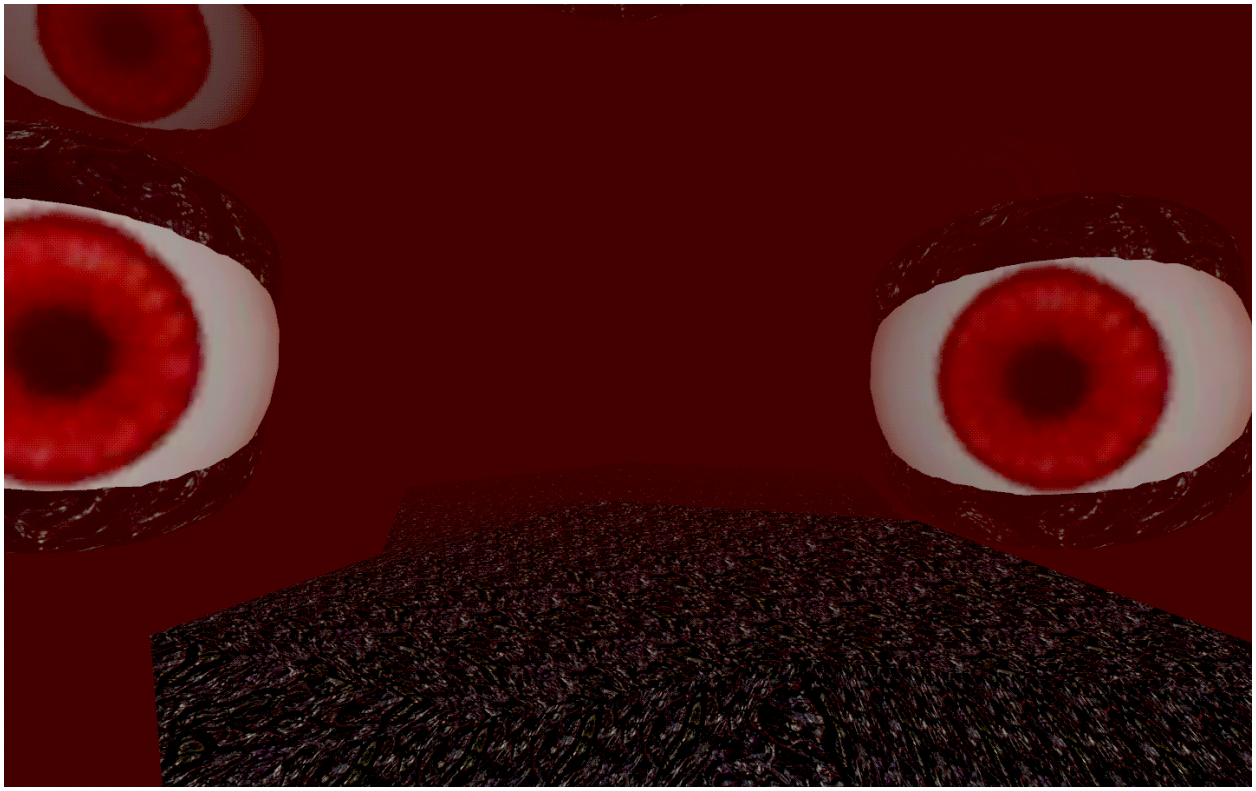


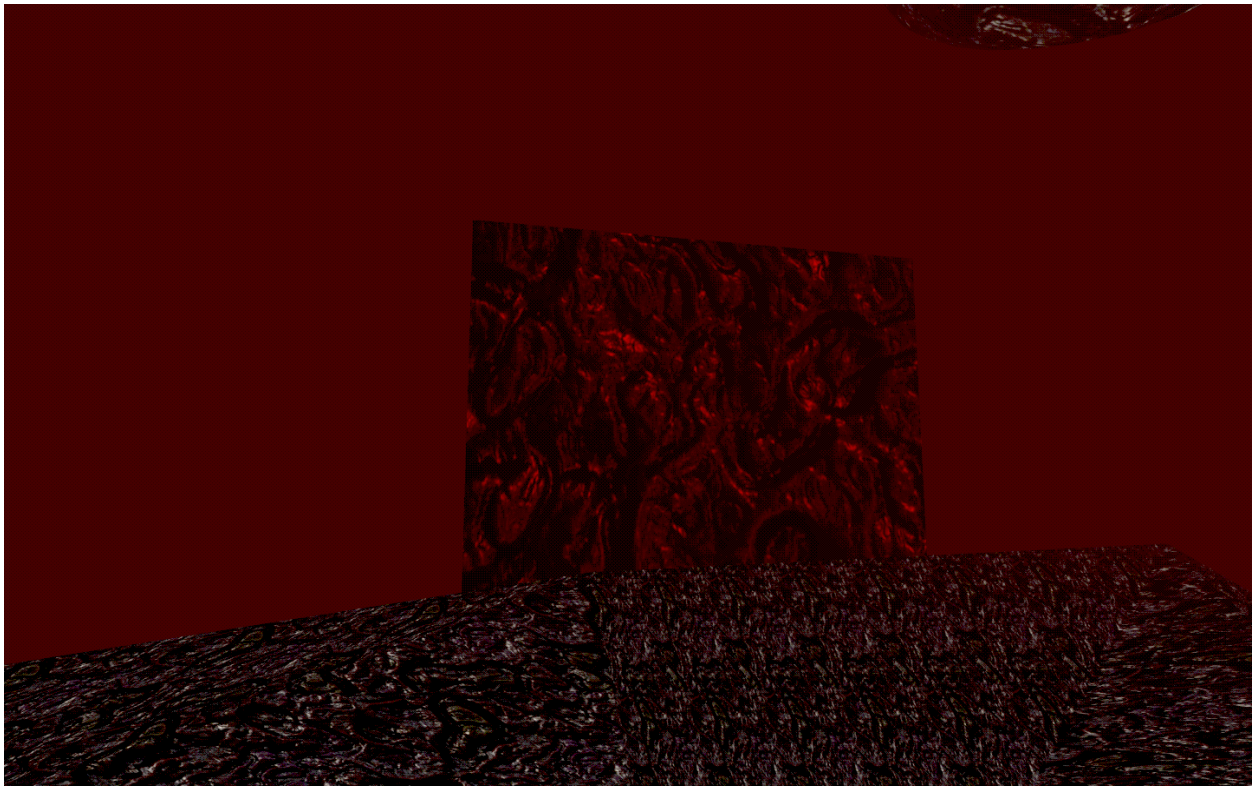
Player is transported to a separate jumpscare scene when faced with a monster attack
Colliders are also utilized with initiation of this process

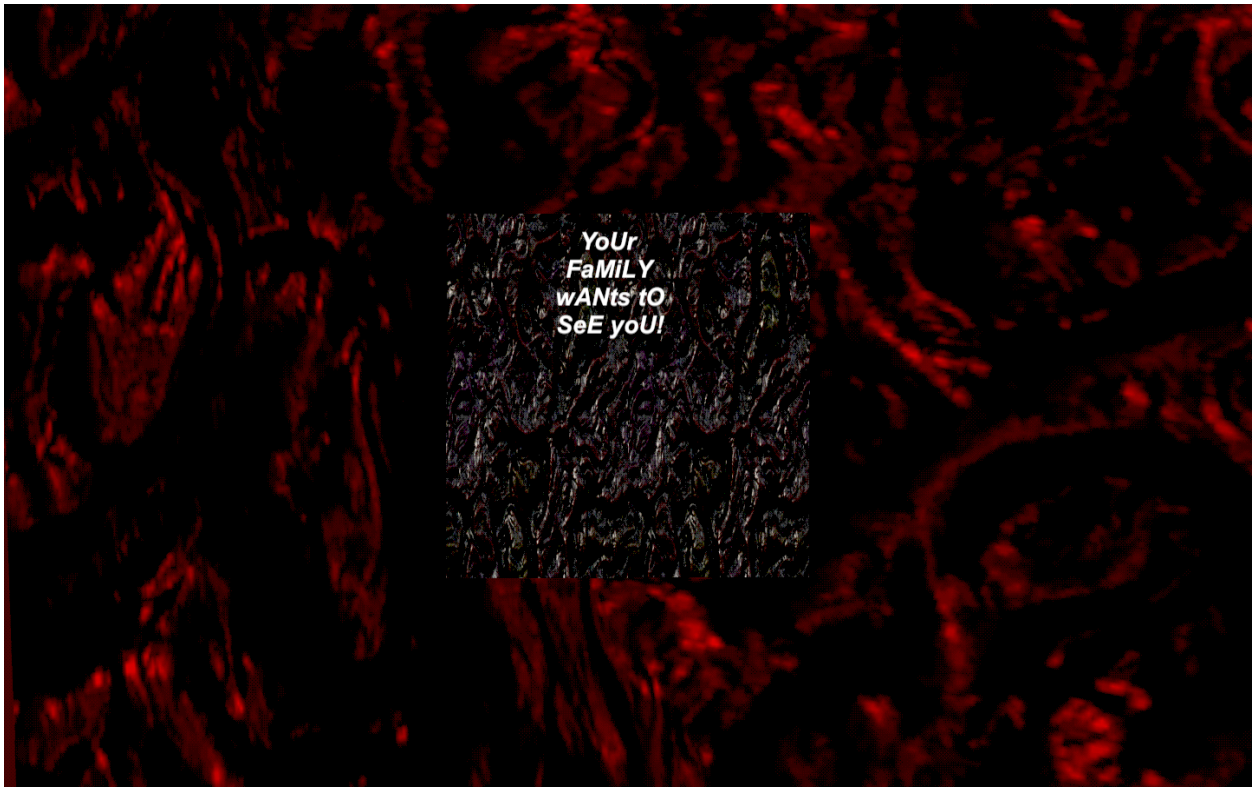


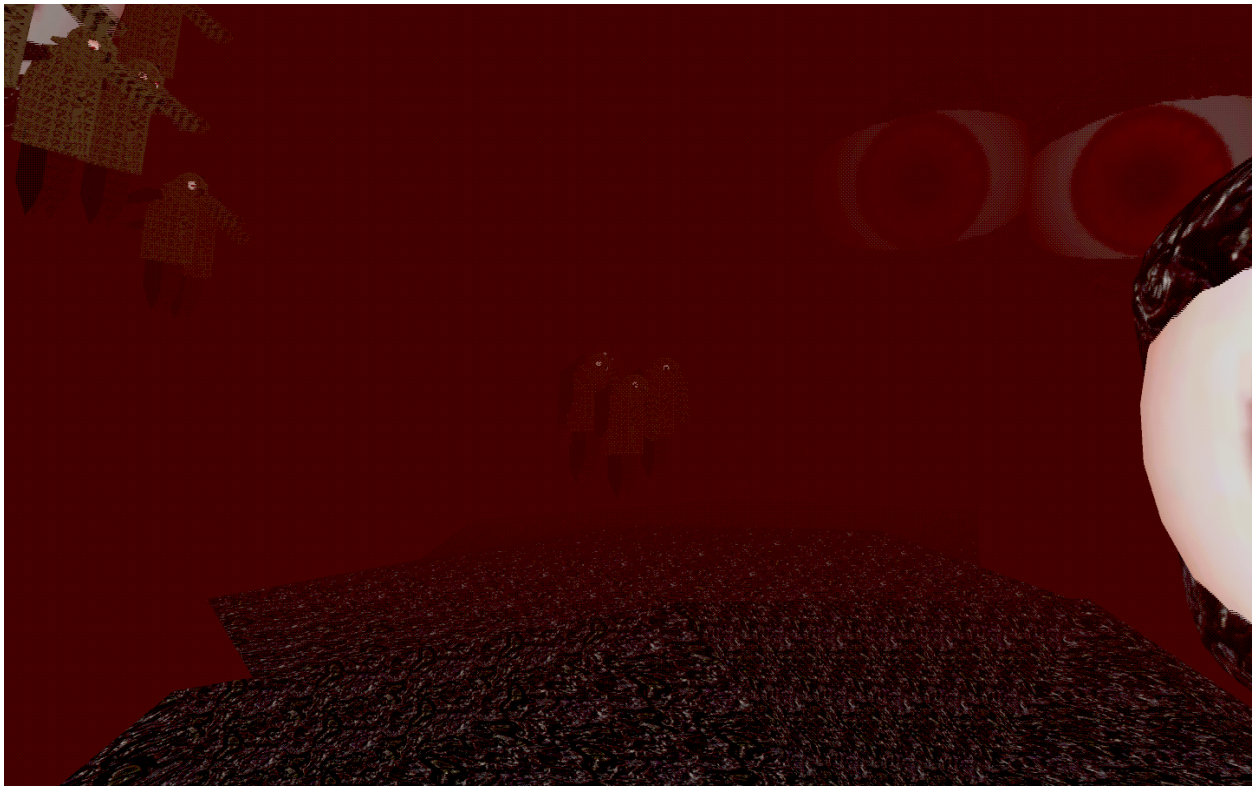


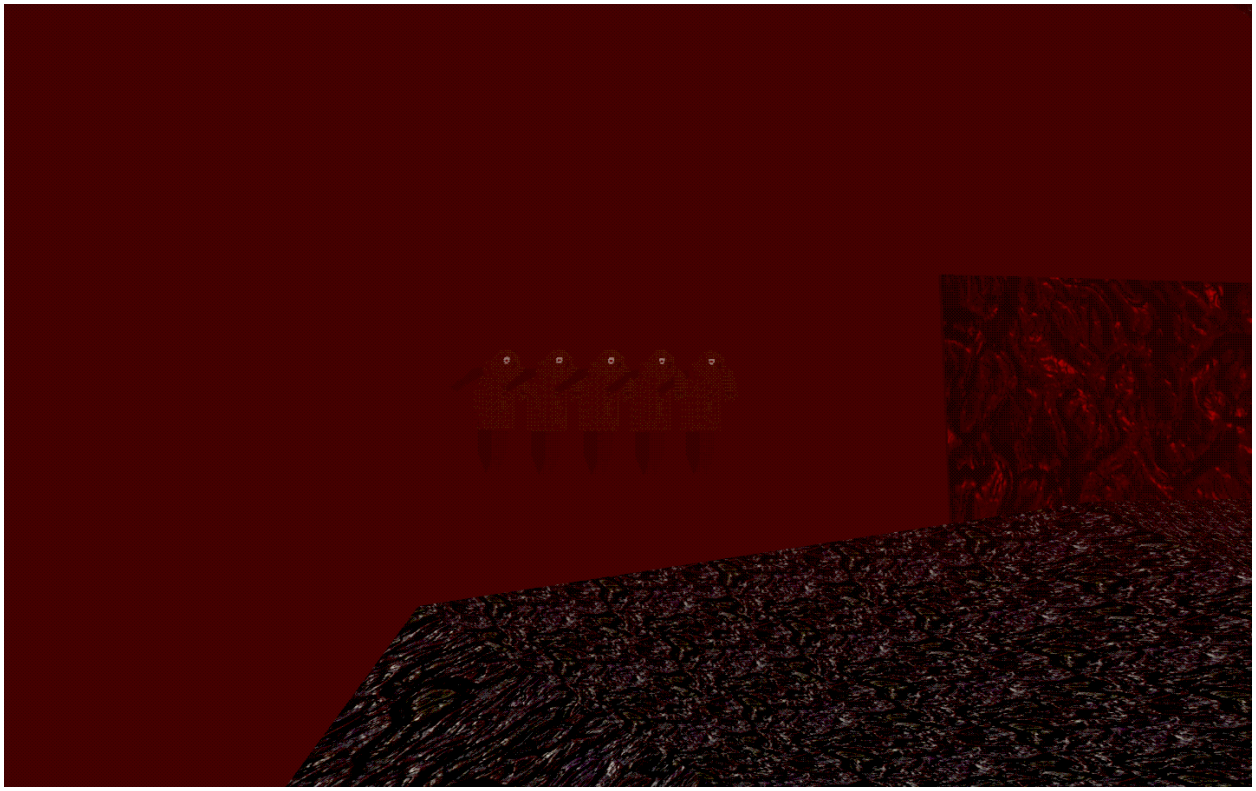


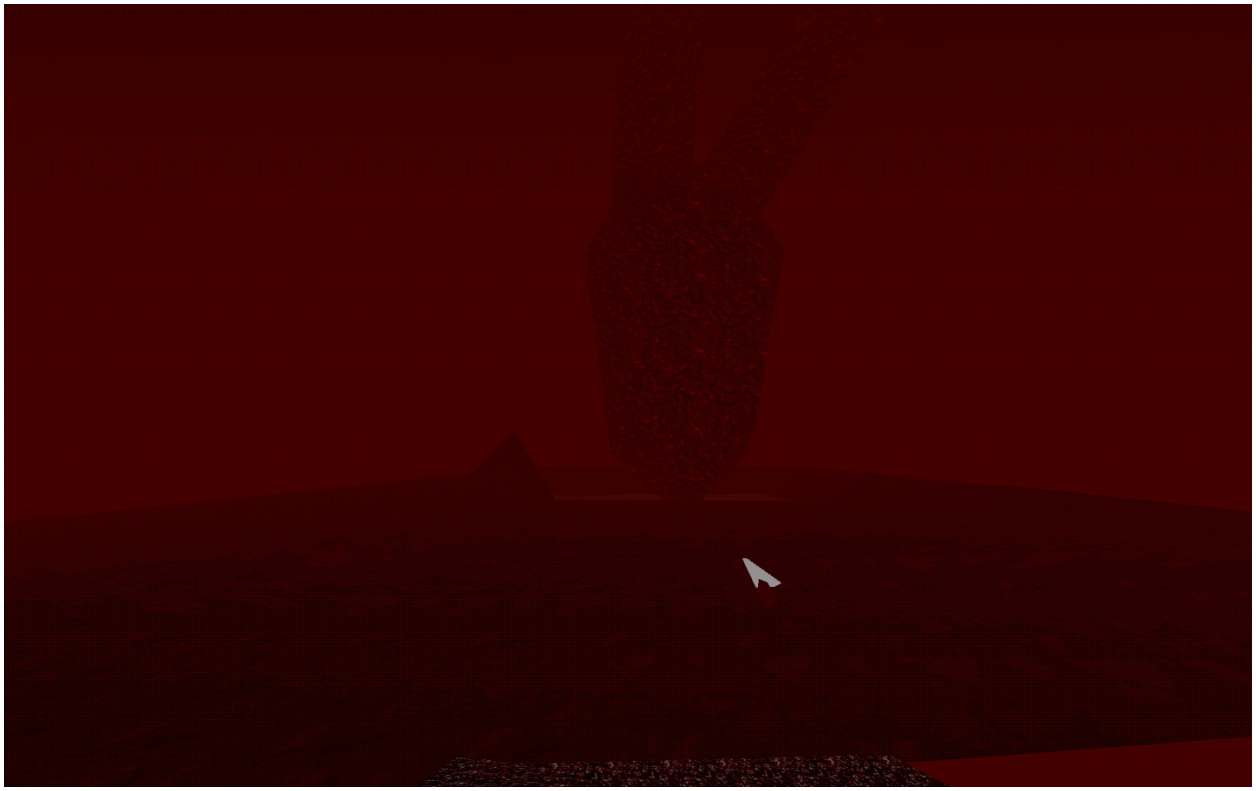


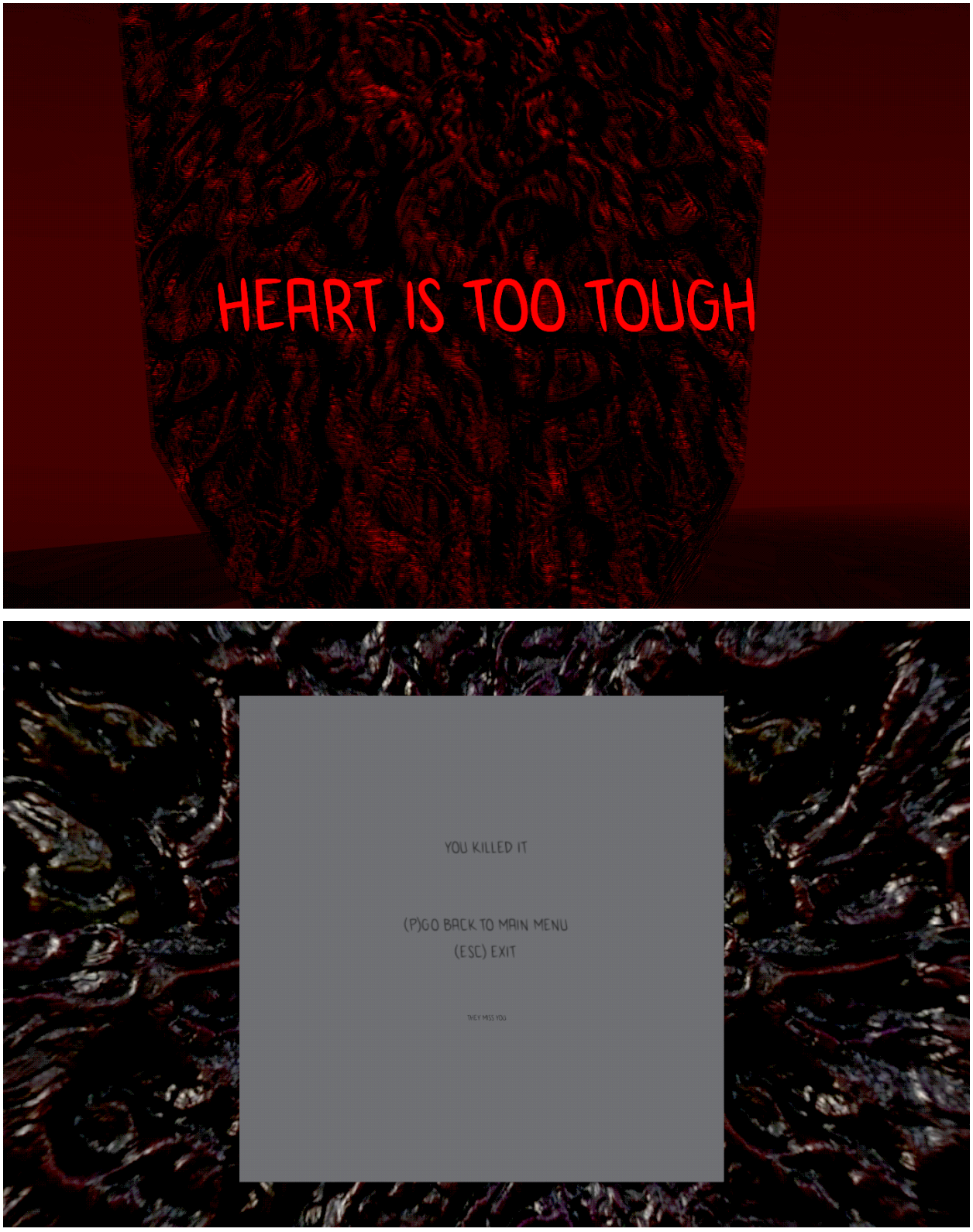



YoUr
FaMiLY
wANts tO
SeE yoU!







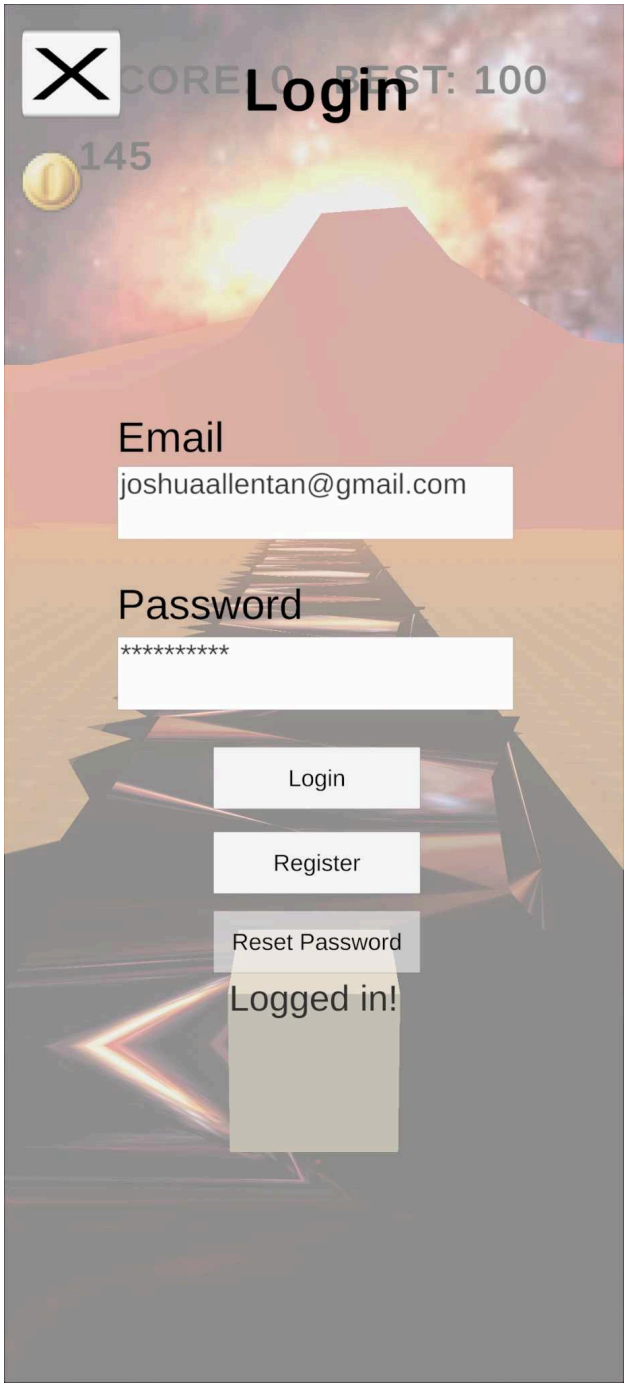


Ending screen

CCQ (Cosmic Classroom Quest)

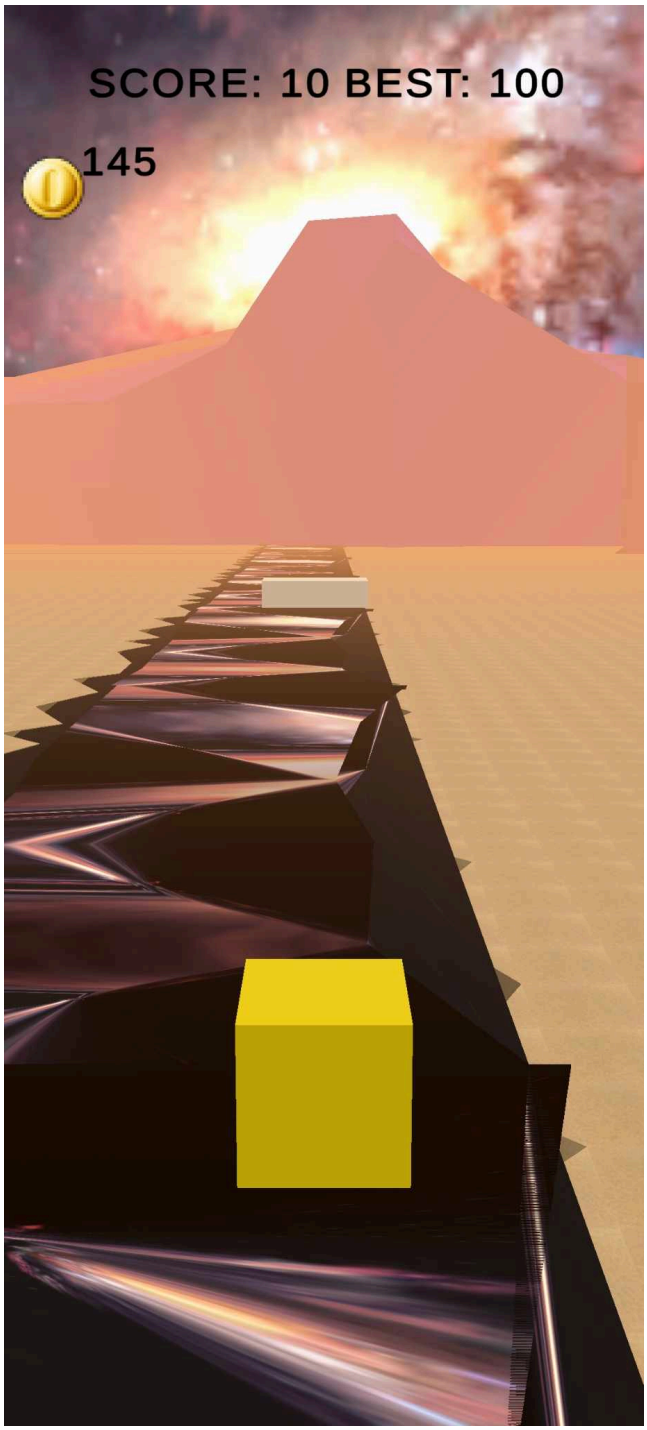
CCQ (Cosmic Classroom Quest) is a trivia endless runner game that combines side to side running, coin collecting and trivia sections in order to educate and provide and enjoyable experience to players. In this open screen you can see the start,

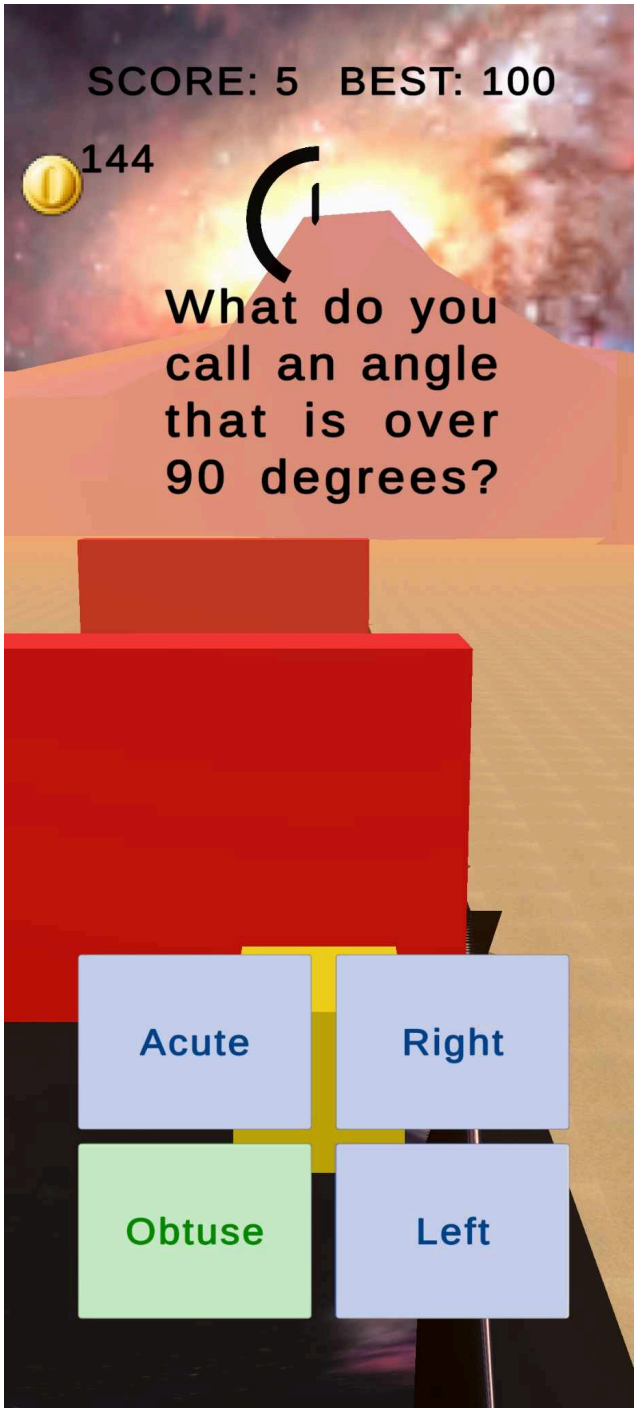
shop, leaderboard, and login buttons. When pressed, each of these buttons takes the player to the given screens.



Login Screen is provided to players if they input a valid account and password they given scores coins and items will be loaded into concerned areas, Otherwise error is showed. Player is also

given Register button that will allow them to Register an email address of their choosing into system if email is in use player will be notified. Finally if player presses Reset password an email will be sent to email account inputted and player can then change account from that location





Gameplay scene in which players can dodge obstacles in white boxes while red boxes enter the player into trivia time in which question and options are loaded and timer is started. When this occurs the Question manager system gives loads on the screen a question. If player presses correct input they will be awarded with

points into their score and given blinking status to denote immunity to damage. Timer is also started where if the player fails to answer question in time they will be given a game over. If player answers wrong to the question they are given minus 10 points and allowed to proceed with same damage immunity.

Online Leaderboard section allows for players to input username into textbox below and will upload highscore into online highscore leaderboard. Leaderboard is automatically loaded into game after button is pressed. Due to this game needs to be operated with wifi On.

The shop section of the game presents different items or colors for the player to buy using the coins in the top left. If they already have an item it will be replaced by the text equip allowing them to use item or color without spending money. This information is also logged into account used in Login page.

Most Recent Project (Work in Progress)