**RANCANG BANGUN MINIGAME SURVIVAL TWIN STICK SHOOTER BERBASIS UNITY GAME ENGINE**

**Laporan ini disusun untuk mata kuliah: Game Programming II**

**Dosen pengajar: Kurnia Jaya Eliazar, S.T., M.T.**

******

Disusun Oleh:

Yusuf Nugraha (061303006)

Febri Adenugraha (061303027)

**SK.Ketua Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT)**

**Nomor : 041/BAN-PT/AK-XIV/S1/XII/2011**

**TEKNIK INFORMATIKA – FAKULTAS TEKNIK**

**UNIVERSITAS WIDYATAMA**

**BANDUNG**

**2017**

# 

# BAB I PENDAHULUAN

## Latar Belakang Masalah

## 1.2. Tujuan

Tujuan yang ingin dicapai dari rancang bangun game berbasis Unity ini adalah sebagai berikut:

* Melatih kemampuan untuk membuat game utuh dari awal sampai dapat dimainkan.
* Menerapkan ilmu-ilmu yang telah dipelajari pada mata kuliah Game Programming II pada produk game yang seutuhnya.

## 1.3. Batasan Masalah

Dalam penyusunan laporan tugas besar ini, kami mencoba membuat batasan-batasan agar tidak terjadi penyimpangan dari tujuan yang diharapkan. Batasan-batasan tersebut adalah sebagai berikut:

* Fitur diminimalisasi hanya pada basis game, yaitu tujuan pemain, dan kemampuan game untuk mengalahkan pemain.
* Grafik pada game untuk sememntara menggunakan bentuk primitif yang disediakan oleh unity.

## 1.4. Sistematika Penulisan

*Bab I Pendahuluan*. Bagian ini berisi tentang penjelasan latar belakang masalah yang ditemukan saat kerja praktek, tujuan masalah, batasan masalah, dan sistematika penulisan laporan.

*Bab II Analisis Kebutuhan dan Perancangan Sistem*. Bagian ini berisi tentang langkah-langkah prosedural yang ditempuh dalam membangun game.

*Bab III Implementasi dan Tampilan Akhir.* Bagian ini berisi tentang manual book dan tampilan akhir dari game yang akan dibangun.

*Bab IV Penutup*. Bagian ini berisi kesimpulan akhir dari hasil tugas besar ini

# BAB II ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

## 2.1. Analisis Kebutuhan

Dalam pembuatan game ini digunakan perangkat yang dibagi menjadi dua perangkat, yaitu perangkat keras (hardware) dan perangkat lunak (software).

### 2.1.1. Kebutuhan Perangkat Keras

Perangkat keras yang digunakan oleh kami untuk merancang bangun game ini adalah

sebuah desktop PC, dengan spesifikasi sebagai berikut:

* Processor Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz (4 CPUs),~2.3GHz.
* Memory 8192MB RAM.
* VGA AMD Radeon (TM) R7 300 Series.

### 2.1.2. Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan oleh kami untuk merancang bangun game ini

adalah sebagai berikut:

* Unity-Game Engine v5.6.1f1 Personal (64bit).

## 2.2. Perancangan Sistem

Perancangan sistem pada tugas besar ini akan mencakup beberapa aspek, yaitu:

* Perancangan *game*; *scene* dan *canvas*.
* *Scripting*.
* Unduhan aspek tambahan game.

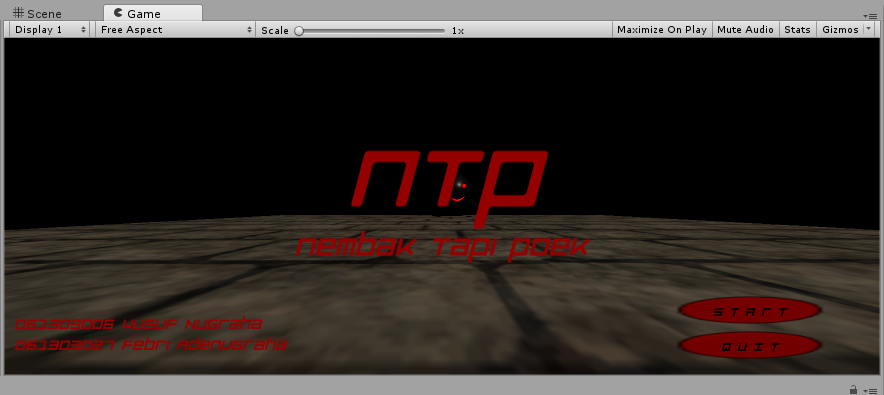
### 

### 2.2.1. Perancangan game; scene dan canvas.

Berikut adalah perincian perancangan keseluruhan game mulai dari scene menu utama dan scene permainan.

**2.2.1.1 Tampilan Menu Utama.**

Setelah game di boot-up, layar pertama yang ditampilkan adalah menu utama. menu utama terdiri dari 1 *canvas* yang berisikan judul, nama dan npm pembuat, dan 2 tombol.



Gambar 2.2.1.1.1 : Tampilan menu utama.

Menu utama memiliki tampilan yang cukup sederhana, terdiri dari judul game pada tengah layar, dan tombol pengatur permainan pada kanan bawah. Latar belakang menu utama menampilkan suasana yang akan ditampilkan pada saat permainan dimulai. Dilengkapi contoh musuh yang akan dilawan nantinya.



Gambar 2.2.1.1.2 : Tombol Pengaturan Menu Utama

Tombol pengaturan pada menu utama untuk sementara hanya terdapat dua buah, yang berfungsi untuk memulai permainan, dan keluar dari aplikasi permainan.

**2.2.1.2 Scene Survival**

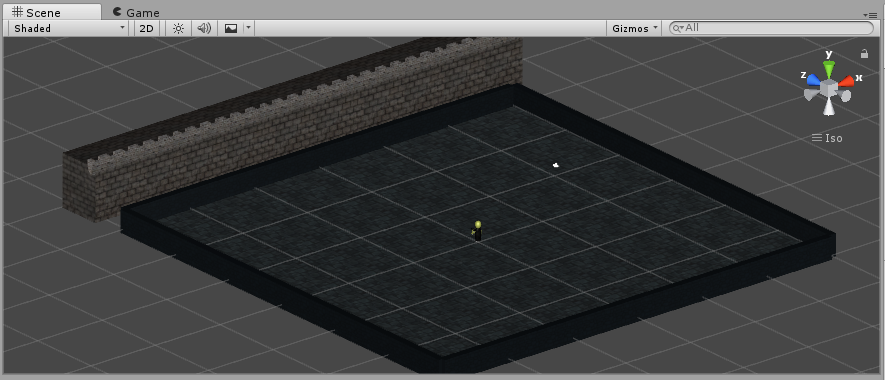
Setelah pemain mengaktifkan tombol start pada menu utama, pemain akan langsung dibawa pada layar permainan yang disebut *Scene Survival*. *Scene survival* memiliki ukuran 60 x 60 unit dibatasi oleh tembok pembatas pada keempat sisi. *Scene survival* dirancang menggunakan sebuah objek *plane* sebagai lantai dan 4 buah *cube* pada masing-masing sisi yang bersifat sebagai pembatas.

Untuk menghias dan memberikan atmosfir permainan, pada sudut utara ditambahkan dinding besar untuk membantu pemain melihat pembatas dari luar batas pencahayaan objek player.

Objek dinding luar adalah bagian dari “Castle Supply LITE” oleh “Aquarius Max”. diunduh dari Asset Store Unity.

Tekstur lantai adalah “Hand Painted Stone Texture” oleh “LowlyPoly”.

Diunduh dari Asset Store Unity.

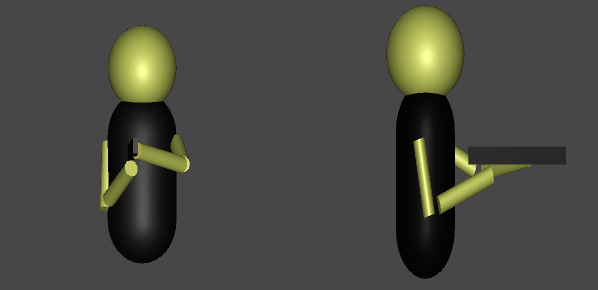


Gambar 2.2.1.2.2 : Scene Survival

Pemain akan menggerakkan sebuah objek karakter yang diberi nama *Player*. *Player* dapat bergerak bebas kearah depan, belakang, kiri dan kanan dengan menggunakan tombol A,S,D,W pada keyboard. Selain itu player dapat menembak menggunakan klik kiri mouse, dan mengisi peluru pada senjatanya menggunakan tombol R.

Pada saat menembak dan mengisi peluru, objek *player* akan mengeluarkan suara sesuai aktivitasnya. Suara tembakan dan pengisian peluru adalah bagian dari “Weapon Soldier Sounds Pack” oleh “MentalDreams”. Diunduh dari Asset Store Unity.

Setiap peluru yang ditembakkan dan mengenai objek musuh, pemain akan memberikan 1 unit tingkat kerusakan pada hitpoint musuh. Player sendiri memiliki 10 unit hitpoint yang akan berkurang apabila terkena kontak oleh musuh, dan permainan akan berakhir jika hitpoint player adalah 0.



Gambar 2.2.1.2.3 : Player Object

Pada saat permainan dimulai, permainan ditampilkan dari sudut *bird-eye view*. Dengan penerangan yang sangat minim untuk menambah kesan menyeramkan. Sudut pandang ini didesain dengan cara menghilangkan *point light* utama, dan memberikan *lighting* tipe *spotlight* yang dipasangkan pada objek senjata milik karakter pemain. Untuk memperjelas posisi dan pergerakan karakter pemain, sebuah *point light* dengan tingkat terang sangat rendah dipasangkan pada tubuh karakter pemain.

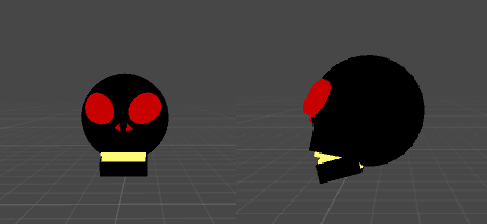


Gambar 2.2.1.2.4 : Sudut Pandang Pemain.

Dalam permainan, pemain dapat menembak musuh-musuh yang akan bermunculan secara acak dalam *scene survival*. Musuh-musuh didesain dengan memiliki kelebihan dan kekurangan masing-masing. Baik dalam ukuran, jumlah *hitpoint*, dan kecepatan bergerak.

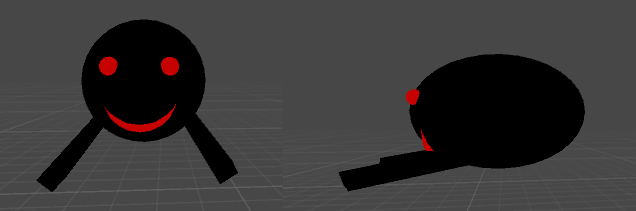
Saat musuh kehabisan *hitpoint*, sebuah file suara akan dibunyikan, suara yang dibunyikan adalah bagian dari “Lil' Critter SFX” oleh “JP Neufeld”. Diunduh dari Asset Store Unity.

Berikut adalah jenis-jenis musuh yang ditampilan dalam game:



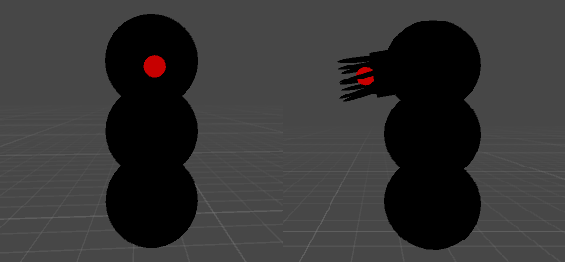
Gambar 2.2.1.2.5 : Jenis Musuh jenis “Skull”.

*Skull* didesain sebagai musuh paling mudah, dengan 3 unit *hitpoint*, namun *Skull* memiliki kelebihan dalam kecepatan, dan ukuran yang kecil sehingga sulit terkena tembakan. Bila terkena objek *Player*, pemain akan kehilangan 3 unit *hitpoint*.



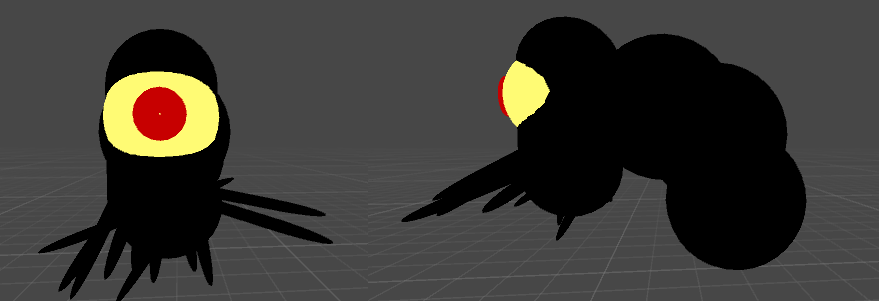
Gambar 2.2.1.2.6 : Jenis Musuh jenis “Monstar”.

*Monstar* didesain sebagai musuh yang dapat menerima lebih banyak peluru daripada skull, dengan 10 unit *hitpoint*. Walaupun lambat dan besar, *Monstar* akan memberikan tingkat kerusakan 5 unit *hitpoint* pada saat kontak dengan objek *Player.*



Gambar 2.2.1.2.6 : Jenis Musuh jenis “Hileud”.

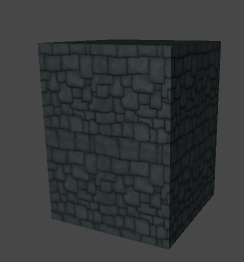
*Hileud* tidak dapat bergerak dan mengejar pemain. Setelah muncul dalam permainan *Hileud* akan berdiri diam dan hanya menghadap player. Hileud memiliki 10 unit *hitpoint*, dan akan memberikan tingkat kerusakan 5 unit *hitpoint* pada saat kontak dengan objek *Player*.



Gambar 2.2.1.2.7 : Jenis Musuh jenis “Eyestar”.

*Eyestar* akan muncul dalam permainan jika pemain berhasil bertahan selama satu menit. Dan akan muncul dengan jumlah kelipatan tiap menit berikutnya. *Eyestar* menembakkan banyak proyektil yang akan memberikan tingkat kerusakan 10 unit *hitpoint*. Dan *Eyestar* memiliki 50 unit *hitpoint*.

Selain musuh, objek *cube* akan dimunculkan secara acak pada scene, *cube* bersifat sebagai penghalang pergerakan objek *Player*, dan ditampilan sebagai tembok.



Gambar 2.2.1.2.7 : “Wall Cube”

### 2.2.2. Scripting.

Untuk sementara, hanya terdapat 14 script aktif dalam game. Berikut adalah penjelasan detail beberapa script utama.

2.2.2.1 Camera Control

void Start(){  
         myTransform = transform;  
     }

public void SetTarget (Transform aTransform){  
         followTarget = aTransform;  
     }

void Update (){  
        if(followTarget != null)  
            myTransform.position = Vector3.Lerp (myTransform.position, followTarget.position + targetOffset, camSpeed \* Time.deltaTime);  
    }

Camera Control berfungsi untuk mengatur kamera agar selalu mengikuti posisi objek *Player*.

2.2.2.2 PlayerControl

protected void Awake()  
     {  
         Animation = GetComponentInChildren <Animation>();  
  
     }  
  
     void Start () {  
         myRigidbody = GetComponent<Rigidbody>();  
         mainCamera = FindObjectOfType<Camera>();  
     }  
      
  
     void Update ()   
     {  
          moveInput = new Vector3(Input.GetAxisRaw("Horizontal"), 0f, Input.GetAxisRaw ("Vertical"));  
        moveVelocity = moveInput \* moveSpeed;  
  
        Ray cameraRay = mainCamera.ScreenPointToRay(Input.mousePosition);  
        Plane groundPlane = new Plane(Vector3.up, Vector3.zero);  
        float rayLength;  
  
  
        if(groundPlane.Raycast(cameraRay, out rayLength))  
        {  
            Vector3 pointToLook = cameraRay.GetPoint(rayLength);  
  
            transform.LookAt(new Vector3(pointToLook.x, transform.position.y, pointToLook.z));  
        }  
        if (Input.GetMouseButtonDown (0))  
            pestol.setNembak (true);  
          
  
        if (Input.GetMouseButtonUp (0))  
            pestol.setNembak (false);  
          
    }  
  
    void FixedUpdate (){  
        myRigidbody.velocity = moveVelocity;  
    }  
}

PlayerControl berfungsi untuk pergerakan dan aksi objek *Player*.

2.2.2.3 Tembak

void Update ()   
     {  
         if (nembak)  
         {  
            shotcounter -= Time.deltaTime;  
            if(shotcounter <= 0 && tempPelor >= 0 && !reload)  
            {  
                float vol = Random.Range (volLowRange, volHiRange);  
                source.PlayOneShot(shootsound,vol);  
                shotcounter = rateoffire;  
                pelors pelorbaru = Instantiate(pelor, ujungBedil.position, ujungBedil.rotation)as pelors;  
                pelorbaru.pelorspeed = pelorsspeed;  
                tempPelor --;  
                ammocount.text = tempPelor.ToString();  
            }  
            if (tempPelor <=0 && !reload)  
            {  
                reload = true;  
                ammocount.text = "RELOADING";  
                source.PlayOneShot(reloadsound,1.0f);  
                StartCoroutine (ReloadWeapon(2));  
                tempPelor = maxPelor;  
            }  
        }  
        else  
        {  
            shotcounter = 0;  
            if(Input.GetKeyDown(KeyCode.R))  
            {  
                reload = true;  
                ammocount.text = "RELOADING";  
                source.PlayOneShot(reloadsound,1.0f);  
                StartCoroutine (ReloadWeapon(2));  
                tempPelor = maxPelor;  
            }  
        }  
    }  
    public IEnumerator ReloadWeapon(float timer)   
    {  
        yield return new WaitForSeconds (timer);  
        reload = false;  
        tempPelor = maxPelor;  
        ammocount.text = tempPelor.ToString();  
    }  
  
    *//gettter & setter*  
  
    public void setNembak(bool nembak){  
        this.nembak = nembak;  
    }  
  
    public bool getNembak(){  
        return nembak;  
    }

Tembak berfungsi sebagai pengatur aksi tembakan *Player*, dan pengendalian kemunculan peluru dan juga pengisian kembali peluru.

2.2.2.4 pelors

void Update ()   
     {  
         transform.Translate(Vector3.forward \* pelorspeed \* Time.deltaTime);  
  
         hapuspelor -= Time.deltaTime;  
         if(hapuspelor <= 0 )  
         {  
             Destroy(gameObject);  
         }  
     }  
     void OnCollisionEnter(Collision other)  
     {  
         if (other.gameObject.tag == "enemy")  
         {  
             float vol = Random.Range (volLowRange, volHiRange);  
             source.PlayOneShot(dmgsound, vol);  
             other.gameObject.GetComponent<EnemyHealth>().dmgingenemy(dmgdone);  
             Destroy(gameObject, 1);  
         }  
     }

Pelors mengelola peluru yang ditembakkan dari objek *Player*, mengontrol tingkat kerusakan, kecepatan, dan menghapus objek peluru setelah objek memberikan kontak pada objek lainnya.

2.2.2.5 EnemyHealth

void Start ()   
     {  
          maxhealth = health;  
     }  
     void Update ()   
     {  
         if(maxhealth <= 0)  
         {  
             Instantiate(blood, transform.position, transform.rotation);  
             Destroy(gameObject);  
         }  
     }  
     public void dmgingenemy(int dmg)  
     {  
         maxhealth -= dmg;  
     }

}

EnemyHealth mengelola tingkat kerusakan peluru pada musuh musuh yang terkena kontak peluru. Selain itu, script ini menampilkan partikel yang berupa darah apabila unit hitpoint musuh bernilai 0 atau kurang, dan menghapus objek musuh tersebut.

2.2.2.6 Timer

void Start ()   
      {  
          starttime = Time.time;  
       }  
     void Update ()   
      {  
          if (finish)  
              return;  
                 float t = Time.time - starttime;  
           string minutes = ((int) t / 60).ToString();  
          string seconds = (t % 60).ToString("f2");  
  timertext.text = minutes + ":" + seconds;  
     }  
     public void ended()  
     {  
         finish = true;  
         timertext.color = Color.green;  
     }

Timer mengelola berapa lama waktu permainan, dan menghentikan waktu permainan apabila unit *hitpoint* pemain bernilai 0 atau kurang.

2.2.2.7 Spawner

void Start ()   
     {  
         StartCoroutine(spawning());  
         InvokeRepeating("BossSpawn",60.0f, 60.0f);  
         bossSpawnTimes = 0;  
     }  
     void Update ()   
     {  
         spawnWait = Random.Range (spawnLeast, spawnMost);  
          spawnedenemies = GameObject.FindGameObjectsWithTag("enemy");  
     }  
  
     IEnumerator spawning()  
     {  
         yield return new WaitForSeconds(startWait);  
 while (!stop)  
         {  
             randEnemy = Random.Range (0,7);  
 Vector3 spawnPos = new Vector3

(Random.Range (spawnValues.x, spawnValues.x),

10, Random.Range (-spawnValues.z, spawnValues.z));  
Instantiate (enemies[randEnemy], spawnPos + transform.TransformPoint (0,0,0),

gameObject.transform.rotation);  
            yield return new WaitForSeconds (spawnWait);  
        }  
    }  
    public void Stop()  
    {  
        stop = true;  
        foreach(GameObject se in spawnedenemies)  
        {  
            se.SetActive(false);  
        }  
  
    }  
    void BossSpawn()  
    {  
        bossSpawnTimes = bossSpawnTimes + 1;  
        source.PlayOneShot(bosssound,2.0f);  
        Vector3 spawnPos = new Vector3 (Random.Range (- spawnValues.x, spawnValues.x), 4.5f, Random.Range (-spawnValues.z, spawnValues.z));  
        Instantiate (boss, spawnPos + transform.TransformPoint (0,0,0), gameObject.transform.rotation);  
    }

Spawner mengelola kemunculan musuh yang diacak jenis dan lokasinya.

**2.2.3 Unduhan aspek tambahan game.**

Untuk memberikan berbagai efek tambahan pada game seperti suara, dan tekstur pada berbagai objek, tim penulis mengunduh beberapa aset dari berbagai sumber.

Aset-aset tersebut adalah sebagai berikut:

Musik Latar:

Nervous by Kevin MacLeod (incompetech.com)

Licensed under Creative Commons: By Attribution 3.0

License : <http://creativecommons.org/licenses/by/3.0/>

Efek Suara Saat pemain kehabisan hitpoint:

17 Evil laugh.wav by The Baron

<https://freesound.org/people/The%20Baron/sounds/98382/>

Efek suara saat musuh kehabisan hitpoint:

Lil' Critter SFX by JP Neufeld

dari Unity Asset Store

[https://www.assetstore.unity3d.com/en/#!/content/5016](https://www.assetstore.unity3d.com/en/%23!/content/5016)

Efek suara tembakan dan pengisian ulang peluru:

Weapon Soldier SOunds Pack by MentalDreams

dari Unity Asset Store

[https://www.assetstore.unity3d.com/en/#!/content/29662](https://www.assetstore.unity3d.com/en/%23!/content/29662)

Font:

RazerBlackwidow-Regular.ttf by Joao Mambelli

<http://www.dafont.com/razer-blackwidow.font>

Tekstur lantai scene:

Hand Painted Stone Texture by LowlyPoly

dari Unity Asset Store

[https://www.assetstore.unity3d.com/en/#!/content/73949](https://www.assetstore.unity3d.com/en/%23!/content/73949)

Objek Dinding luar:

Castle Supply LITE by Aquarius Max

dari Unity Asset Store

<https://www.assetstore.unity3d.com/en/#!/content/23699>

# BAB III IMPLEMENTASI DAN TAMPILAN AKHIR

## 3.1. Implementasi

Tujuan permainan adalah bertahan selama mungkin dari serangan berbagai musuh, pemain dapat mengulang permainan dan berusaha memecahkan rekor waktu sebelumnya.

Kontrol karakter pemain terdiri dari tombol A,S,D,W, yang berguna sebagai penggerak posisi karakter, tombol click kiri mouse untuk menembakkan senjata, dan tombol R untuk pengisian kembali peluru.

## 3.2. Tampilan Akhir

# BAB IV PENUTUP

## 4.1. Kesimpulan

Setelah kami menyelesaikan tugas besar ini, kami dapat menyimpulkan beberapa point, yaitu:

* Unity-Game Engine adalah sebuah development kit yang sangat powerful dalam membangun sebuah game.
* Metode pembelajaran dengan cara mempraktekan apa yang kita pelajari ke dalam sebuah project sangatlah efektif. Kami menjadi mengerti bagaimana mengimplementasikan teori yang sudah kami pelajari ke dalam sebuah project.
* Tingginya tingkat kesulitan membangun game dari awal, dan tugas ini dapat menjadi pembelajaran agar tidak menganggap sepele industri video game.