ポートフォリオ

総合学園ヒューマンアカデミー
横浜校　ゲームカレッジ
プログラマー専攻

佐々木遼介

目次

プロフィール

氏名　　：佐々木遼介
性別　　：男
年齢　　：19歳
希望職種：プログラマー

開発環境

・Unity
・Visual Studio2022
・GitHub

開発支援ツール

・Adobe　Animate
・Illustrator
・Photoshop

その他のツール

・Slack
・GoogleDrive

使える言語

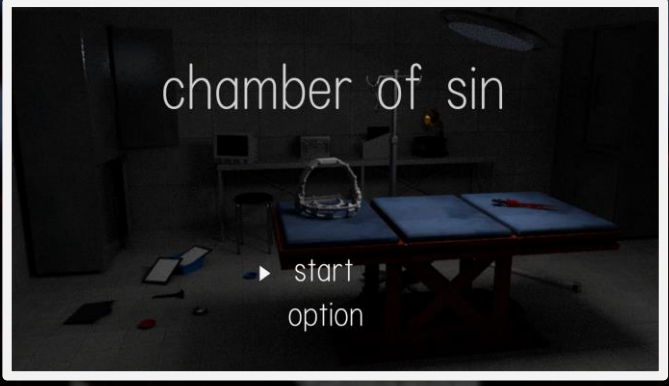
・C/C++
・C#
・Action Script
・HTML/JavaScript
・PHP

Unity チーム制作作品

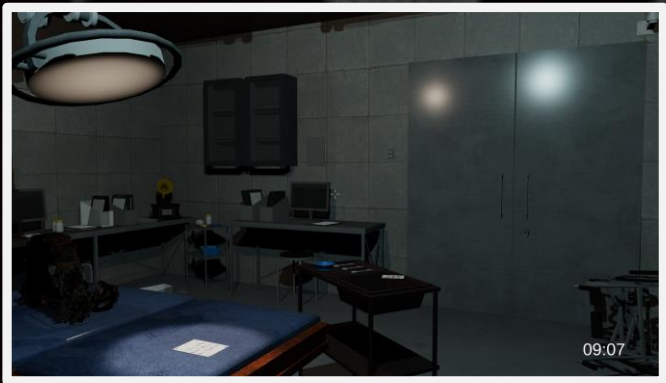
3Dホラーゲーム作品

chamber of sin 罪の部屋

ジャンル:3Dアクションホラー
制作人数:プログラマー3人、CGデザイナー2人
役割:リードプログラマー
制作期間:4か月間
担当箇所:Git管理、プレイヤー操作、ギミック実装、ステージ作成、タイトルUI、ポーズUI、ゲームクリアUI、BGM



このチーム制作で、初めての3Dゲーム作品です。ホラーを題材にして作りました。このゲームの目的は、謎を解いて頭の装置を外し、最終的に、部屋からの脱出を目指します。



この部屋にはいくつかのギミックがあり、身の回りにあるアイテムをヒントに制限時間内に部屋から脱出するための手がかりを探します。

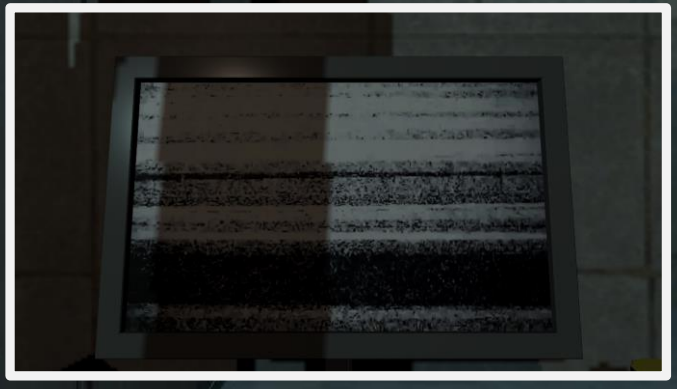
このチーム制作ではプランナーがいなかったので、私はこの制作で初めてリードプログラマーになった時に最後までやり遂げられるか不安がありましたが、チーム内のメンバーの助けもあり、なんとか最後までやり遂げゲームを完成させることができました。

操作方法は、AWSDボタンを使い、前に動いたり後ろに下がったりします。

部屋の中には、横の画像のように触れることができる物やアイテムがあり、Eボタン、もしくはマウスの左クリックでで調べることができます。

こだわったポイント

今回のホラーゲームでこだわったところは、演出の部分です。例えば、下の画像のように明かりをつけると突然モニターが砂嵐になったり、アイテムを手に入れたときに照明が落ちたりします。それ以外にも奇妙な要素があります。



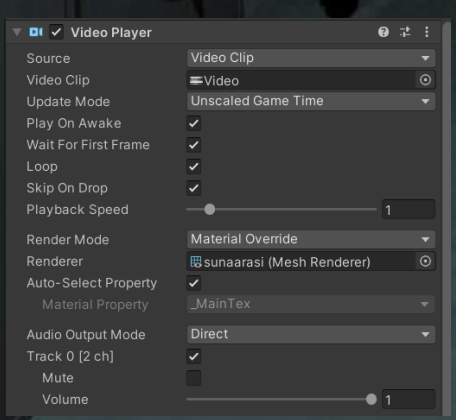
担当箇所・モニターの演出

```
#region RayCast設定
// RayCastに関する設定
1 個の参照
public void RayCast()
{
    // Rayをカメラの位置から飛ばす
    var rayStartPosition = fpsCamera.transform.position;

    // Rayをカメラの向いている方向に飛ばす
    var rayDirection = fpsCamera.transform.forward.normalized;

    var rayHit = Physics.Raycast(rayStartPosition, rayDirection, out raycastHit, distance, LayerMask.GetMask("Item"));

    // Rayを可視化する
    Debug.DrawRay(rayStartPosition, rayDirection * distance, Color.red);
// RayがItemレイヤーを持つObjectにあたった場合処理
if (Physics.Raycast(rayStartPosition, rayDirection, out raycastHit, distance))
{
    if (raycastHit.collider.gameObject == What[0])
    {
        What[0].SetActive(false);
        Debug.Log("HitObject : " + raycastHit.collider.gameObject.name + "を入手しました。");
    }
    else if (raycastHit.collider.gameObject == What[1] && textLabel.text == string.Empty)
    {
        // 扉が閉じている時に処理
        if (!lightOn)
        {
            Yes();
            Plight.SetActive(true);
            SecondMovie.SetTrigger("SecondMovie1");
        }
```

レイキャストを使い、オブジェクトに触れているか判定を行っています。アウトラインを表示することでどのオブジェクトが対象なのか、わかりやすくしています。Lightをつけると、部屋が明るくなり、モニターが突然つきます。
一度きりのアニメーションなので、if文の中でSetTriggerを使って再生しています。

⇒

↘



砂嵐に関しては、VideoPlayerを使い、オブジェクトに動画を映しています。

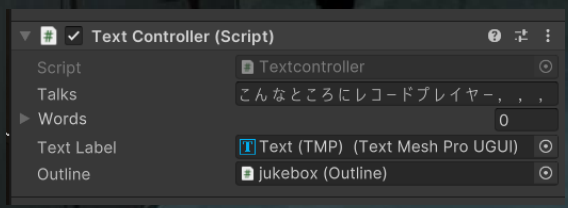
担当した場所の工夫した点

```
// コルーチンを使って、１文字ごと表示する。
2 個の参照
IEnumerator Dialogue()
{

    // 半角スペースで文字を分割する。
    words = talks.Split(' ');

    foreach (var word in words)
    {
        // 0.09秒刻みで１文字ずつ表示する。
        textLabel.text += textLabel.text + word;
        yield return new WaitForSeconds(0.09f);
    }

    // 全ての文字を表示した後、2.7秒後にテキストを非表示にする
    yield return new WaitForSeconds(2.7f);
    textLabel.text = "";

}
```

エ夫したところは、このゲームの主人公のセリフを1文字ずつ表示したことです。StartCoroutineを使って、0.09秒の間隔で1文字ずつ表示するようにしました。全ての文字を表示し切ったら2.7秒後に消えるのは、プレイヤーが、そのセリフを見逃すのを防ぐためです。1文字ずつの表示方法にしたことで実際に喋ってるかのような感じを出してみました。

⇒


```
//publicにしてそれぞれのオブジェクトに対するTextを自由に設定できるようにする。
public string talks;
public string[] words;
public TextMeshProUGUI textLabel;
// Outlineコンポーネント
public Outline outline;
```

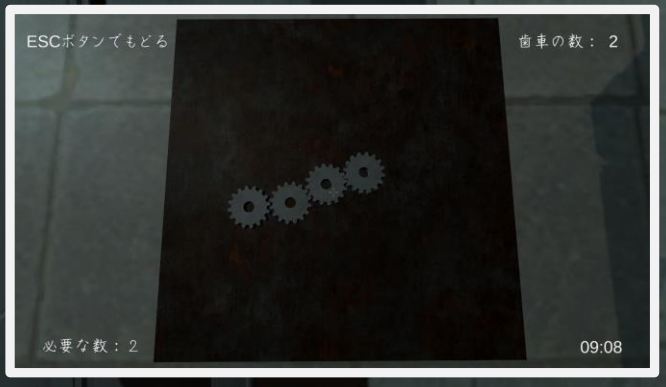


また、public変数にすることで、色々なオブジェクトにつけて、自由にセリフを入力できるようにしました。

担当箇所・ギミック要素

```
Unity メッセージ | 0 個の参照
private void FixedUpdate()
{
    // 1つ目の歯車の回転
    Gear_.transform.Rotate(Vector3.up, rotationSpeed * Time.fixedDeltaTime);

    // 2つ目の歯車の連動した回転
    if (connectedGear != null)
    {
        connectedGear.transform.Rotate(Vector3.up, -rotationSpeed * Time.fixedDeltaTime);
        // 3つ目の歯車の連動した回転
        if (connectedGear1 != null)
        {
            connectedGear1.transform.Rotate(Vector3.up, rotationSpeed * Time.fixedDeltaTime);
            connectedGear2.transform.Rotate(Vector3.up, -rotationSpeed * Time.fixedDeltaTime);
        }
    }
}
```

これは歯車のギミックです。必要な数分の歯車を集めると謎が解ける仕組みになっています。実際の歯車の挙動と近いように再現しました。1個目の歯車が回転してる時、それにくっついてる2個目は1個目と反対側に回転するように計算しコードを書きました。

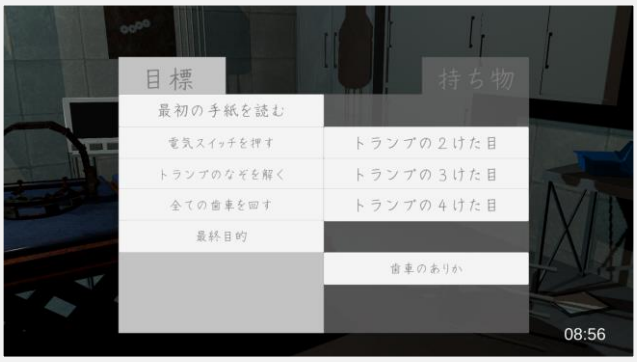


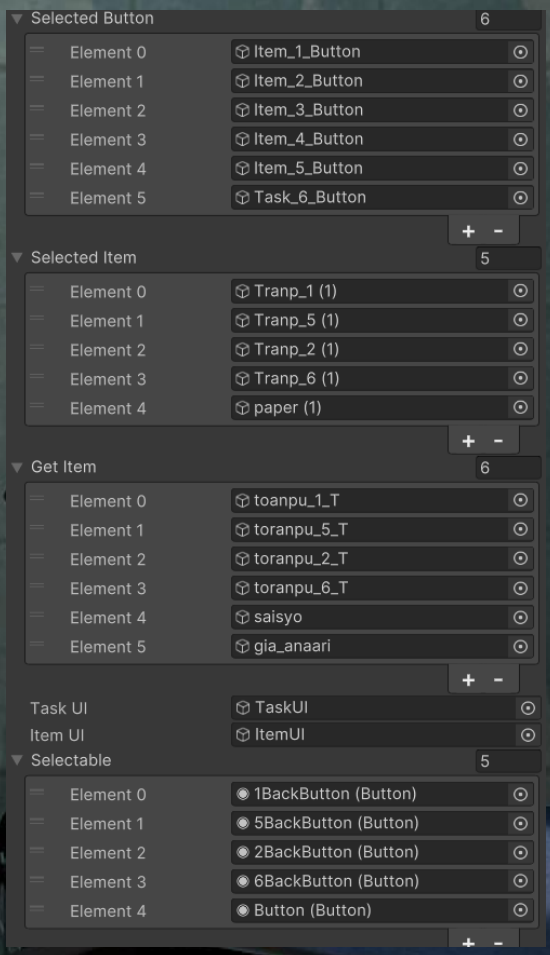

```
// カウントのテキストを更新するメソッド
1 個の参照
void UpdateGearCountText()
{

    if (gearCountText != null && DisableGears[0].activeSelf == false)
    {
    // gearCountが0の場合、1に変更
    if (gearCount == 0)
    {
        gearCount = 1;
    }
    rokka_light.SetActive(true);
    gearCountText.text = "歯車の数： <font=¥"LiberationSans SDF¥">" + gearCount.ToString() + "</font>";
    }
```

歯車を手に入れると、歯車のカウントをします。メソッドを使い、一つの歯車が消えたら、数字を1つ上げるコードです。

担当箇所・その他の要素

謎解き以外の要素として、インベントリを追加しました。随時確認可能。インベントリの中身は、持ち物の確認だけでなく「目標」という項目があります。
これは、Player側が謎解きで行き詰った場合の救済措置として入れました。ヒントのようなものです。
持ち物の項目に関しては、アイテムを取ればそのアイテムを確認できるようにしています。



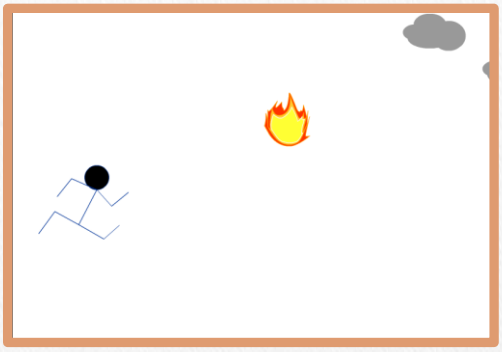



```
public void Update()
{
    for (int i = 0; i < GetItem.Length; i++)
    {
        // GetItemが非アクティブの場合、SelectedButtonをアクティブにする
        if (!GetItem[i].activeSelf)
        {
            SelectedButton[i].SetActive(true);
        }
    }

    for (int i = 0; i < selectable.Length; i++)
    {
        // SelectItemがアクティブの場合、SelectedButtonをアクティブにする
        if (SelectedItem[i].activeSelf && Input.GetMouseButtonUp(0) || SelectedItem[i].activeSelf)
        {
            selectable[i].Select();
        }
    }
}
```

```
public void CloseItem(int itemIndex)
{
    if (itemIndex >= 0 && itemIndex < SelectedItem.Length)
    {
        // 特定のインデックスにある要素をアクティブにする
        SelectedItem[itemIndex].SetActive(false);
        TaskUI.SetActive(true);
        ItemUI.SetActive(true);
        FirstSelect.Select();
    }
}

0 個の参照
public void ShowItem(int itemIndex)
{
    if (itemIndex >= 0 && itemIndex < SelectedItem.Length)
    {
        // 特定のインデックスにある要素をアクティブにする
        SelectedItem[itemIndex].SetActive(true);
        TaskUI.SetActive(false);
        ItemUI.SetActive(false);
        textLabel.text = "";
        selectable[itemIndex].Select();
    }
}
```

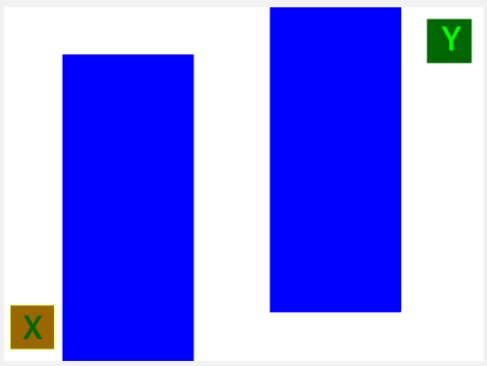
アイテムはたくさんあるので、配列を作りそこに登録できるようにしました。

逃げろ



Animateで逃げろというゲームを作りました。炎が敵で、棒人間がPlayerです。炎が追いかけてくるので、それから逃げるゲームです。炎に捕まるとゲームオーバーです。

Festival



AnimateでFestivalというゲームを作りました。色んなテーマのゲームをクリアして最終的に全クリを目指します。