

LAPORAN TUGAS
Untuk Memenuhi Nilai Ujian Akhir Semester
Mata Kuliah Grafika Komputer Semester Genap Tahun Ajaran 2022



Disusun oleh:

Ivan Chandra C14200119

Ivan Vincent Kwenandar C14200138

Sebastian Jonathan C14200157

Dosen:

ANDHIKA EVANTIA IRAWAN, S.Kom.

LILIANA, S.T., M.Eng., Ph.D.

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS KRISTEN PETRA
SURABAYA**

DAFTAR ISI

DAFTAR ISI	i
PENDAHULUAN	ii
ALUR CERITA	iii
HASIL AKHIR	4
1. CAMERA	4
1.1 Third Person Camera	4
1.2 First Person Camera	6
1.3 CCTV	6
1.4 Freddy Point of View (PoV)	8
1.5 Switch Character	9
2. LIGHT	11
2.1 Office	11
2.2 Hallway Kanan	11
2.3 Hallway Kiri	12
2.4 Room 1	13
2.5 Room 2	13
2.6 Gudang	14
2.7 Hall	15
2.8 Stage	16
2.9 Senter	16
3. COLLISION HANDLER	17
4. MODEL / OBJECT	19
DESAIN	21
PENUTUP	23

PENDAHULUAN

Pada Ujian Akhir Semester (UAS) ini kami mendapatkan proyek dengan tujuan utama untuk membuat sebuah sket yang mempunyai kombinasi *lighting*, *camera*, dan *collision handler*. Kemudian dilakukan implementasi dari sket tersebut menggunakan bahasa pemrograman (C# → C-Sharp) dan penggabungan tiap komponen dari masing-masing fitur.

Proyek ini kami buat menggunakan *Visual Studio* 2022 dan memanfaatkan penggunaan *library* OpenTK yang merupakan kumpulan *binding* C# tingkat rendah yang cepat, portabel, untuk OpenGL, OpenGL ES, OpenAL, dan OpenCL. OpenTK dapat digunakan di semua *platform* utama dan mendukung ratusan aplikasi, game, dan program penelitian ilmiah. Untuk proses desain objek, kami menggunakan *Blender* sebagai alat bantu yang mempermudah proses penggerjaan kami. Tema dari proyek yang kami buat adalah “***Five Night At Freddy (FNAF)***”.

ALUR CERITA

Cerita ini berlatar pada restoran *Freddy Fazbear Family Diner* dimana anda adalah seorang penjaga malam baru yang sedang melakukan pengawasan di restoran tersebut. Restoran ini memiliki beberapa ruangan dan sebuah ruang utama yang difungsikan sebagai tempat berpesta. Pada awalnya, anda dengan santai berdiam diri di kantor (*Office*). Tiba-tiba anda menerima telepon dari seorang misterius yang memberikan pesan singkat mengenai restoran ini kepada anda. Setelah percakapan di telepon selesai, anda mengecek CCTV yang dimana awalnya tidak ada hal yang aneh namun beberapa saat kemudian *Freddy* menghilang dari panggung setelah lampu berkedip beberapa kali.

Anda yang panik pun mengingat kembali pesan yang disampaikan oleh pria misterius di telepon tadi. Anda bergegas mengambil senter dan melakukan eksplorasi untuk mengecek dan mencari kemana *Freddy* pergi. Anda keluar dari kantor anda dan melakukan penyelidikan ke ruang utama. Anda mengecek ke daerah meja-meja makan dan gudang di samping panggung namun tidak menemukan petunjuk apa-apa. Tiba-tiba semua lampu mati dan terdengar sebuah musik. Anda dengan panik dan ketakutan kembali ke Office. Kemudian, lampu tiba-tiba menyala kembali. Anda merasa lega. Secara tiba-tiba *Freddy* mengejutkanmu dari belakang.

HASIL AKHIR

1. Camera

1.1 Third Person Camera

Pada implementasi *Third Person Camera* ini, kelompok kami melakukan rotasi kamera dengan menggunakan *pivot* karakter. Kamera ini menggunakan implementasi input delta mouse dimana terdapat delta X berupa input yang melakukan rotasi berdasarkan sumbu *yaw* dari karakter, sedangkan input delta Y akan melakukan translasi kamera berdasarkan sumbu *y/up*. Derajat pada delta Y akan selalu dibatasi dengan variabel *topAngle* dan *bottomAngle*. Kemudian agar kamera selalu menghadap karakter, *front* kamera akan selalu diupdate terhadap posisi karakter.

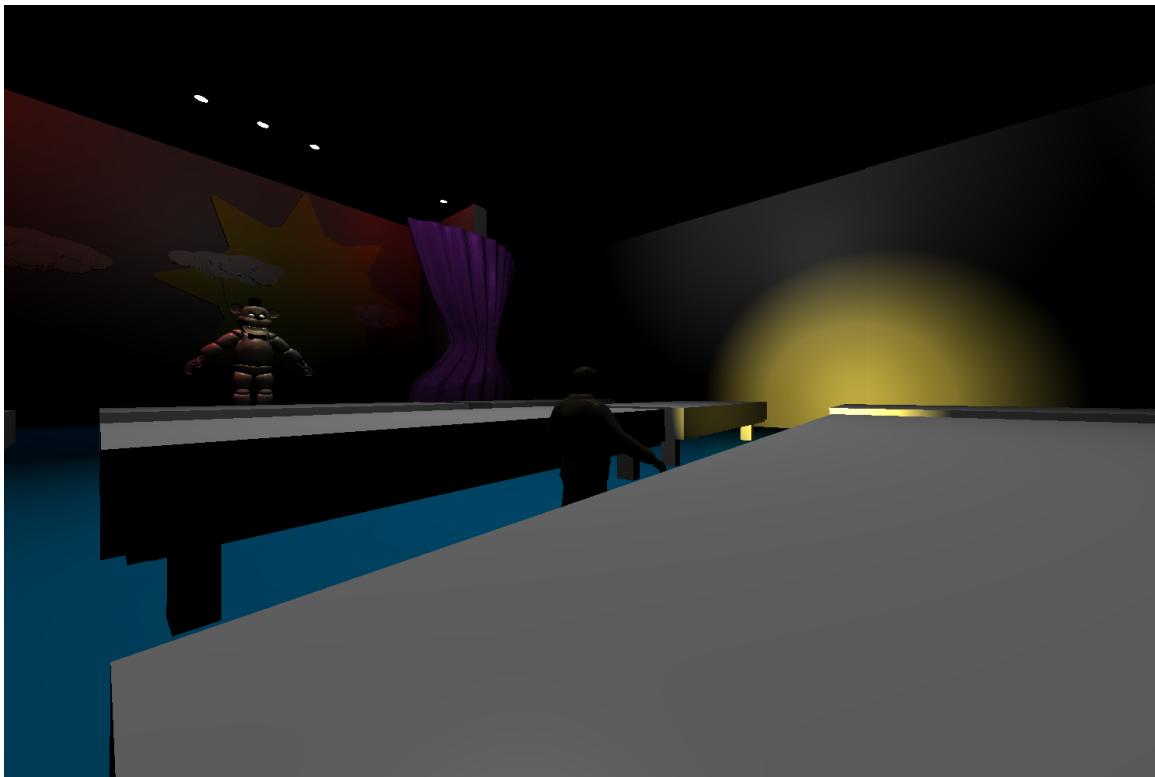
```
if (yTotal >= topAngle)
{
    if (deltaY > 0)
        yInput = 0;
    else if (deltaY < 0)
        yInput = deltaY;
}
else if (yTotal <= bottomAngle)
{
    if (deltaY < 0)
        yInput = 0;
    else if (deltaY > 0)
        yInput = deltaY;
}
else //if (yTotal > -clampAngle && yTotal < clampAngle)
{
    yInput = deltaY;
}

yTotal += yInput;

|
_camera.Position -= ObjectCenter;
_camera.Position = Vector3.Transform(_camera.Position,
    generateArbRotationMatrix(_euler[1], ObjectCenter, deltaX, true).ExtractRotation());
_camera.Position += _camera.Up * (yInput / 2);
_camera.Position += Vector3.Normalize(Vector3.Cross(_camera.Up, _camera.Right)) * (float)args.Time * (yInput);

if (_canMove)
{
    _camera.Position += dir * charaSpeed * (float)args.Time;
}

_camera.Position += ObjectCenter;
```



Pergerakan terhadap karakter dimulai dengan menggunakan input *Vector2*, input-input ini kemudian akan dikalkulasikan menggunakan rumus atan2 (x ,y) kamera. Ini bertujuan untuk menghitung derajat yang akan dirotasikan kepada karakternya agar selalu sesuai dengan arah input. Derajat putar ini kemudian akan ditambah dengan derajat yaw terhadap kamera, agar arah input karakter selalu mengarah terhadap *front* kamera.

```
{  
    var targetAngle = MathHelper.RadiansToDegrees(MathF.Atan2(Dir.X, Dir.Z)) - tps.xTotal;  
    if (targetAngle != lastAngle)  
    {  
        var HasilAngle = targetAngle - lastAngle;  
        //chara.rotate(chara.objectCenter,chara._euler[1], HasilAngle);  
        _chara.rotate(_chara.chara.objectCenter, _chara.chara._euler[1], HasilAngle);  
    }  
    lastAngle = targetAngle;  
    //Console.WriteLine(camera.Yaw);  
    MoveDir = Quaternion.FromEulerAngles(0, MathHelper.DegreesToRadians(targetAngle), 0) * -Vector3.UnitZ;  
    _chara.move(MoveDir * charaSpeed * (float)args.Time, objectList);  
}
```

Adapun input keyboard pergerakan karakter dapat diakses dengan tombol:

- W → Maju
- A → Kiri
- S → Mundur
- D → Kanan

T → beralih ke first person

1.2 First Person Camera

Pada implementasi First Person Camera, rotasi dilakukan melalui poros *yaw* dan *pitch* kamera. Untuk menggerakkan kamera ke depan/belakang akan membutuhkan *cross product* dari *up* dan *right* kamera, sedangkan menggerakkan kamera ke kiri/kanan akan membutuhkan *right* dari kamera. Adapun kita dapat beralih ke *third person* kembali dengan menekan tombol Y.

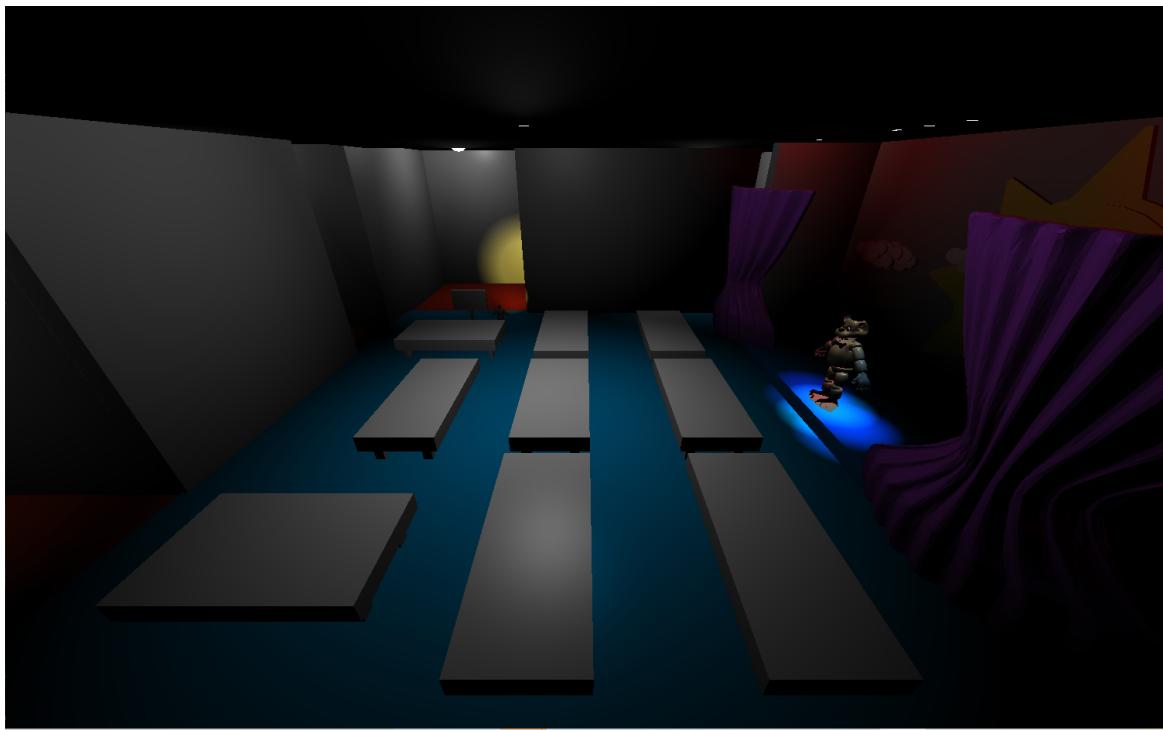
1.3 CCTV

CCTV menggunakan implementasi camera yang posisinya di *fixed* pada suatu titik (x, y, z) dan dibatasi pergerakan yang dapat dicapai dengan *mouse*. Pada hasil akhir terdapat 4 CCTV yang diletakkan pada 4 titik yang berbeda dan dapat diakses dengan menekan tombol:

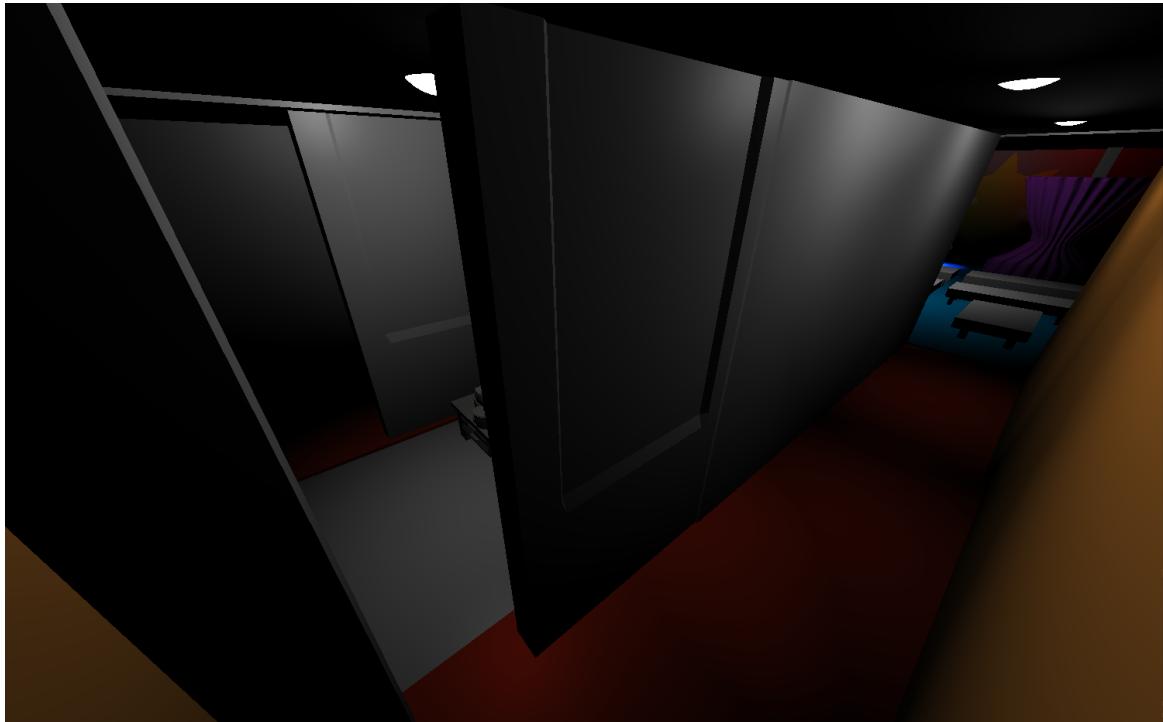
H → Gudang (*Storage*)



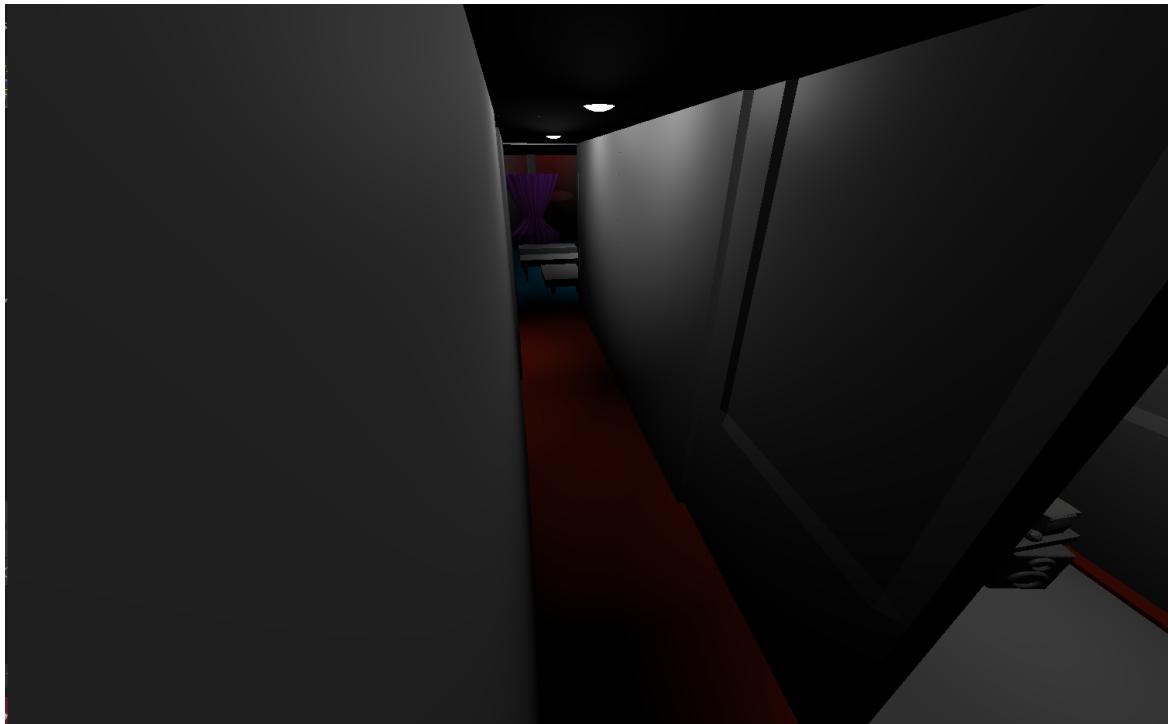
J → Hall (Ruang utama dan Panggung)



K → *Hallway Kanan*

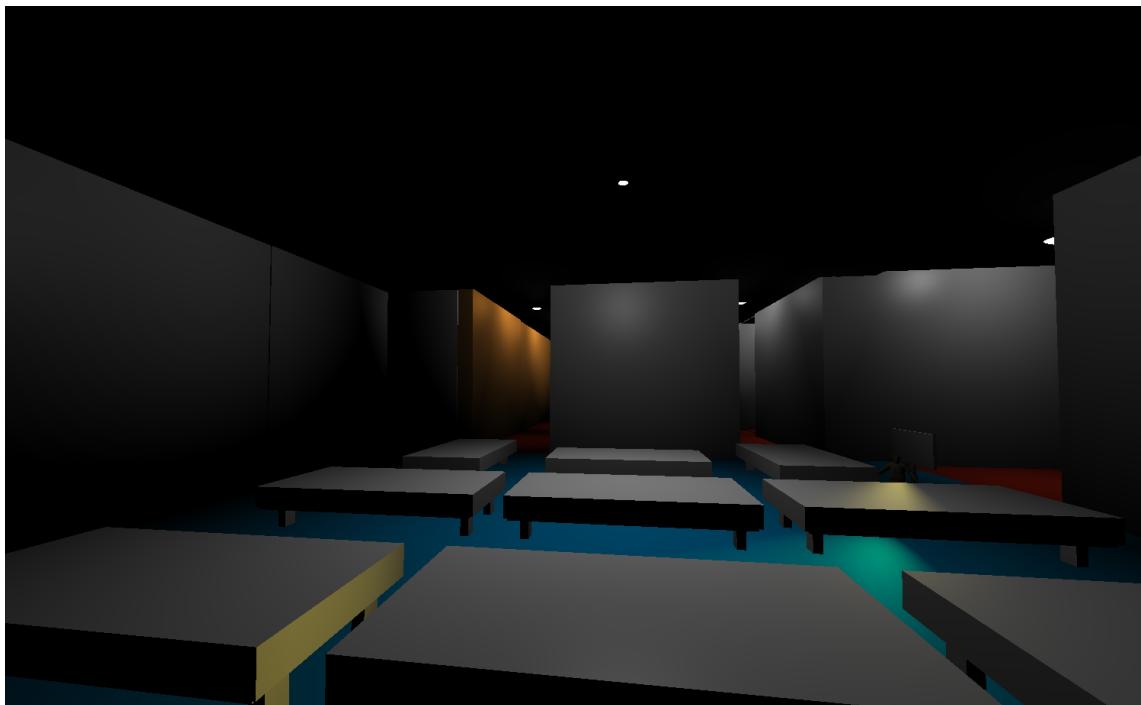


L → *Hallway Kiri*



1.4 *Freddy Point of View (PoV)*

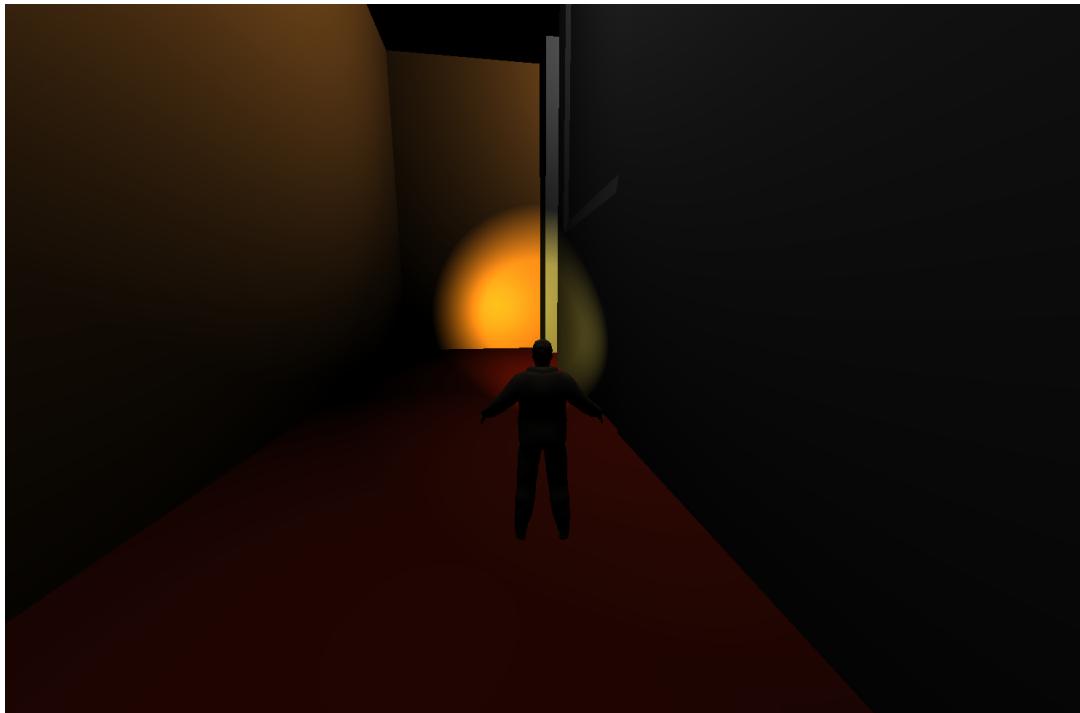
Freddy PoV menggunakan implementasi camera yang posisinya di *fixed* pada suatu titik (x, y, z) dan dibatasi pergerakan yang dapat dicapai dengan *mouse* dengan variabel Pada hasil akhir terdapat camera yang diletakkan pada 1 titik (*Freddy* di panggung) yang dapat diakses dengan menekan tombol angka ‘0’.



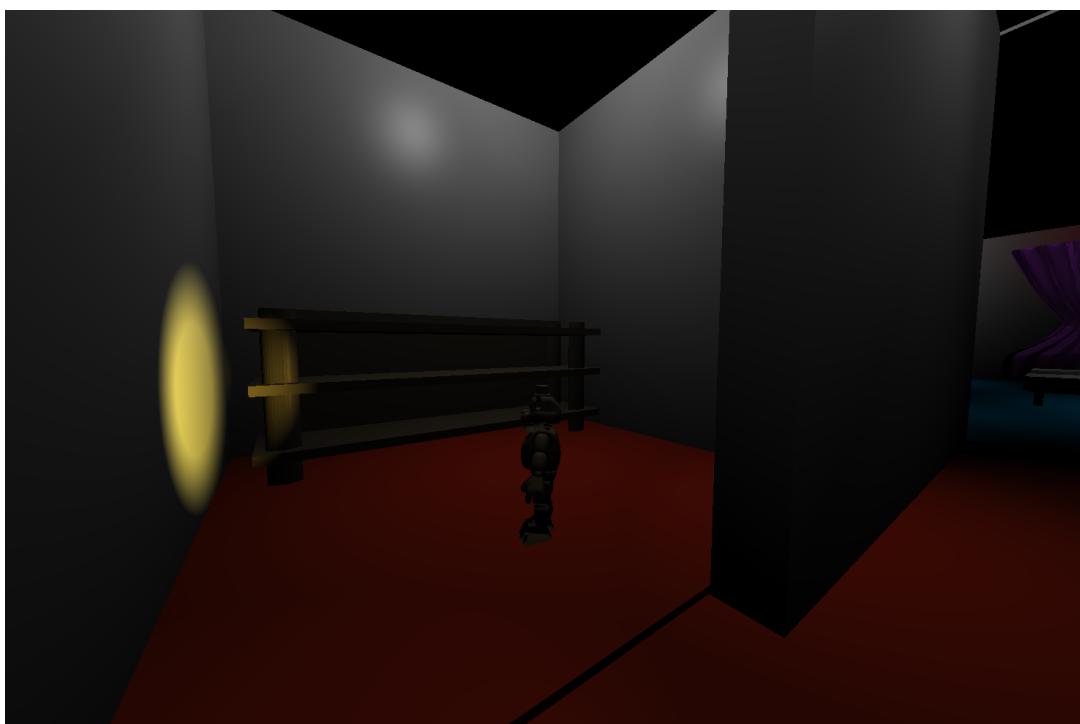
1.5 *Switch Character*

Pada implementasi ini, karakter dapat berpindah kamera dengan karakter yang berbeda. Kelompok kami menggunakan *camera1*, dan *camera2* sebagai kamera sementara sehingga apabila ingin mengganti kamera akan langsung menyimpan posisi terakhir kamera yang ditukar. Adapun inputan pada keyboard yang digunakan untuk melakukan penukaran karakter dapat diakses dengan tombol:

U → Karakter - Penjaga malam



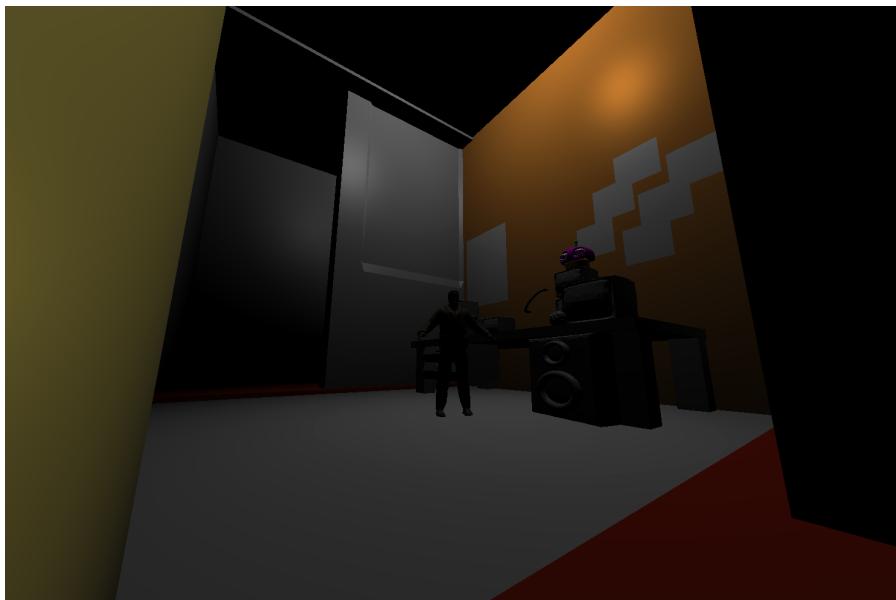
I → Karakter - *Freddy*



2. LIGHT

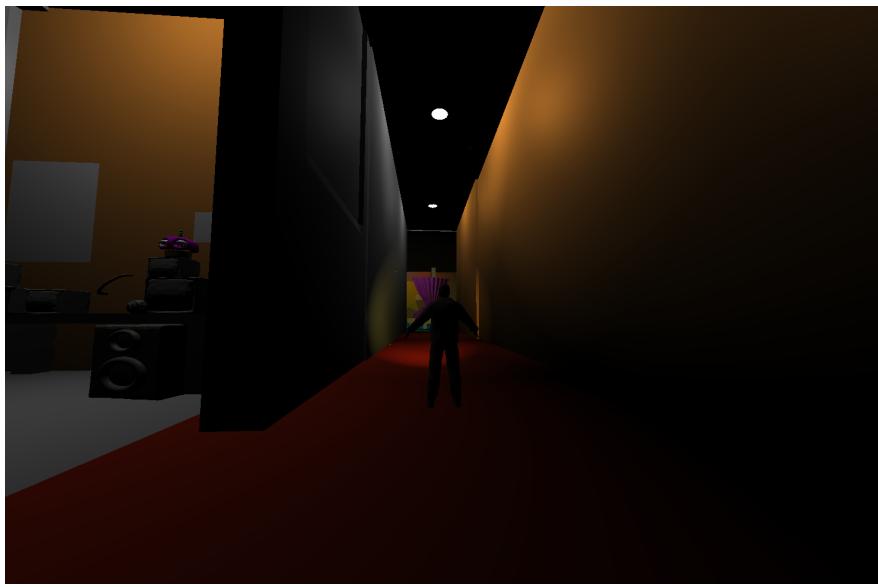
2.1 *Office*

Pada ruang *Office* terdapat 1 lampu. Lampu ini berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



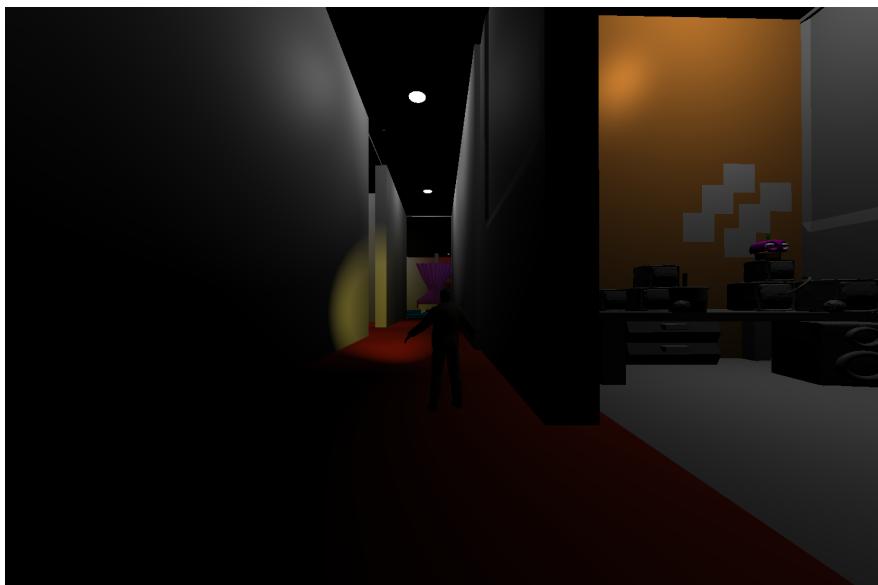
2.2 *Hallway Kanan*

Pada *Hallway Kanan* terdapat 2 lampu. Kedua lampu ini berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



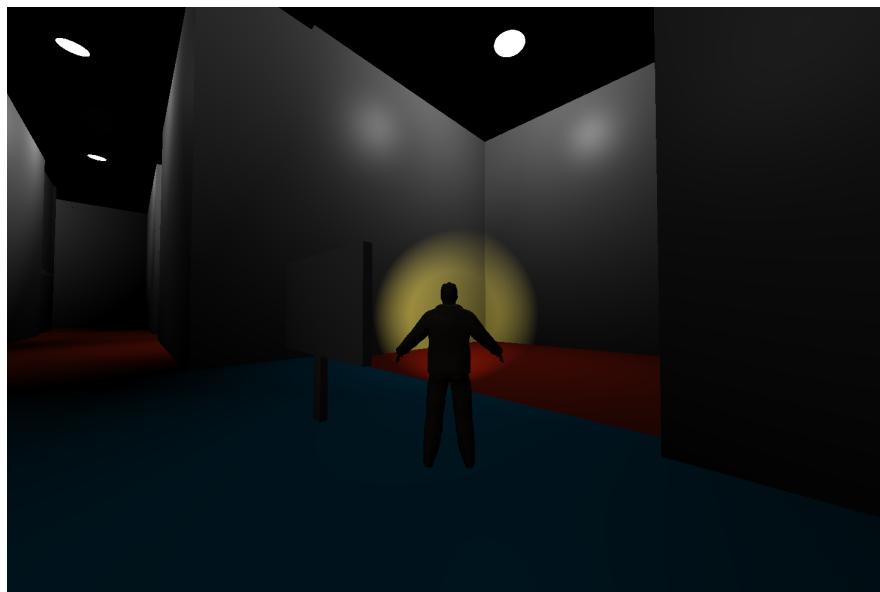
2.3 *Hallway Kiri*

Pada *Hallway Kiri* terdapat 2 lampu. Kedua lampu ini berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



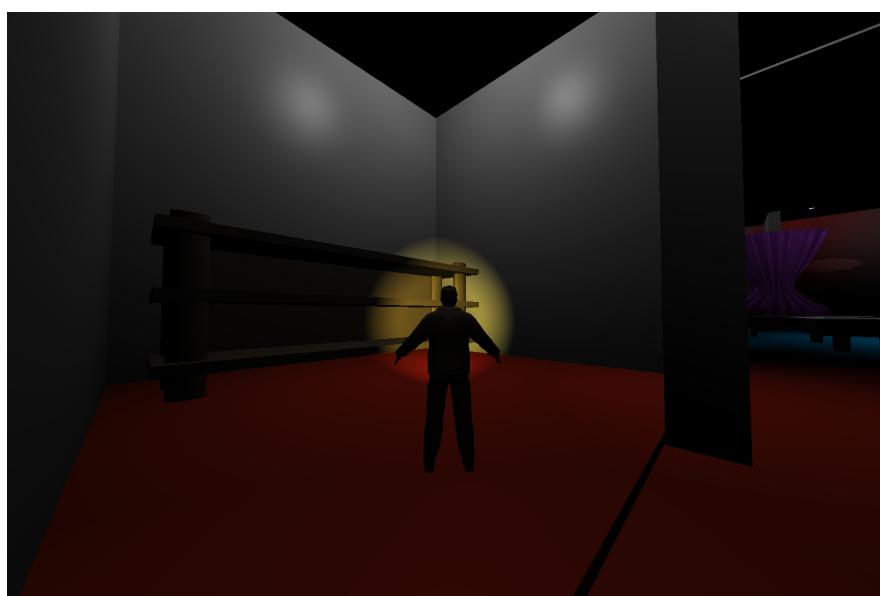
2.4 Room 1

Pada *Room 1*, terdapat 1 lampu berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



2.5 Room 2

Pada *Room 2*, terdapat 1 lampu berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



2.6 Gudang

Pada ruang gudang atau *Storage*, terdapat 1 lampu berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



2.7 Hall

Pada ruang *Hall*, terdapat 1 lampu berupa *Point Light* dengan warna putih $rgb(1,1,1)$. Lampu pada *Hall* juga dapat digerakan dengan menekan tombol pada keyboard sebagai berikut:

- Z dan X → Untuk menggerakan lampu pada sumbu X
- C dan V → Untuk menggerakan lampu pada sumbu Y
- B dan N → Untuk menggerakan lampu pada sumbu Z

Lampu ini juga bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



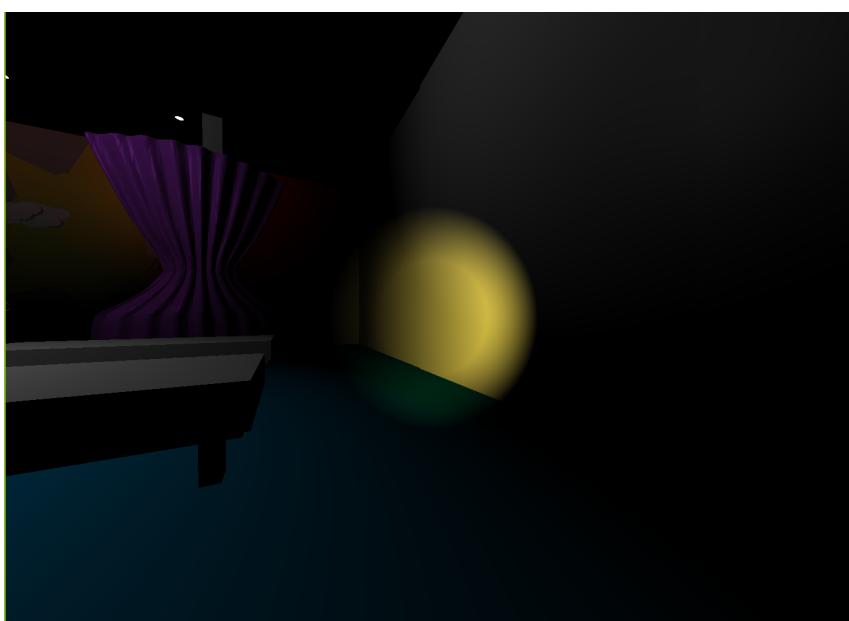
2.8 Stage

Stagelight merupakan sejumlah lampu-lampu yang menyinari panggung. Stagelight terdiri dari 3 spotlight yang mengarah ke Freddy dan 2 point light yang menyinari belakang panggung dengan warna merah. Lampu bisa dimatikan maupun dinyalakan bersama lampu-lampu lain kecuali senter dengan menekan tombol ‘O’.



2.9 Senter

Senter berupa spotlight dengan warna kuning. Posisi spotlight bergantung pada posisi karakter. Arah cahaya dari flashlight sama dengan arah karakter. Senter bisa dinyalakan maupun dimatikan dengan meng-klik tombol ‘P’.



3. COLLISION HANDLER

Konsep kode collision handler adalah membuat *boundary box* untuk setiap objek dan karakter. Kemudian, setiap kali karakter ingin bergerak, akan dicek apakah *boundary box* akan ada *overlap*, jika iya karakter tidak berjalan. Jika tidak, karakter diperbolehkan untuk berjalan.

Langkah pertama untuk *collision handler* adalah membuat *boundary box* untuk setiap objek. Hal ini bisa dicapai dengan mencari nilai maksimum dan minimum dalam sumbu X dan Z. Dari 4 nilai ini, kita bisa membuat sebuah kotak yang didefinisikan dengan 2 titik $l_1 (X_{min}, Z_{max})$ dan titik $r_1 (X_{max}, Z_{min})$.

Saat karakter ingin maju, kita cek apakah *boundary box* karakter (telah ditambah dengan translasi) dengan *boundary box* objek lain ada yang bertabrakan (*overlap*). Jika ada yang *overlap*, maka karakter tidak diperbolehkan untuk jalan. Jika tidak ada *overlap*, karakter bisa berjalan.

```
1 reference
public bool check_Boundary(Vector3 _pos, List<mesh> all_object, mesh chara)
{
    Vector2 l1 = new Vector2(chara.Xmin + _pos.X, chara.Zmax + _pos.Z);
    Vector2 r1 = new Vector2(chara.Xmax + _pos.X, chara.Zmin + _pos.Z);

    foreach (mesh obj in all_object)
    {
        Vector2 l2 = new Vector2(obj.Xmin, obj.Zmax);
        Vector2 r2 = new Vector2(obj.Xmax, obj.Zmin);
        bool overlap = check_overlap(l1, r1, l2, r2);
        if (overlap)
        {
            //Overlap -> Charcater can't move
            return false;
        }
    }
    //No Overlap -> Character can move
    return true;
}
```

```
1 reference
public bool check_overlap(Vector2 l1, Vector2 r1, Vector2 l2, Vector2 r2)
{
    if (l1.X == r1.X || l1.Y == r1.Y || r2.X == l2.X || l2.Y == r2.Y)
    {
        return false;
    }

    // If one rectangle is on left side of other
    if (l1.X > r2.X || l2.X > r1.X)
    {
        return false;
    }

    // If one rectangle is above other
    if (r1.Y > l2.Y || r2.Y > l1.Y)
    {
        return false;
    }

    return true;
}
```

4. MODEL / OBJECT

Kami membuat metode untuk membaca file .obj (*Wavefront*) yang di *export* dari *Blender*. Saat *load* file .obj menggunakan fungsi di class mesh ‘*LoadOBJFile*’, ada beberapa perubahan di kodingan dimana kita menambahkan fungsi *indices_n.add* yang bertujuan untuk menyimpan titik *indices* dari *normal vector*. Kemudian pada saat menggunakan fungsi ‘*LoadOBJFile*’, posisi serta *normal vector* dari objek yang telah di *load* akan digabungkan dengan menggunakan fungsi ‘*combine*’ menjadi 1 variabel array yang dinamakan *_realVertices*. Setelah digabungkan, maka kita dapat memanggil fungsi *Vertex Buffered Object* (VBO) dan *Vertex Array Object* (VAO) serta *setShader* nya, kemudian kita dapat memanggil fungsi ‘*Render*’ untuk menampilkan karakter maupun *environment* yang sudah di *load*.

```
2 references
public void LoadObjFile(string path)
{
    if (!File.Exists(path))
    {
        throw new FileNotFoundException("Unable to open \\" + path + "\", does not exist.");
    }

    using (StreamReader streamReader = new StreamReader(path))
    {
        while (!streamReader.EndOfStream)
        {

            List<string> words = new List<string>(streamReader.ReadLine().ToLower().Split(' '));
            words.RemoveAll(s => s == string.Empty);

            if (words.Count == 0)
            {
                continue;
            }

            string type = words[0];
            words.RemoveAt(0);
        }
    }
}
```

```

switch (type)
{
    case "v":
        vertices.Add(new Vector3(float.Parse(words[0]) / 1000000, float.Parse(words[1]) / 1000000, float.Parse(words[2]) / 1000000));
        break;

    case "vt":
        texture.Add(new Vector3(float.Parse(words[0]), float.Parse(words[1]),
                               words.Count < 3 ? 0 : float.Parse(words[2])));
        break;

    case "vn":
        normals.Add(new Vector3(float.Parse(words[0]), float.Parse(words[1]), float.Parse(words[2])));
        break;

    case "f":
        foreach (string w in words)
        {
            if (w.Length == 0)
                continue;

            string[] comps = w.Split('/');
            indeces.Add(uint.Parse(comps[0]) - 1);
            indeces_n.Add(uint.Parse(comps[2]) - 1);
        }
        break;

    default:
        break;
}

```

1 reference

```

private void combine()
{
    int n_ = 0;
    int counters = 0;
    foreach (int i in indeces)
    {

        n = (int)indeces_n[counters];

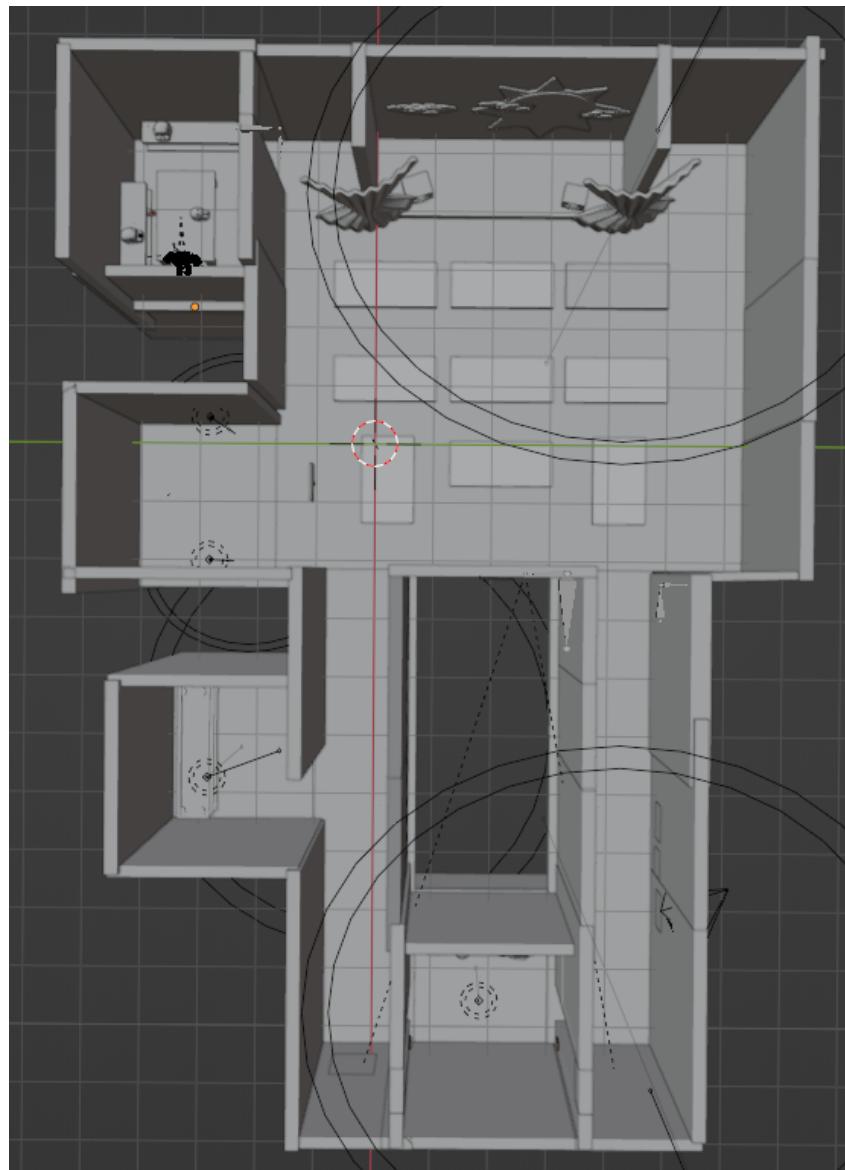
        _combine.Add(vertices[i].X);
        _combine.Add(vertices[i].Y);
        _combine.Add(vertices[i].Z);
        _combine.Add(normals[n].X);
        _combine.Add(normals[n].Y);
        _combine.Add(normals[n].Z);

        counters++;
    }
    _realVertices = _combine.ToArray();
}

```

DESAIN

Desain *environment* dibuat menggunakan *Blender* dengan memanipulasi *cube* dan *sphere* lalu di *sculpting* dan dibuat sedemikian sehingga dapat menjadi bentuk objek yang diinginkan. Untuk proses *export* kami pecah per bagian agar dapat diberi warna secara terpisah.



Untuk karakternya sendiri dibuat dengan cara yang sama, dilakukan proses pembentukan dan *sculpting* pada tiap bagian yang ada.



PENUTUP

Demikian laporan yang telah kami buat, kami berharap bahwa laporan ini dapat menjelaskan proyek kami dengan jelas. Kami mohon maaf apabila ada kesalahan pada ejaan dan penulisan kata. Atas waktunya kami ucapan terima kasih.