BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Game Out Of Sight memiliki *gameplay* dengan *first person perspective* yang dipadukan dengan genre survival horror. Dibuat dengan bahasa pemrograman C# game ini mampu berjalan dengan sempurna dengan berbagai fitur. Para pemain dapat menembak para musuh yang menghadang, puzzle kombinasi nomor yang digunakan untuk membuka sebuah pintu, dan melihat berbagai object tertentu seperti kertas yang berisikan clue dari cerita game ini. Disamping berbagai macam fitur yang disebutkan sebelumnya, para pemain juga dapat melihat cutscene yang dimana cerita tersebut dimulai. Dengan dibalut *first person perspective* para pemain dapat merasakan mencekamnya lorong gelap di dalam game Out Of Sight, dengan adanya bantuan suara menakutkan dan teriakan entah darimana, menambah kengerian game ini.

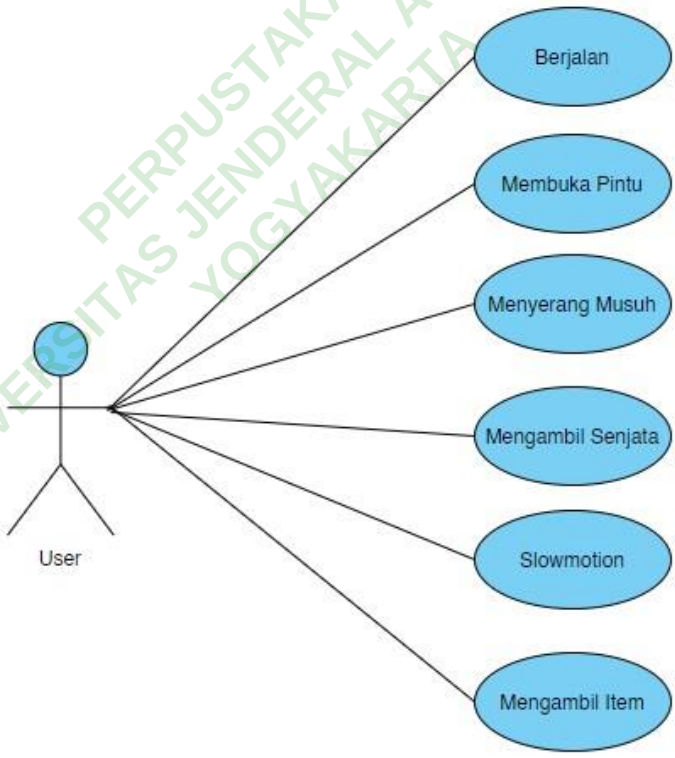
4.2 *CONCEPT* (PENGONSEPAN)

Konsep dalam game ini adalah memadukan antara *Pyschological horror game* dan *Survival horror game*. Game ini ditujukan untuk para remaja berumur 17 tahun hingga 25 tahun dengan dibalut beberapa *puzzle* dan cerita yang menarik untuk membangun keseruan dan ketegangan di dalam game.

4.3 *DESIGN* (PERANCANGAN)

4.3.1 Use Case Diagram

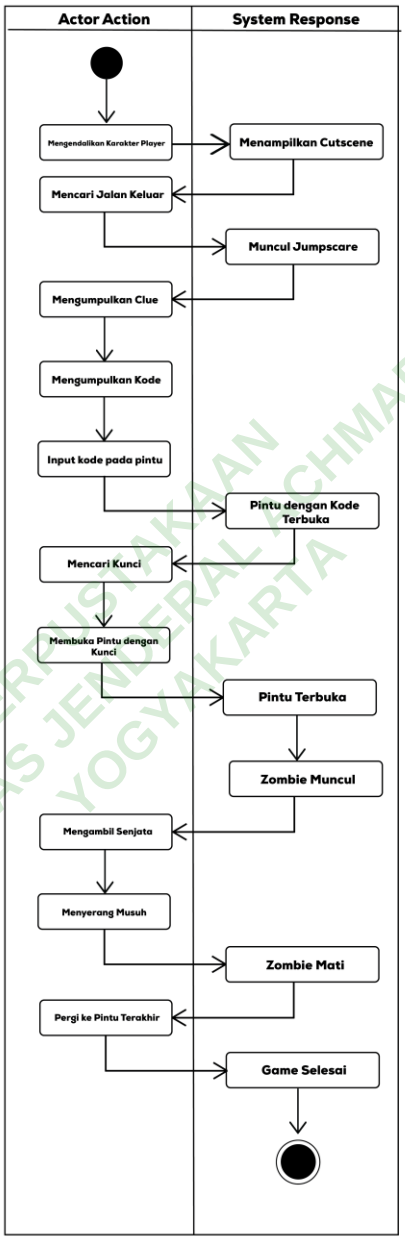
Use case diagram pada *game* yang akan dibuat menggambarkan interaksi antara user dengan *game* Out Of Sight. *Game* ini memiliki 6 fitur utama, yaitu berjalan, membuka pintu, menyerang musuh, mengambil senjata, *slowmotion*, dan mengambil item. Use case diagram menggambarkan user dapat melakukan semua 6 fitur ini. Pemain akan dapat berjalan menggunakan W,A,S,D. Membuka pintu, mengambil senjata, dan mengambil item menggunakan tombol E pada keyboard. Menyerang musuh dengan klik kanan mouse dan mengarahkan pada musuh. Menggunakan *slowmotion* dengan menekan tombol Q pada keyboard.



Gambar 4.1 Use Case Diagram

4.3.2 Activity Diagram

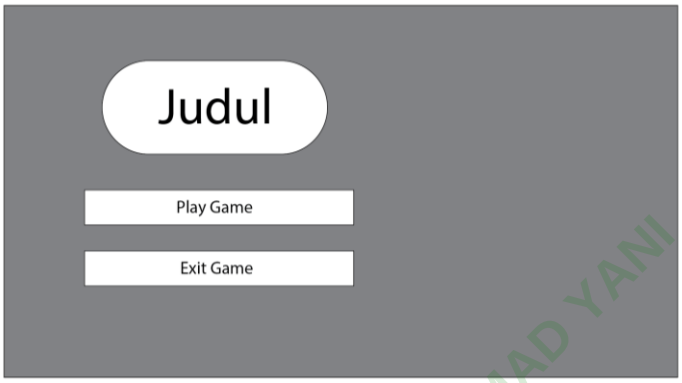
Activity diagram dapat dilihat di gambar 4.2.



Gambar 4.2 Activity Diagram *scene game*.

4.3.3 Gaya Tampilan

Berikut tampilan awal dari *game* Out Of Sight. Dibuat hanya untuk gambaran awal pada *menu main menu*, gaya *gameplay* dan *scene gameover*. Tampilan awal *scene main menu* ditunjukkan pada gambar 4.3 di bawah ini.



Gambar 4.3 Tampilan Awal Main Menu.

Gaya gameplay yang ingin dibangun untuk *game* ini menggunakan pandangan *first person*. Contoh pandangan *first person* ditunjukkan pada gambar 4.4 di bawah ini.

Gambar 4.4 *First Person Gameplay*

Tampilan awal pada *scene game over* hanya menggunakan tulisan dan button *back to menu* yang menghubungkan pemain ke menu utama. Tampilan awal pada *scene game over* ditunjukkan pada gambar 4.5 dibawah ini.

Gambar 4.5 Tampilan Awal *Scene Game Over*.

4.3.4 Material Collecting

Beberapa komponen yang perlu di kumpulkan adalah semua asset yang diperlukan untuk game ini. Misalkan, *zombie*, pistol, pintu, lampu, sound effect, dll. Agar game ini terlihat realistis dan menarik. Untuk bahan model asset bisa di unduh di website resmi unity assets store. Jika sound effect bisa diunduh di platform YouTube.

4.4 *MATERIAL COLLECTING*

Pada tahapan ini, materi atau asset yang digunakan untuk membuat game di-*download* dari website Unity assets store dan YouTube secara gratis. Berikut materi atau asset yang digunakan dalam game ini:

4.4.1 Zombie

Zombie pada game ini berperan sebagai musuh utama. Asset ini dibuat dan diunggah oleh Pxltiger di website resmi Unity assets store. Model *zombie* yang digunakan ditunjukkan pada gambar 4.6.

Gambar 4.6 Model Zombie.

4.4.2 Pistol

Pistol akan ditemukan oleh pemain di lorong tertentu. Pistol ini berfungsi untuk melawan para *zombie* yang nanti akan muncul di dalam game. Model pistol ini dibuat dan diunggah oleh Nokobot di Unity assets store. Model pistol ditunjukkan pada gambar 4.7.

Gambar 4.7 Model senjata dari Unity assets store.

4.4.3 Mouldings

Untuk penyempurnaan desain arena agar terlihat lebih natural dan seperti lorong rumah pada umumnya, maka dibutuhkan *moulding* pada setiap pojok dinding. Model ini dibuat dan diunggah ke website resmi Unity assets store oleh Ferocious Industries. Model *mouldings* ditunjukkan pada gambar 4.8.

Gambar 4.8 Model *mouldings* dari Unity assets store.

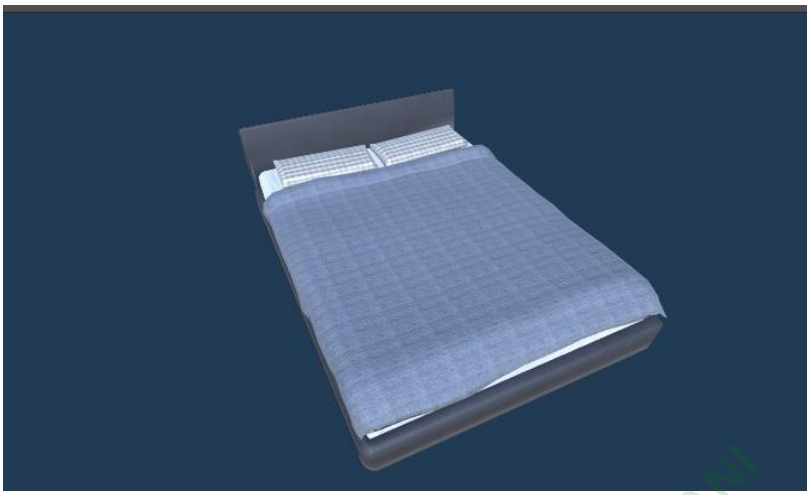
4.4.4 Kabinet kuno

Untuk mendukung suasana arena agar terlihat kuno dan menyeramkan, maka diperlukan model objek kabinet kuno yang akan diletakkan di dalam game. Kabinet tua ini dibuat dan diunggah di website resmi Unity assets store oleh Tatiana Gladkaya. Model kabinet kuno yang digunakan ditunjukkan pada gambar 4.9.

Gambar 4.9 *Old Cabinet* dari Unity assets store.

4.4.5 Interior

Agar desain arena di dalam game ini terlihat hidup dan tidak hanya sebuah lorong biasa, maka ada beberapa ruangan seperti kamar tidur dan kamar mandi yang diisi beberapa interior. Johnny Kasapi membuat dan menggunggahnya di website resmi Unity assets store yang berjudul *furnished cabin*. Model interior ditunjukkan pada gambar 4.10 dan 4.11.

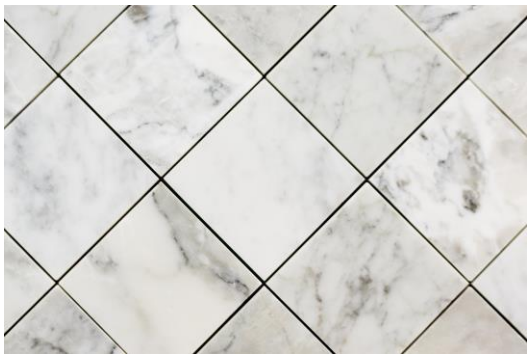


Gambar 4.10 Interior dari Unity assets store.

Gambar 4.11 Interior dari Unity assets store.

4.4.6 Texture

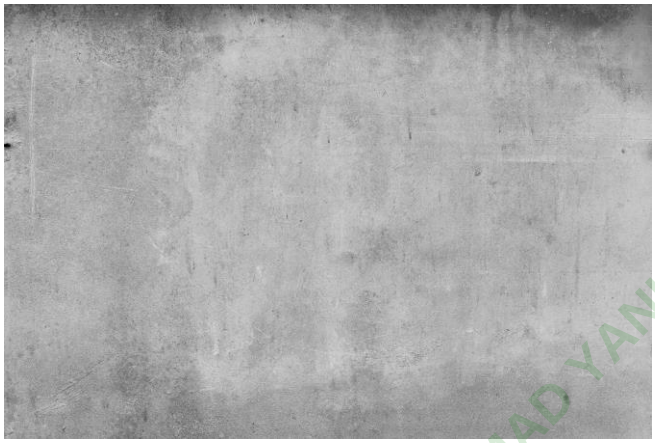
Texture di dalam game ini dibuat secara manual di dalam Unity *engine*. dibuat agar terlihat lebih realistis dengan menambahkan *normal map* ketika sedang membuat tembok maupun lantai di ruangan tertentu. *Texture* yang digunakan ditunjukkan pada gambar 4.12 dan gambar 4.13.



Gambar 4.12 *Texture* untuk lantai kamar mandi.

Gambar 4.13 *Texture* untuk dinding kamar mandi.

Berikut *texture* untuk lantai lorong, lantai kamar, dinding lorong, dinding kamar, langit – langit lorong, dan langit – langit kamar. Ditunjukkan pada gambar 4.14 dan 4.15.

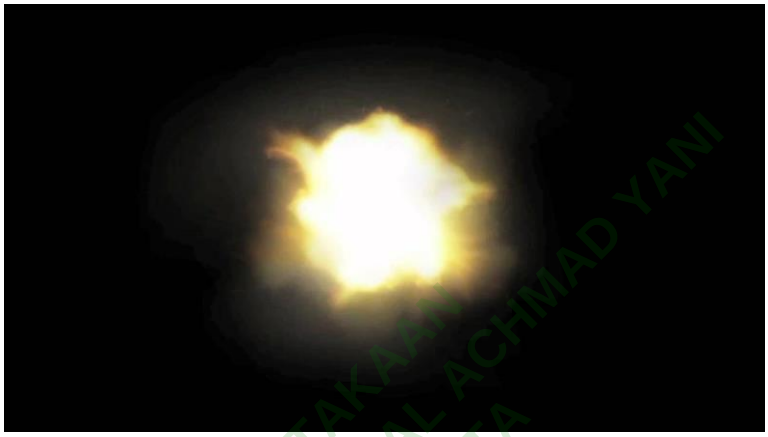


Gambar 4.14 *Texture* dinding lorong dan kamar.

Gambar 4.15 *Texture* lantai lorong dan kamar.

4.4.7 Muzzle Flash

Muzzle flash digunakan untuk memberikan kesan realistis ketika pemain menggunakan pistol yang ada di dalam game ini. *Muzzle flash* ini bisa didapatkan di https://www.youtube.com/watch?v=Kc5E9WaC4ew. Contoh *muzzle flash* ditunjukkan pada gambar 4.16.



Gambar 4.16 *Muzzle flash.*

4.4.8 Skybox

Karena tema yang gelap dan mencekam, maka perlu diberikan *skybox* agar langit pada game ini mengikuti suasana game ini. Contoh model *skybox* yang akan digunakan ditunjukkan pada gambar 4.17..

Gambar 4.17 Skybox yang didapatkan gratis di Unity assets store.

4.4.9 Kunci

Pada game ini terdapat kunci yang wajib di cari oleh para pemain untuk membuka pintu yang terkunci agar dapat pergi dari serangan *zombie*. Model ini dibuat dan diunggah di website resmi unity assets store oleh Aleksn09. Model kunci ditunjukkan pada gambar 4.18.

Gambar 4.18 Model kunci.

4.4.10 Lampu

Object lampu dibutuhkan karena agar lorong di dalam game ini terlihat seperti lorong pada umumnya. Pembuat dan yang menggunggah nya di website resmi unity assets store adalah Flatriver. Model lampu ditunjukkan pada gambar 4.19.

Gambar 4.19 Model lampu.

4.4.11 Pintu

Pintu yang dapat digunakan dan ada di dalam game ini juga berasal dari Unity assets store. Yang dibuat dan diunggah oleh Jan Fidler. Gambar model pintu ditunjukkan pada gambar 4.20.

Gambar 4.20 Model pintu.

4.4.12 Suara Wanita Menangis

Suara tangisan wanita yang akan membuat merinding para pemain. Berikut link dari sumber suara tangisan wanita https://www.youtube.com/watch?v=F3PCVIzyQeo.

4.4.13 Horror Ambience

Suara natural yang akan muncul untuk membuat para pemain merasa ada yang tidak beres di sekitar pemain. Berikut link dari sumber suara yang didapat https://www.youtube.com/watch?v=xAO3x-Uhfoo

4.4.14 Suara Ketukan Pintu

Suara ketukan yang berbunyi ketika pemain mendekati ruangan tertentu. Yang bertujuan untuk membuat terkejut pemain. Berikut link dari suara ketukan pintu https://www.youtube.com/watch?v=mBzgvb7uLyM.

4.4.15 Background Musik Horror

Background musik yang akan membuat para pemain ketakutan. Dengan didukung gelapnya lorong di game ini. Berikut link untuk background musik horror https://www.youtube.com/watch?v=h4oQ5CTu_cE.

4.4.16 Suara Pintu Terbuka

Suara pintu terbuka yang berbunyi ketika pemain membuka pintu. Berikut link suara pintu terbuka https://www.youtube.com/watch?v=92YfzPu3Ukc.

4.4.17 Suara Item

Suara item ini berfungsi ketika pemain mengambil item (clue) yang terdapat pada game ini. Berikut link untuk suara mengambil item https://www.youtube.com/watch?v=CPBnq75A_ls.

4.4.18 Suara Hujan

Suara hujan berfungsi membuat keadaan arena menjadi penuh dengan kengerian. Berikut link untuk suara hujan https://www.youtube.com/watch?v=nzCns4EY15c.

4.4.19 Suara Spotlight

Pada lorong tertentu terdapat beberapa spotlight yang akan menyala. Ketika pemain melewati spotlight tersebut maka akan terdengar suara spotlight yang menyala. Berikut link untuk suara spotlight https://www.youtube.com/watch?v=uP_NV2RNJiU.

4.4.20 Suara Ketukan Pintu Keras

Suara ini akan menjadi salah satu *jumpscare* pada game ini. Berikut link untuk suara ketukan pintu secara keras https://www.youtube.com/watch?v=Y3UxNtEUlcs.

4.5 ASSEMBLY

4.5.1 Pembuatan Gambar

Pada tahap ini pembuatan semua objek atau bahan multimedia mulai dikerjakan. Seperti perancangan antarmuka, storyboard, dan struktur navigasi *game* ini. Desain antarmuka pada game ini adalah sebuah lorong gelap yang tidak ada hentinya, lorong tersebut dapat dilihat pada gambar 4.21.

Gambar 4.21 Desain lorong.

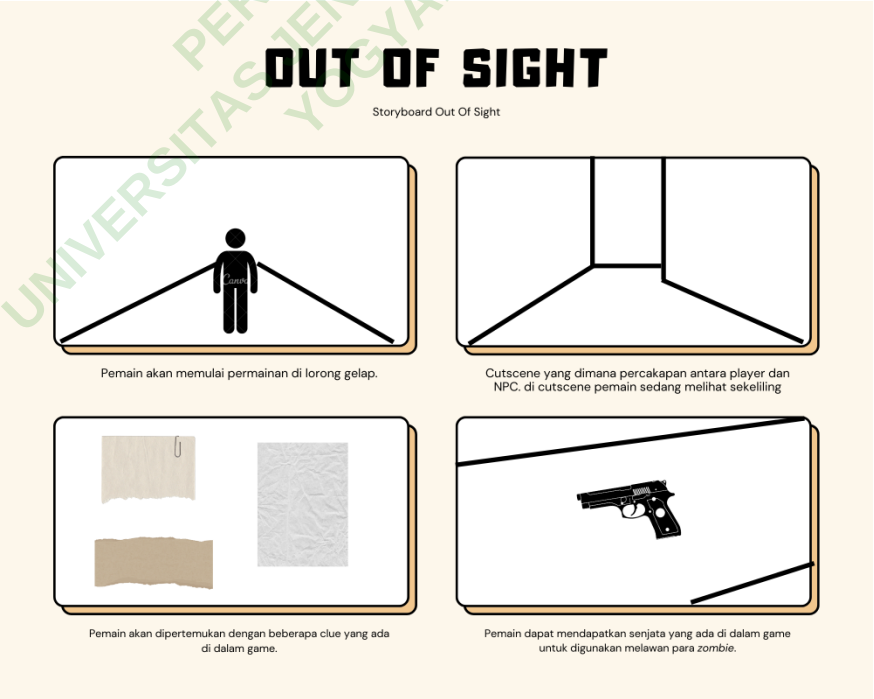
Lorong tanpa henti yang minim cahaya ini memang sengaja dibuat agar memunculkan rasa tidak beres kepada para pemain. Gambar desain lorong lainnya terlihat seperti pada gambar 4.22.

Gambar 4.22 Desain lorong lainnya.

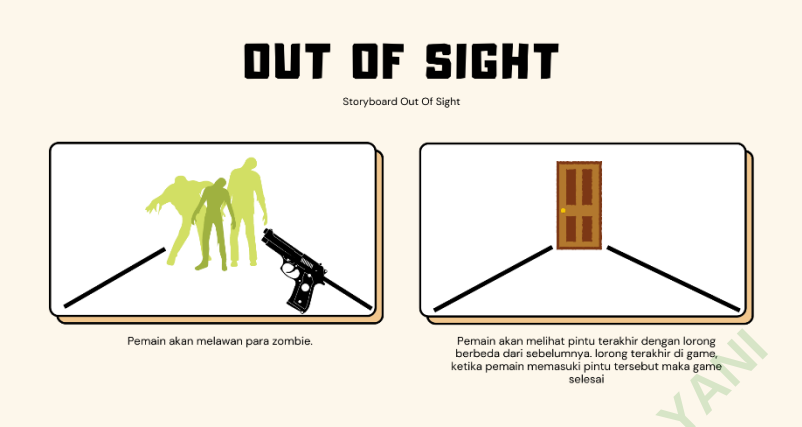
Ruangan seperti kamar mandi juga di desain agar terlihat seperti kamar mandi sungguhan dengan cahaya yang minim. Desain kamar mandi dapat terlihat pada gambar 4.23.

Gambar 4.23 Desain kamar mandi.

Storyboard yang dibuat terlihat pada gambar 4.24 dan gambar 4.25.



Gambar 4.24 Storyboard game 1.



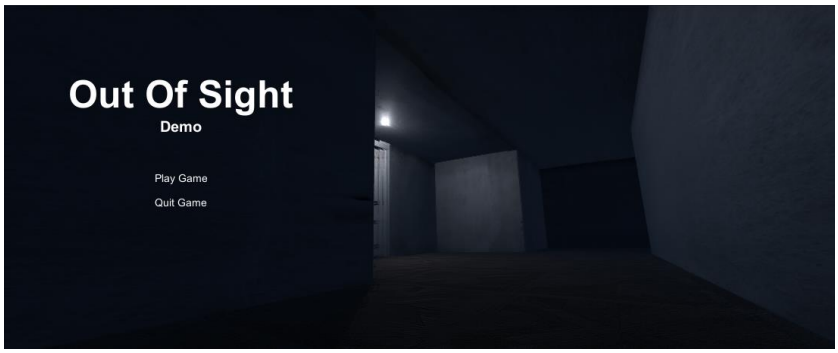
Gambar 4.25 Storyboard game 2.

4.5.2 Pembuatan Game

Tools yang digunakan dalam pembuatan *game* ini adalah unity 2019 dan *script*nya menggunakan bahasa C#. Dalam *game* ini terdapat 3 scene yaitu: *MainMenu*, *Scene Game*, dan *Scene END*. Berikut ini tahap pembuatan *game* Out Of Sight:

1. *MainMenu*

   Pada *MainMenu* hanya terdapat camera, desain lorong, *button*, lampu, audio, dan *game tittle*. Di dalam *mainmenu* terdapat *background music* dan dua tombol yaitu *playgame button*, dan *quit game button*. *Playgame button* digunakan untuk menuju pada *scene game*, sedangkan *quit game button* digunakan untuk keluar dari *game*. *MainMenu* dapat dilihat pada gambar 4.26 di bawah.



Gambar 4.26 *MainMenu*

Kode *scene management* yang digunakan ketika pemain menekan tombol *play game* maka akan masuk ke scene *game*, ketika pemain menekan tombol *quit game* maka *game* akan ter-*close*.

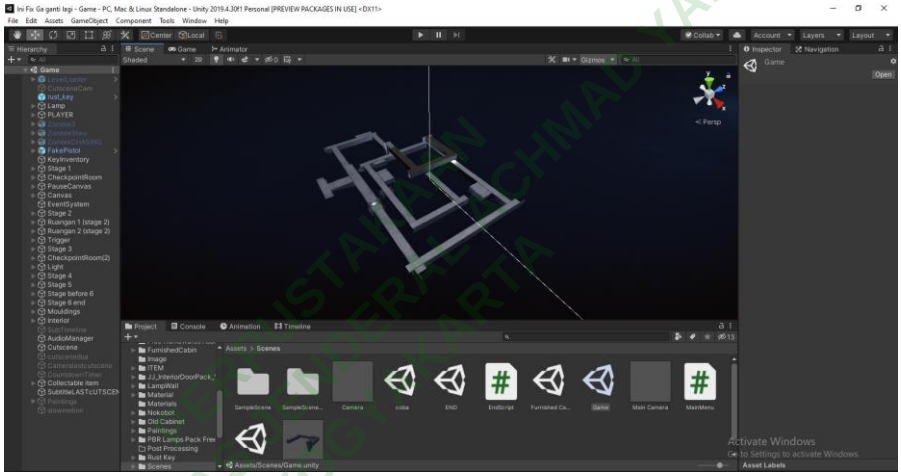
MainMenu.cs

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

public class MainMenu : MonoBehaviour
{
    public void ButtonStart()
    {
        SceneManager.LoadScene(1);
    }
    public void ButtonQuit()
    {
        Application.Quit();
    }

}
```

Gambar 4.27 Kode *MainMenu.*

2. *Scene Game*

Di dalam *scene game* terdiri dari *player*, zombie, *stage 1* hingga *stage 6*, *canvas*, beberapa *gameobject* berupa interior rumah, beberapa *trigger* yang tersebar di dalam *game* ini, *checkpointRoom*, audio dan beberapa item. Pada gambar 4.28 merupakan tampilan *scene game*, pada *scene* inilah objek yang dibutuhkan akan dijalankan. Sehingga pembuatan *scene* ini melalui tahap pembuatan semua seperti, membangun lorong, *camera*, *gameobject*, *texture*, *light*, *trigger* yang berupa *jumpscare*, aktifnya audio, dan *trigger* untuk mengambil item.



Gambar 4.28 *Scene Game*

Kode untuk player agar dapat berjalan kanan dan kiri. Pada *gameobject* player terdapat juga kode *player casting*. Yang dimana ketika pemain mendekati pintu, item, atau senjata mereka dapat menekan tombol *action* (E).

C# PlayerMovement.cs

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

public class PlayerMovement : MonoBehaviour
{
    public CharacterController controller;
    public GameObject slowmotionAbility;
    [SerializeField] private float speed = 1f;
    [SerializeField] private float gravity = -9.81f;

    Vector3 velocity;

    // Start is called before the first frame update
    void Start()
    {
        controller = GetComponent<CharacterController>();
    }

    // Update is called once per frame
    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        Vector3 move = transform.right * x + transform.forward * z;

        controller.Move(move * speed * Time.deltaTime);

        velocity.y += gravity * Time.deltaTime;

        controller.Move(velocity * Time.deltaTime);
    }
}
```

Gambar 4.29 Kode *PlayerMovement*.

```
C# PlayerCasting.cs

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

public class PlayerCasting : MonoBehaviour
{
    public static float DistanceFromTarget;
    public float ToTarget;


    // Update is called once per frame
    void Update()
    {
        RaycastHit Hit;
        if (Physics.Raycast(transform.position, transform.TransformDirection(Vector3.forward), out Hit))
        {
            ToTarget = Hit.distance;
            DistanceFromTarget = ToTarget;
        }
    }
}
```

Gambar 4.30 Kode *PlayerCasting.*

Kode yang diletakkan pada *gameobject camera*. Berguna untuk melihat sekitar menggunakan mouse.

```
C# MouseLook.cs

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

public class MouseLook : MonoBehaviour
{

    [SerializeField]
    private float Mousesensitivity = 100f;

    public Transform playerBody;

    float xRotation = 0f;

    // Start is called before the first frame update
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked;
    }

    // Update is called once per frame
    void Update()
    {
        float mouseX = Input.GetAxisRaw("Mouse X") * Mousesensitivity * Time.deltaTime;
        float mouseY = Input.GetAxisRaw("Mouse Y") * Mousesensitivity * Time.deltaTime;

        xRotation -= mouseY;
        xRotation = Mathf.Clamp(xRotation, -90f, 90f);

        transform.localRotation = Quaternion.Euler(xRotation, 0f, 0f);
        playerBody.Rotate(Vector3.up * mouseX);
    }
}
```

Gambar 4.31 Kode *MouseLook.*

Salah satu kode *trigger* yang digunakan untuk memunculkan *jumpscare*, zombie, lampu yang pada awalnya mati dan *ambience* yang mendukung suasana di dalam *game*. *Trigger jumpscare* berupa audio dan *ambience* sama, hanya berbeda di atribut.

```
C# triggerFX.cs

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

public class triggerFX : MonoBehaviour
{
    [SerializeField] private AudioSource playSound;
    [SerializeField] private AudioSource jumpscare;
    [SerializeField] private AudioSource bassAmbience;



    private void OnTriggerEnter(Collider other)
    {
        playSound.Play();
        jumpscare.Play();
        bassAmbience.Play();
        this.GetComponent<BoxCollider>().enabled = false;
    }
}
```

Gambar 4.32 Kode *trigger jumpscare*.

EnemyShowTrigger.cs

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

public class EnemyShowTrigger : MonoBehaviour
{

    [SerializeField] private GameObject enemy;
    [SerializeField] private GameObject enemytwo;
    [SerializeField] private AudioSource enemymusic;
    [SerializeField] private GameObject objectivetwo;

    private void OnTriggerEnter(Collider other)
    {
        enemy.SetActive(true);
        enemytwo.SetActive(true);
        enemymusic.Play();
        objectivetwo.SetActive(true);
        Destroy(gameObject);
    }
}
```

Gambar 4.33 Kode *EnemyShowTrigger*.

LightAutomatic.cs

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

public class LightAutomatic : MonoBehaviour
{
    [SerializeField] private GameObject redlight;
    [SerializeField] private AudioSource lampunyala;


    private void OnTriggerEnter(Collider other)
    {
        redlight.GetComponent<Light>().enabled = true;
        lampunyala.Play();
        this.GetComponent<BoxCollider>().enabled = false;
    }
}
```

Gambar 4.34 Kode *LightAutomatic*.

Kode *trigger* yang digunakan agar pemain dapat membuka pintu, mengambil senjata, dan mengambil item (kertas, kunci).

```
DoorKey.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DoorKey : MonoBehaviour
{
    public float TheDistance;
    public GameObject ActionDisplay;
    public GameObject ActionText;
    public GameObject crosshair;
    public GameObject theKey;

    // Update is called once per frame
    void Update()
    {
        TheDistance = PlayerCasting.DistanceFromTarget;
    }

    void OnMouseOver()
    {
        if (TheDistance <= 1.8)
        {
            ActionDisplay.SetActive(true);
            ActionText.SetActive(true);
            crosshair.SetActive(false);
        }
        if (Input.GetButtonDown("Action"))
        {
            if (TheDistance <= 1.8)
            {
                this.GetComponent<BoxCollider>().enabled = false;
                ActionDisplay.SetActive(false);
                ActionText.SetActive(false);
                theKey.SetActive(false);
                GlobalInventory.firstDoorKey = true;
            }
        }
    }

    private void OnMouseExit()
    {
        ActionDisplay.SetActive(false);
        ActionText.SetActive(false);
        crosshair.SetActive(true);
    }
}
```

Gambar 4.35 Kode *DoorKey*.

PickUpPistol.cs

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

public class PickUpPistol : MonoBehaviour
{
    public float TheDistance;
    public GameObject ActionDisplay;
    public GameObject pickupthegun;
    public GameObject FakePistol;
    public GameObject RealPistol;

    // Update is called once per frame
    void Update()
    {
        TheDistance = PlayerCasting.DistanceFromTarget;
    }

    void OnMouseOver()
    {
        if (TheDistance <= 4)
        {
            pickupthegun.GetComponent<Text>().text = "Pick Up Gun";
            ActionDisplay.SetActive(true);
            pickupthegun.SetActive(true);

        }
        if (Input.GetButtonDown("Action"))
        {
            if (TheDistance <= 4)
            {
                this.GetComponent<BoxCollider>().enabled = false;
                ActionDisplay.SetActive(false);
                pickupthegun.SetActive(false);
                FakePistol.SetActive(false);
                RealPistol.SetActive(true);
            }
        }
    }

    private void OnMouseExit()
    {
        ActionDisplay.SetActive(false);
        pickupthegun.SetActive(false);
    }
}
```

Gambar 4.36 Kode *PickUpPistol*.

C# COLLECTABLEITEM.cs

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

public class COLLECTABLEITEM : MonoBehaviour
{
    [SerializeField] private float itemdistance;
    [SerializeField] private GameObject itemActionDisplay;
    [SerializeField] private GameObject itemActionText;
    [SerializeField] private GameObject Item;
    [SerializeField] private GameObject crosshair;
    [SerializeField] private AudioSource collectItem;
    [SerializeField] private GameObject pausebar;
    [SerializeField] private GameObject fakeitem;
    [SerializeField] private GameObject player;
    [SerializeField] private GameObject cameraplayer;



    void Update()
    {
        itemdistance = PlayerCasting.DistanceFromTarget;
    }

    void OnMouseOver()
    {
        if (itemdistance <= 1.5)
        {
            itemActionDisplay.SetActive(true);
            itemActionText.SetActive(true);
            crosshair.SetActive(false);
        }
        if (Input.GetButtonDown("Action"))
        {
            if (itemdistance <= 1.5)
            {
                this.GetComponent<BoxCollider>().enabled = false;
                player.GetComponent<PlayerMovement>().enabled = false;
                cameraplayer.GetComponent<MouseLook>().enabled = false;
                Destroy(fakeitem);
                itemActionDisplay.SetActive(false);
                itemActionText.SetActive(false);
                collectItem.Play();
                Item.SetActive(true);
                pausebar.SetActive(true);
                StartCoroutine(itemExit());
            }
        }
    }

    IEnumerator itemExit()
    {
        yield return new WaitForSeconds(3);
        player.GetComponent<PlayerMovement>().enabled = true;
        cameraplayer.GetComponent<MouseLook>().enabled = true;
        Destroy(Item);
        itemActionDisplay.SetActive(false);
        itemActionText.SetActive(false);
        pausebar.SetActive(false);
        crosshair.SetActive(true);
    }
}
```

Gambar 4.37 Kode *COLLECTABLEITEM*.

Kode zombie yang diletakkan pada *gameobject* zombie model. Agar zombie dapat bergerak, mengejar *player*, dan menyerang.

EnemyMovement.cs

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

public class EnemyMovement : MonoBehaviour
{

    NavMeshAgent _navMeshAgent;
    [SerializeField] private float _attackRange = 2f;
    [SerializeField] private float health = 50f;
    public Animator _anim;
    bool Death = false;

    // Start is called before the first frame update
    void Awake()
    {
        _anim = GetComponent<Animator>();
        _navMeshAgent = GetComponent<NavMeshAgent>();
    }

    // Update is called once per frame
    void Update()
    {

        var player = FindObjectOfType<PlayerMovement>();
        _navMeshAgent.SetDestination(player.transform.position);

        if (Vector3.Distance(transform.position, player.transform.position) < _attackRange && !Death)
        {
            _anim.SetTrigger("Attack");
        }

    }

    public void TakeDamage(float damageAmount)
    {
        health -= damageAmount;
        if(health <= 0)
        {
            _anim.SetTrigger("Death");
            GetComponent<CapsuleCollider>().enabled = false;
            Destroy(gameObject, 10);
            Death = true;
            _navMeshAgent.Stop();
        }
    }


}
```

Gambar 4.38 Kode *EnemyMovement*.

Di dalam *game* terdapat angka – angka yang tersebar. Pemain diharuskan mengingat agar dapat membuka pintu yang terkunci oleh kombinasi angka. Berikut kode untuk kombinasi angka yang ada di dalam *game* ini.

keypad.cs

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

public class keypad : MonoBehaviour
{
    [SerializeField] private GameObject door;
    [SerializeField] private AudioSource CreakSound;
    [SerializeField] private Text Ans;
    [SerializeField] private GameObject cameraplayer;
    [SerializeField] private GameObject keypadUI;
    private string Answer = "7789";


    public void Number(int number)
    {
        Ans.text += number.ToString();
    }

    public void Execute()
    {
        if(Ans.text == Answer)
        {
            Ans.text = "Correct";
            door.GetComponent<Animation>().Play("DoorHinge");
            CreakSound.Play();
            StartCoroutine(keypadHilang());
        }
        else
        {
            Ans.text = "Invalid";
            StartCoroutine(textHilang());
        }
    }

    IEnumerator textHilang()
    {
        yield return new WaitForSeconds(2);
        cameraplayer.GetComponent<MouseLook>().enabled = true;
        keypadUI.SetActive(false);
    }

    IEnumerator keypadHilang()
    {
        yield return new WaitForSeconds(1);
        cameraplayer.GetComponent<MouseLook>().enabled = true;
        keypadUI.SetActive(false);
        Cursor.visible = false;
    }
}
```

Gambar 4.39 Kode Kombinasi Angka.

Di dalam game pemain memiliki kemampuan untuk memperlambat waktu. Berikut kode untuk memperlambat waktu.

slowmotion.cs

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

public class slowmotion : MonoBehaviour
{
    public float slowTime = 0.2f;
    private bool isSlowed = false;


    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("slowmotion"))
        {
            if (isSlowed)
            {
                isSlowed = false;
                Time.timeScale = 1f;
                Time.fixedDeltaTime = Time.deltaTime;
            }
            else
            {
                isSlowed = true;
                Time.timeScale = slowTime;
                Time.fixedDeltaTime = slowTime * Time.deltaTime;
            }
        }
    }
}
```

Gambar 4.40 Kode Memperlambat Waktu.

Pemain dapat menggunakan fitur *pause* yang berguna ketika pemain ingin jeda sebentar dari *game*. Berikut kode *pause*.

```
C# pausemenu.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{

    public static bool Paused = false;
    public GameObject pausemenuUI;
    public GameObject crosshair;



    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Paused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pausemenuUI.SetActive(false);
        Time.timeScale = 1f;
        Paused = false;
        crosshair.SetActive(true);
    }

    void Pause()
    {
        pausemenuUI.SetActive(true);
        Time.timeScale = 0f;
        Paused = true;
        crosshair.SetActive(false);
    }
}
```

Gambar 4.41 Kode *pause*.

3. *Scene END*

Di dalam *scene gameover* terdiri dari *camera, directional light, canvas,* dan *button.* Canvas digunakan untuk meletakkan text "Thank you for playing" dan *button*. Ketika pemain menekan *button back to menu*, maka pemain akan kembali ke *mainmenu*. *Scene END* dapat dilihat pada gambar 4.43 di bawah.

Gambar 4.42 *Scene END*

Kode pada *scene END* yang dimana ketika pemain selesai memainkan game, pemain dapat kembali ke menu utama dengan cara menekan tombol *Back to menu*. Berikut kode *scene END*.

```
EndScript.cs

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class EndScript : MonoBehaviour
{

    private void Start()
    {
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
    }

    public void backtomenu()
    {
        SceneManager.LoadScene(0);
    }
}
```

Gambar 4.43 Kode *EndScript*.

4.6 *TESTING*

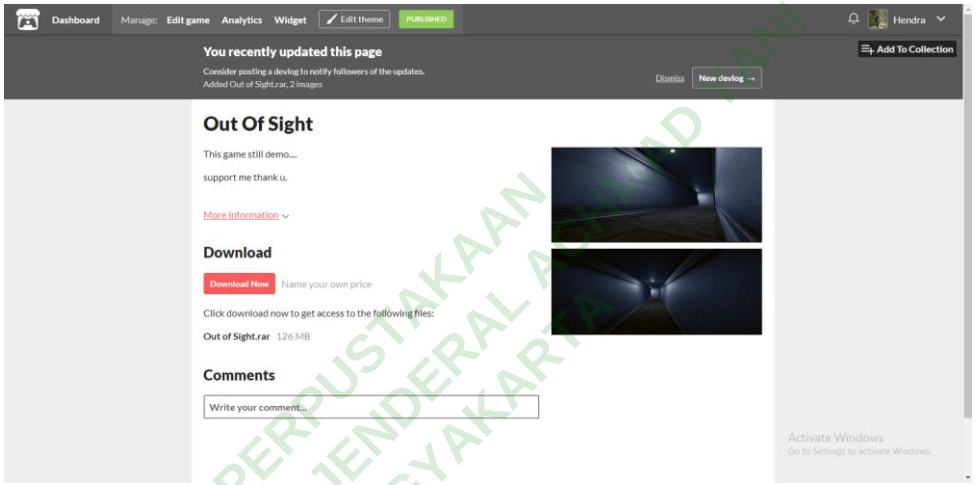
Metode pengujian dilakukan menggunakan metode *black box*. Pengujian ini dilakukan berfokus pada fungsional dari program. Sehingga dengan pengujian ini dapat mendefinisikan kumpulan kondisi input dan melakukan pengujian pada spesifikasi fungsional dari program. Hasil pengujian *game* dengan metode *black box* dapat dilihat pada table berikut ini:

No	**Skenario Pengujian**	**Hasil yang diharapkan**	**Hasil**
1	UI dan Button	UI dan Button berfungsi	Valid
2	Menggerakan karakter	Karakter dapat bergerak	Valid
3	Melihat sekitar area	Karakter dapat melihat sekitar area	Valid
4	Mengambil senjata	Player dapat mengambil senjata	Valid
5	Mengambil item (clue)	Player dapat mengambil item (clue)	Valid
6	Menembak menggunakan senjata	Player dapat menembak menggunakan senjata yang ada	Valid
7	Membuka pintu	Player dapat membuka pintu, dan pintu terbuka	Valid
8	Zombie bergerak	Zombie dapat bergerak	Valid
9	Zombie menyerang	Zombie dapat menyerang player	Valid
10	Pintu kombinasi	Pintu kombinasi berfungsi	Valid
11	Pintu terkunci dan kunci	Pintu dapat dibuka ketika sudah mendapatkan kunci	Valid

Tabel 4.1 Tabel *black box* testing.

4.7 DISTRIBUTION

Game yang telah dibuat dan telah melewati uji coba kemudian dipublikasikan dan didistribusikan melalui website itch.io, website untuk para developer *game* indie untuk mempublikasikan *game* nya. *Link* pengunduhan di promosikan melalui media sosial yaitu instagram dengan tujuan agar lebih banyak pengguna yang dijangkau. Berikut adalah alamat pengunduhan pada website itch.io https://hendra0010.itch.io/out-of-sight dan gambar di bawah ini merupakan hasil publikasi melalui website itch.io.



Gambar 4.44 Publikasi *Game* Melalui Website itch.io.