BAB III
PERANCANGAN DAN IMPLEMENTASI

3.1 Gambaran Umum Aplikasi

Pada bab 3 ini akan membahas tentang perancangan aplikasi, struktur navigasi, pembuatan aplikasi, uji coba, dan implementasi. Tujuan dari pembuatan aplikasi ini adalah untuk mengenalkan masyarakat terkait adanya kampanye "Stop Illegal Fishing!" dan adanya langkah tegas Kementerian Kelautan dan Perikanan Indonesia terhadap aktivitas tersebut. Aplikasi ini berbentuk *games* menembak kapal, yang mana nantinya dijelaskan bahwa kapal – kapal yang ditembak tersebut merupakan kapal yang melanggar peraturan yang berlaku dengan dasar pasal 69 Undang – undang No. 45/2009 tentang perikanan(UU Nomor 45, 2009).

Pembuatan aplikasi ini menggunakan *Unity* dengan bahasa pemrogramannya yaitu C#. Pada aplikasi ini terdapat tampilan *Splash Screen*, menu utama, menu Play, menu Info, menu Exit. Pada menu Play berisi *games* menembak kapal yang mana apabila scorenya mencapai angka 100 maka permainan itu akan selesai dan pindah ke menu Info secara otomatis. Pada menu Info berisi informasi animasi cerita *games* dan informasi mengenai kampanye "Stop Illegal Fishing!" dan di akhir animasi akan kembali ke menu Utama. Menu Exit untuk keluar dari aplikasi.

3.2. Spesifikasi *Hardware* dan *Software* Minimum

Pada pembuatan aplikasi ini, penulis menggunakan spesifikasi, sebagai berikut:

Hardware:

1. Processor Intel® Core™ I5-6600.
2. RAM 8 GB.
3. *Hard disk* dengan ruang bebas minimal 100 MB untuk pembuatan aplikasi.

Software:

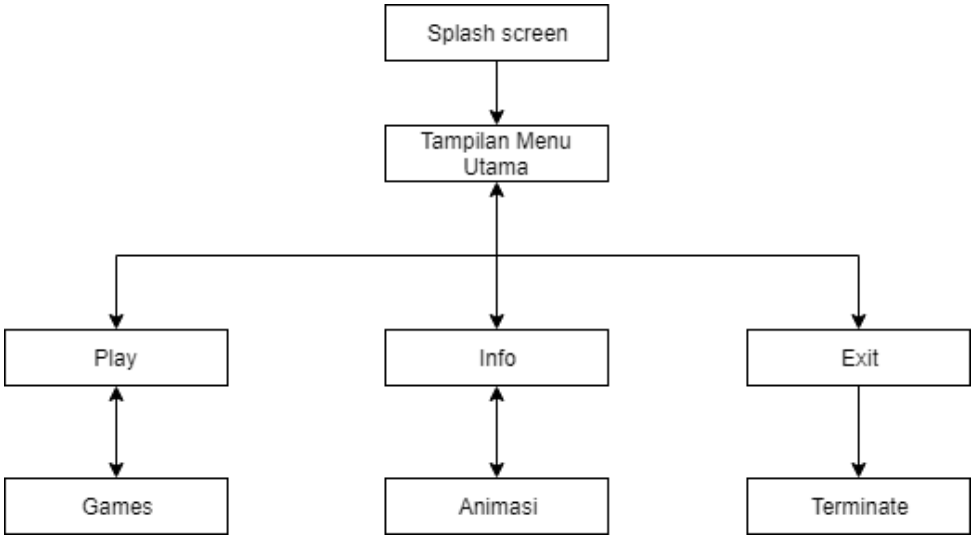
1. Unity 2018.2.21f1
2. Adobe Premier Pro

3.3 Rancangan Aplikasi

Berikut adalah tahapan perancangan dari aplikasi yang akan dibuat.

3.3.1 Struktur Navigasi

Langkah awal dalam perancangan aplikasi yang dibuat adalah membuat struktur navigasi. Langkah ini merupakan langkah yang penting untuk dilakukan. Struktur navigasi yang digunakan adalah navigasi campuran atau *composite*. Struktur navigasi campuran pada aplikasi ini merupakan gabungan dari struktur navigasi linear, non-linear, dan hirarki.



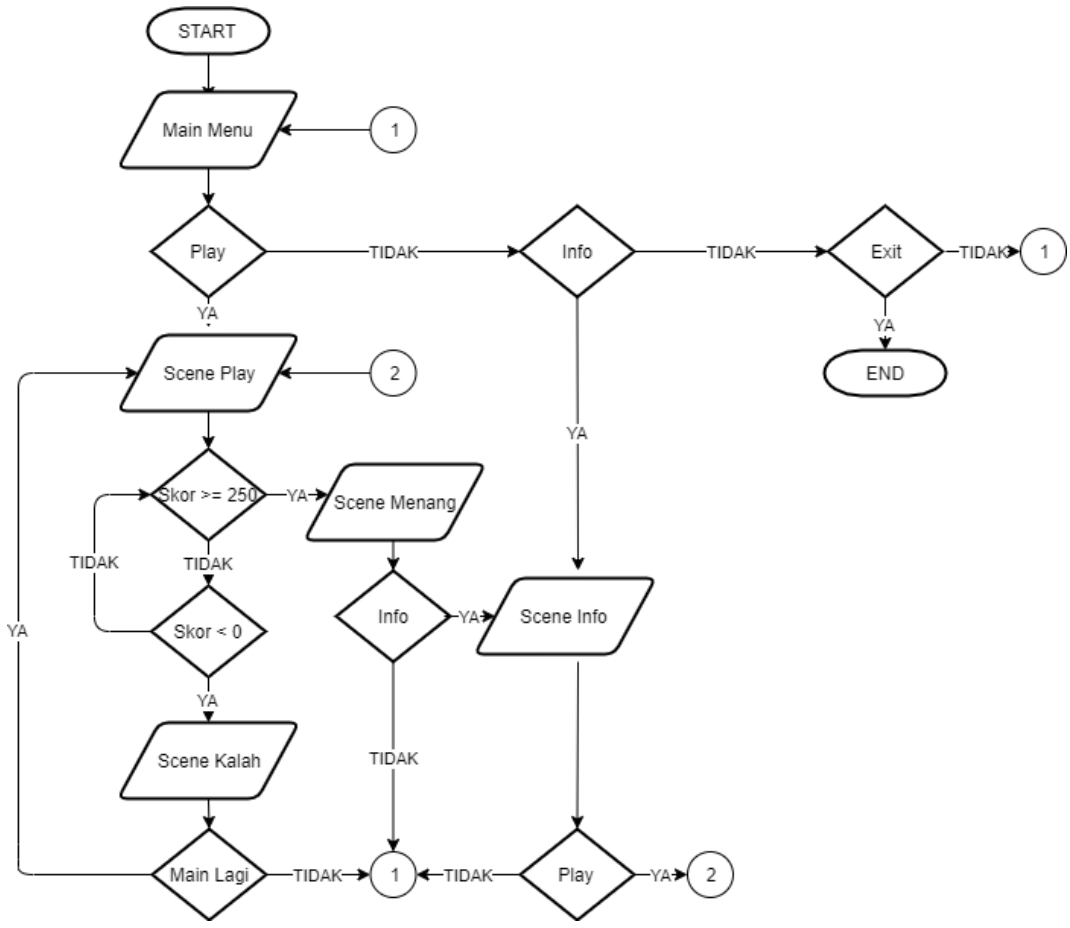
Gambar 3.1 Struktur Navigasi Program

Pada gambar 3.1 menjelaskan pada awal aplikasi dijalankan maka akan muncul splash screen judul *game* yaitu “Tenggelamkan”, kemudian akan berlanjut ke tampilan Menu Utama, yang mana terdapat 3 pilihan Menu yaitu Menu Play, Menu Info dan Menu Exit. Pada Menu Play, akan berisikan *games* menembak kapal yang apabila sudah berhasil mengumpulkan score 100 maka akan berpindah ke Animasi yang berada pada Menu Info. Pada Menu Info, berisikan animasi cerita pada *games* dan informasi mengenai

"*Stop Illegal Fishing!*", setelah *video* selesai, maka akan berpindah ke halaman Menu Tampilan Utama. Pada Menu Exit, berisikan terminate aplikasi yang akan menyebabkan keluar dari aplikasi.

3.4 Flowchart

Dalam pembuatan dan pengembangan aplikasi tersebut, penulis membuat bagan alur atau flowchart untuk menjelaskan langkah-langkah dalam menggunakan aplikasi pembelajaran ini bagi pengguna agar dapat mengerti kerja sistem dari aplikasi tersebut. Pada Gambar 3.2 merupakan gambar bagan alir program yang diciptakan oleh penulis untuk aplikasi ini.



Gambar 3.2 Bagan Alir Program

3.5 Pseudocode

Berikut adalah *pseudocode* yang digunakan.

```
01| Start
02| If pointer mengarah ke box play then pindah ke scene Play
03|     Show score = 0
04|     If pointer mengarah ke object kapal illegal then score tambah 50
05|     Else If pointer mengarah ke object kapal legal score kurangi 50
06|     Else If pointer mengarah ke object nelayan then score kurangi 500
07|     End If
08|     Update score
09|     If score greater than or equal to 250 then pindah ke scene Win
10|         Show Text “You’re Win!”
11|         If pointer mengarah ke box info then pindah ke scene Info
12|         Else If pointer ke box exit then pindah ke scene Menu
13|         End If
14|     Else If score less than 0 then pindah ke scene Lose
15|         Show Text “You’re Lose!”
16|         If pointer mengarah ke box play then pindah ke scene Play
17|         Else If pointer ke box exit then pindah ke scene Menu
18|         End If
19|     End If
20| Else If pointer mengarah ke box info then pindah scene Info
21|     Autoplay Video Animasi within loop condition
22|     If pointer mengarah ke box Exit then pindah scene Menu
23|     End If
24| Else If pointer mengarah ke box exit then Terminate
25| End If
```

Dari *pseudocode* diatas, maka saat *game* dijalankan *player* berada pada *scene* Menu (baris 1-25), pada *scene* Menu terdapat 3 kondisi yang disajikan melalui visualisasi *box*. Tiap *box* akan memindahkan *player* menuju *scene* sesuai dengan label tersebut, seperti *box* Play akan memindahkan

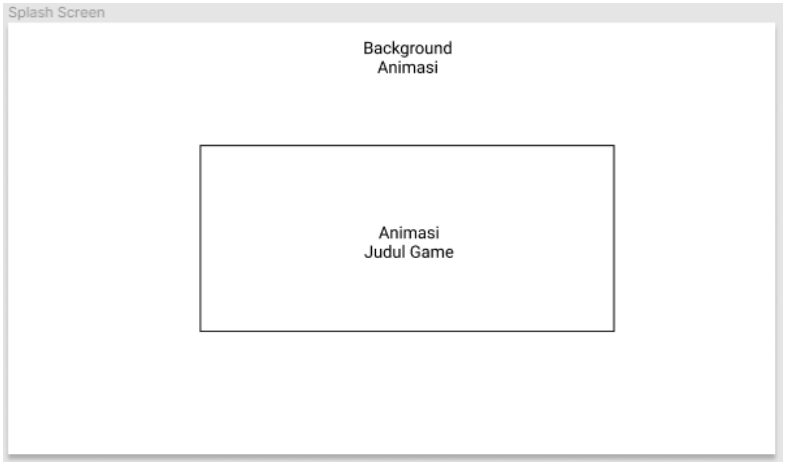
player ke *scene* Play (baris 2-19), *box* Info akan memindahkan *player* ke *scene* Info (baris 20-23), dan *box* Exit akan keluar dari *game*.

3.6 Rancangan Tampilan

Pada tahap ini menjelaskan tentang rancangan tampilan dari aplikasi yang dibuat yang tediri dari tampilan *splash screen*, tampilan menu utama, tampilan play, tampilan info, tampilan win/lose.

3.5.1 Rancangan Tampilan Splash Screen

Rancangan dari tampilan awal aplikasi yang dibuat adalah *Splash Screen*, yaitu tampilan awal sebelum masuk ke tampilan menu utama.

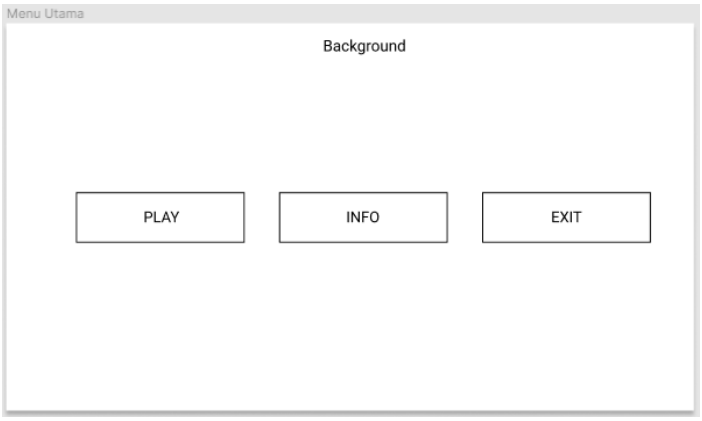


Gambar 3.3 Rancangan Tampilan Splash Screen

Pada saat aplikasi dijalankan maka tampilan pertama yang keluar adalah *Splash Screen* seperti pada gambar 3.3, tampilan ini memuat judul dari aplikasi yang dibuat yaitu "*Tenggelamkan*". Tampilan ini hanya akan muncul dalam beberapa saat saja, kemudian tampilan yang akan muncul adalah tampilan menu utama.

3.6.2 Rancangan Tampilan Menu Utama

Tampilan menu utama muncul setelah tampilan *Splash Screen* selesai, pada tampilan ini pengguna aplikasi diminta untuk menentukan pilihan seperti pada gambar 3.4.



Gambar 3.4 Rancangan Tampilan Menu

Pada tampilan ini pengguna aplikasi dihadapkan dengan 3 pilihan yang berupa object box dengan nama yang berbeda yaitu Play, Info dan Exit. Pengguna aplikasi hanya perlu mengarahkan pointer pada salah satu box, kemudian box itu akan membuka scene masing-masing sesuai dengan namanya, box Play untuk Tampilan Play, box Info untuk Tampilan Info sedangkan untuk box Exit maka keluar dari aplikasi.

3.6.3 Rancangan Tampilan Play

Tampilan play muncul apabila pengguna aplikasi memilih untuk mengarahkan pointer ke box Play pada tampilan menu.

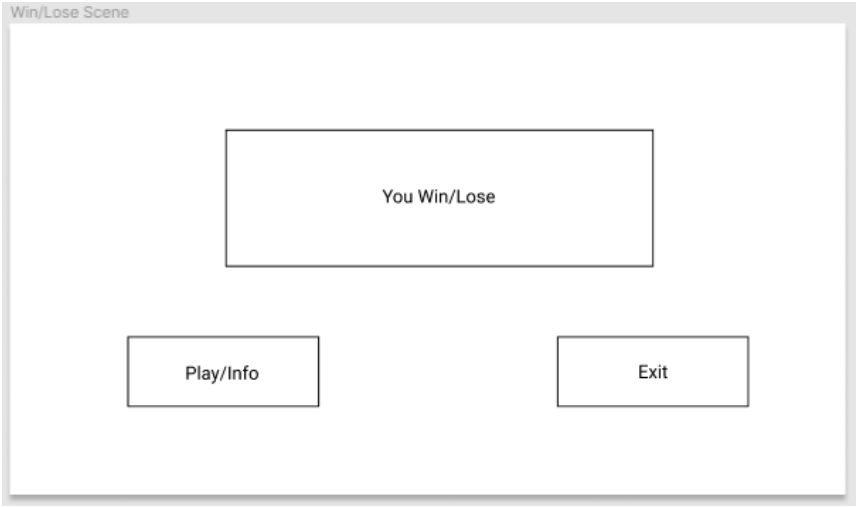
Gambar 3.5 Rancangan Tampilan Play

Pada tampilan ini pengguna aplikasi harus menembak object yang berbentuk segitiga pada Gambar 3.5, setelah mengarahkan pointer kepada object tersebut maka object tersebut akan hilang, dan terjadi perubahan score.

Pada tampilan ini tidak terdapat box untuk pindah scene, untuk pindah scene dari tampilan ini diperlukan kondisi yang harus dipenuhi pada perubahan score, yaitu apabila terkumpul 250 score maka akan pindah scene ke Win, sedangkan apabila kurang dari 0 maka pindah scene ke Lose.

3.6.4 Rancangan Tampilan Win/Lose

Tampilan win/lose muncul apabila salah satu kondisi win/lose pada tampilan play terpenuhi. Pada dasarnya tampilan ini sama, hanya saja pada aplikasi yang dibuat akan menjadi 2 scene yang berbeda.

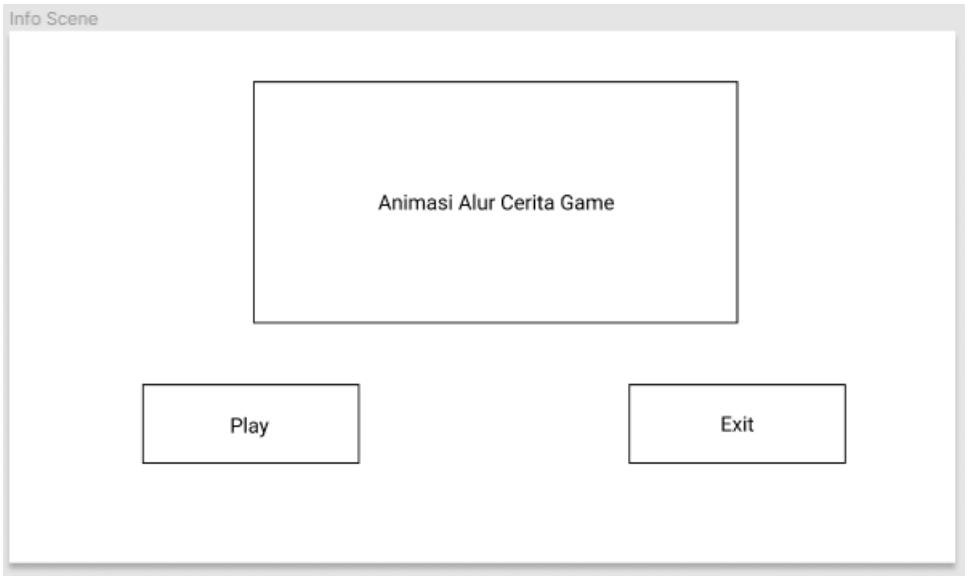


Gambar 3.6 Rancangan Tampilan Win/Lose

Seperti pada Gambar 3.6, tampilan menjadi 2 scene yang berbeda, yang mana apabila win maka text yang digunakan adalah “You Win” dan box yang muncul adalah box Play dan Exit. Jika, kondisi lose yang terpenuhi maka text yang digunakan adalah “You Lose” dan box yang muncul adalah box Info dan Exit. Box Play akan mengarah ulang ke Tampilan Play, sedangkan box Info akan mengarahkan ke Tampilan Info, dan box Exit akan mengarahkan ke tampilan Menu.

3.6.5 Rancangan Tampilan Info

Tampilan ini akan muncul apabila pada Tampilan Menu memilih box Info atau pada Tampilan Win memilih box Info.



Gambar 3.7 Rancangan Tampilan Info

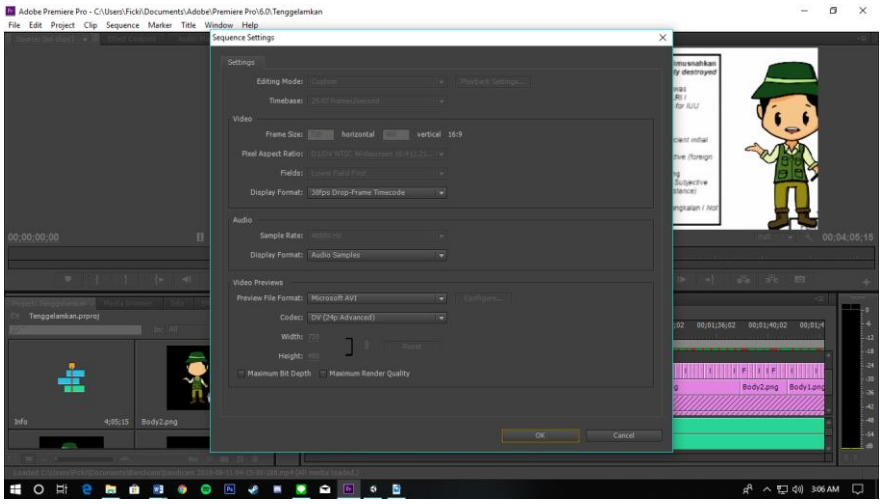
Seperti Gambar 3.7, tampilan pengguna aplikasi akan diputarkan *video* animasi tentang alur cerita *game* dan *video* ini akan bersifat looping, sehingga apabila pengguna aplikasi sudah mengerti dari penjelasan *video* maka bisa memilih box Play untuk ke Tampilan Play atau box Exit untuk kembali ke Tampilan Menu.

3.7 Pembuatan Aplikasi

Pada sub bab ini akan membahas cara pembuatan aplikasi *game* "Tenggelamkan" dan animasi *video* pada *game* tersebut.

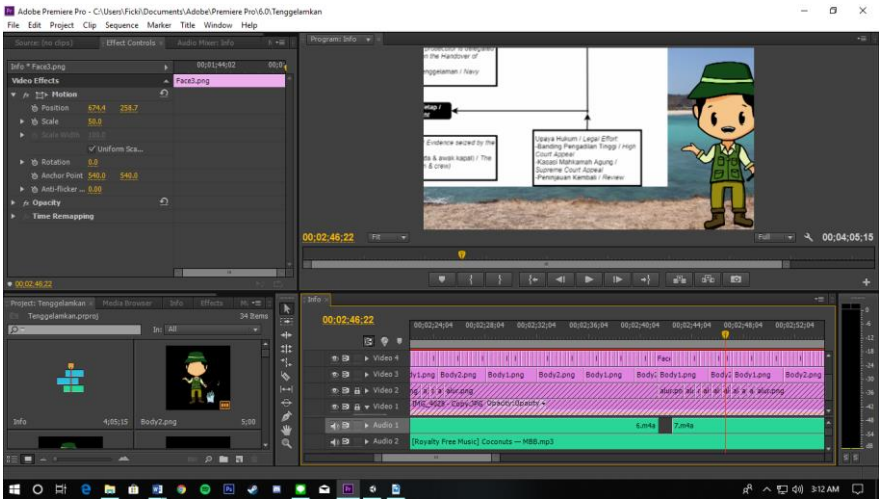
3.7.1 Pembuatan Animasi

Pada tahap pembuatan animasi, aplikasi yang digunakan adalah Adobe Premiere Pro CS6. Animasi ini berisi tentang alur proses penenggelaman kapal sesuai prosedur yang sudah ditetapkan, dan cuplikan tata cara permainan *game* pada aplikasi ini. Untuk pembuatan animasi tentang alur proses, penulis menggunakan beberapa aplikasi pendukung lainnya seperti *voice recorder* dan *corel draw*, yang mana masing-masing aplikasi ini digunakan untuk merekam suara untuk penjelasan dan membuat gambar karakter animasi 2D.



Gambar 3.8 Sequence Settings Adobe Premiere Pro

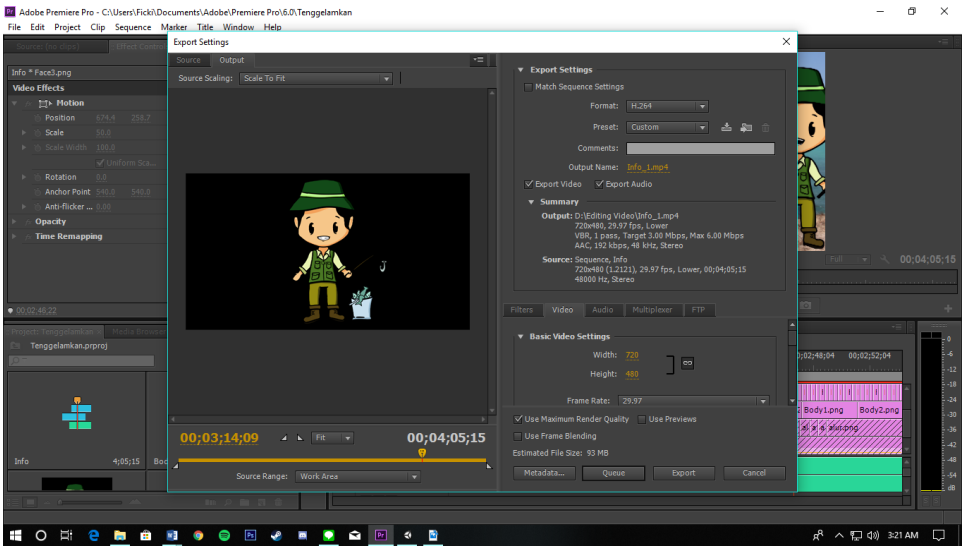
Gambar 3.8 adalah tampilan jendela *Sequence Settings* yang mana, pada saat pambuatan animasi setelah memilih *New Project*, akan diminta untuk menentukan *Frame Size* dan *Timebase*. Dalam pembuatan animasi ini, penulis menggunakan *Frame Size* dengan lebar 720 dan tinggi 480, serta *Timebase* 29.97 *frame/second.*



Gambar 3.9 Layer Animasi

Tampilan pada gambar 3.9 adalah layer (*tab* kanan bawah) yang digunakan penulis pada pembuatan animasi. Pada animasi ini, penulis menggunakan 4 *layer video* dan 2 *layer audio*. Penulis menggunakan *layer audio* 1 sebagai *layer* untuk penjelasan alur, sedangkan *layer audio 2* untuk *backsound*. Untuk *layer video*, Penulis menggunakan *layer video* 1 untuk

background, *layer video* 2 untuk gambar skema alur proses penenggelaman, *layer video* 3 untuk gambar badan karakter, dan *layer video* 4 untuk mimic wajah karakter.

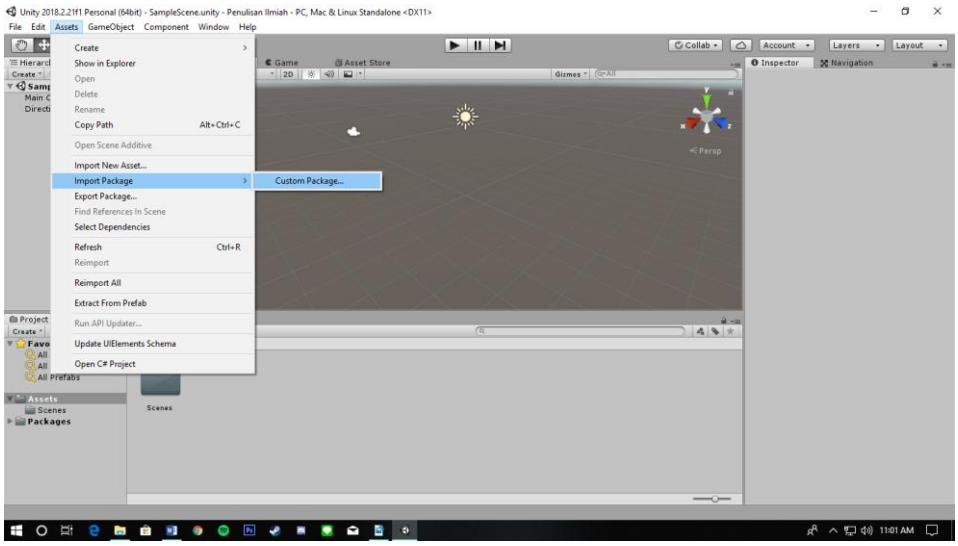


Gambar 3.10 Export Media

Pada tahap akhir pembuatan animasi, Penulis menggunakan pengaturan seperti pada gambar 3.10, yang mana *Format* nya adalah H.264 dengan *Preset custom* yaitu sesuai dengan lebar dan tinggi *Sequence* yang digunakan. Tahap pembuatan animasi, berlangsung 2 kali yaitu sampai bagian alur penjelasan, kemudian lanjut tahap pembuatan *game*, kemudian kembali ke pembuatan animasi dengan menggunakan animasi sebelumnya tetapi dengan menambahkan *video* tata cara bermainnya. Untuk *video* tata cara bermain, Penulis menggunakan aplikasi tambahan lainnya yaitu Bandicam, untuk merekam layar.

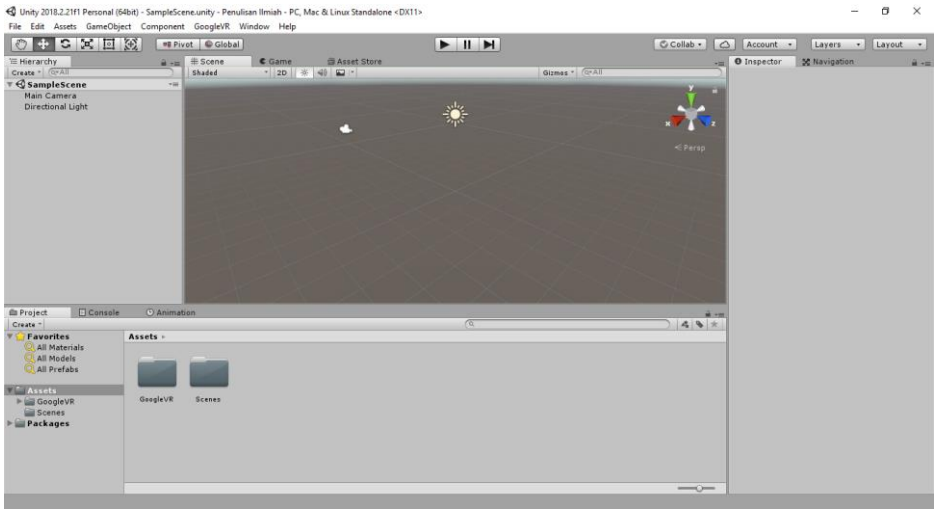
3.7.2 Tahap Pembuatan *Game*

Pada tahap pembuatan *game*, Penulis mmenggunakan aplikasi Unity dengan versi 2018.2.21f1. Penulis menggunakan versi ini karena sudah didukung dengan *assets store* yang mana fitur ini memudahkan pengguna Unity untuk menggunakan *object* yang sudah disediakan. Sebelum membuat *game*, Penulis sudah mengunduh *package* untuk pembuatan *game VR*. Package tersebut bisa di unduh pada laman resmi Google VR Developer.



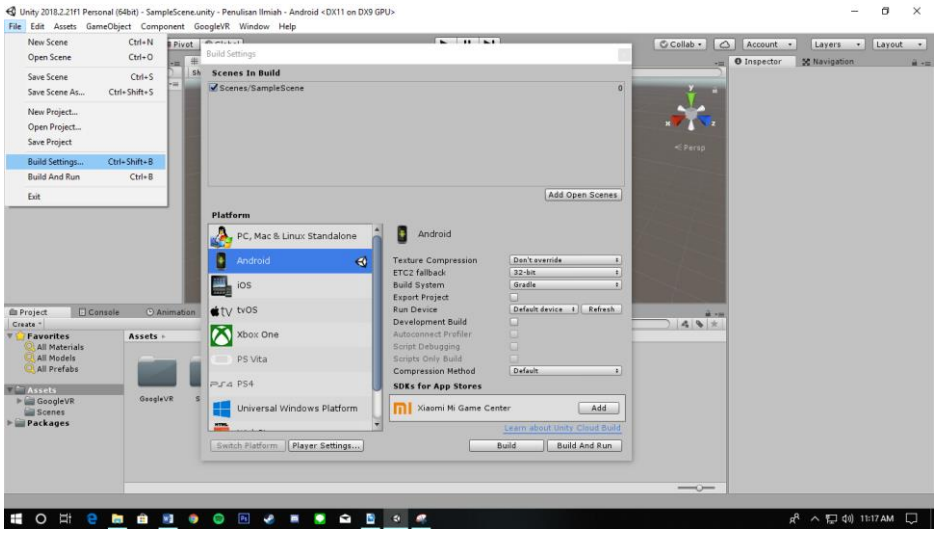
Gambar 3.11 Tampilan Import Package

Untuk melakukan *import package VR* yang sudah diunduh, yaitu seperti pada gambar 3.11, dengan menggunakan *Assets* pada *menu bar*. *Custom package* akan mengarahkan *file* mana yang akan di *import,* kemudian Penulis memilih *file* yang sebelumnya sudah diunduh yaitu dengan nama "GoogleVRForUnity_1.190.1". *Import package* berhasil apabila muncul *folder* pada tab *assets* seperti pada gambar 3.12.



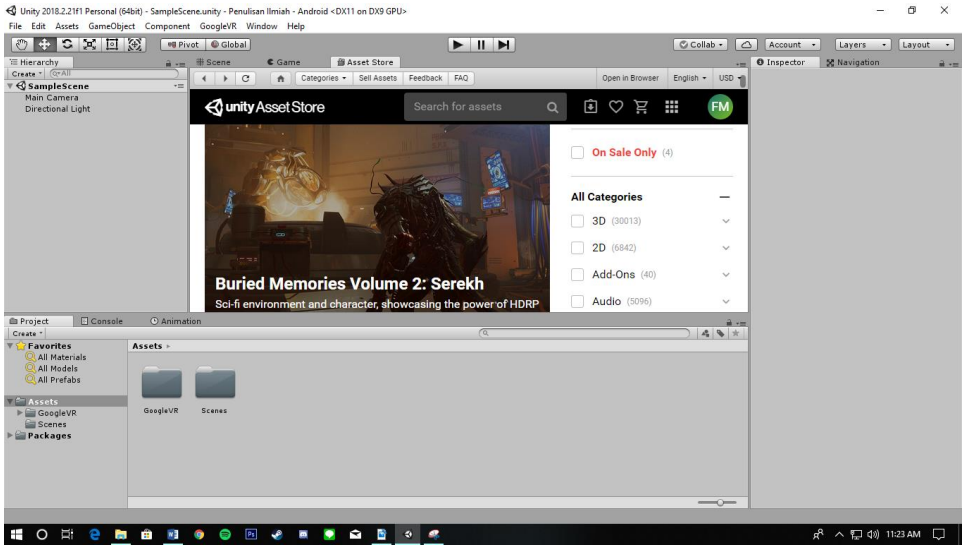
Gambar 3.12 Assets

Setelah melakukan *import package*, selanjutnya penulis mengatur *Build Settings*, hal ini dilakukan untuk menentukan *platform* dari aplikasi tersebut. Pengaturan yang penulis gunakan dapat dilihat pada gambar 3.13.



Gambar 3.13 Switch Platform

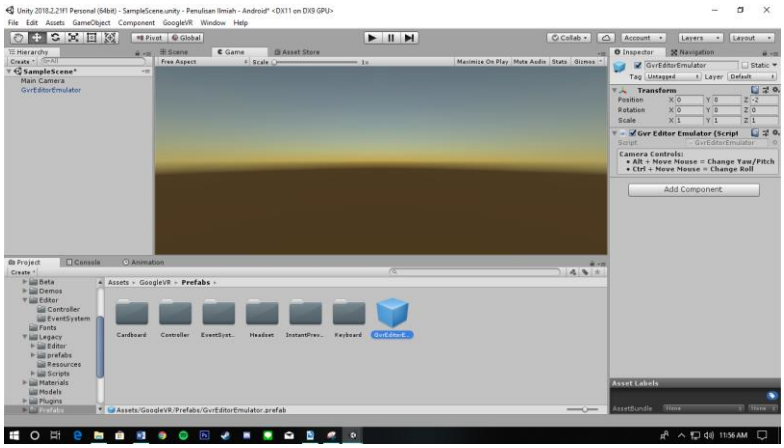
Setelah menyesuaikan *platform*, penulis juga melakukan pengaturan pada *Player Settings*, hal ini berguna untuk mengaktifkan fitur *VR*, sehingga *game* yang dibuat *compatible* untuk *VR*. Selanjutnya, penulis mencari modeling object yang telah tersedia resmi pada *asset store* Unity, seperti pada gambar 3.14. Untuk pencarian lebih ringkas, bisa menggunakan mesin pencarian Unity *Asset Store* untuk *object* yang dibutuhkan.



Gambar 3.14 Tampilan Laman Asset Store

Selanjutnya penulis membuat scene Play, yaitu scene yang digunakan sebagai permainan pada aplikasi ini. Kemudian, penulis menghapus object *light* untuk membuat suasana pada scene ini terkesan mencekam. Kemudian

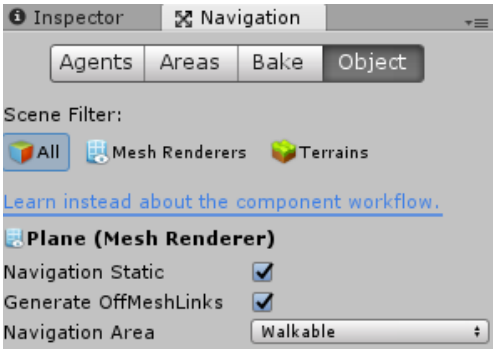
penulis menambahkan GvrEditorEmulator pada tab *hierarchy*. Prefab ini, dapat ditemukan pada folder GoogleVR seperti pada Gambar 3.15.



Gambar 3.15 PlayScene Gvr

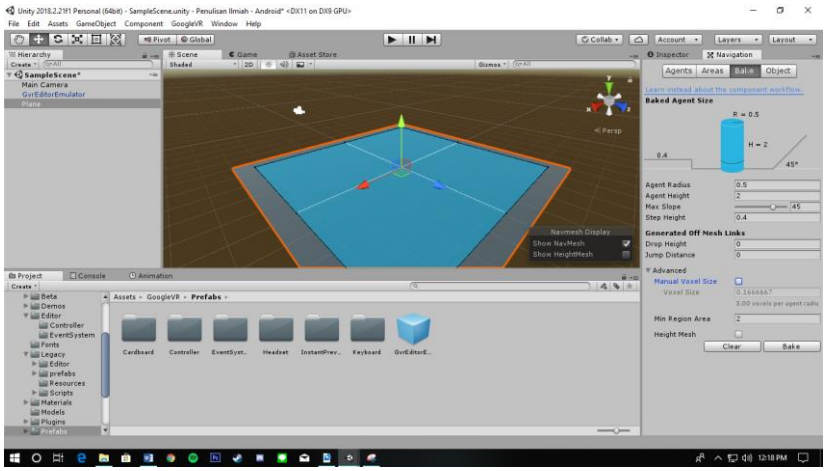
Fungsi GvrEditorEmulator sendiri adalah sebagai *controller* dari *gyroscope* yang digunakan untuk mendeteksi gerakan pada *Smartphone* yang digunakan. Kemudian penulis menambahkan plane sebagai landasan, untuk langit-langit penulis membiarkan pada kondisi defaultnya dengan material skybox.

Setelah menambahkan object plane, kemudian penulis melakukan bake pada plane untuk mengaktifkan NavMeshAgent yang mana akan berfungsi sebagai landasan bebas object untuk bergerak. Untuk menambahkannya berada pada tab Navigation seperti pada gambar 3.16.



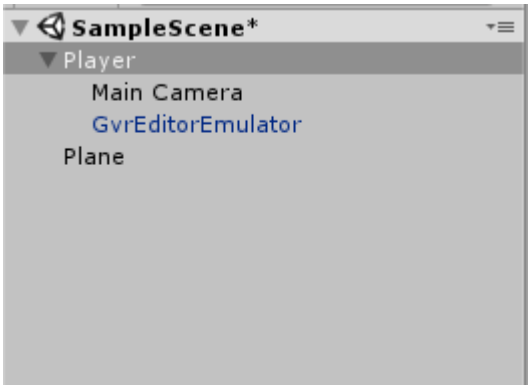
Gambar 3.16 Centang Nav

Kemudian penulis memilih Bake sehingga muncul seperti plane baru pada atas bagian plane seperti pada gambar 3.17, plane inilah yang dinamakan NavMeshAgent.



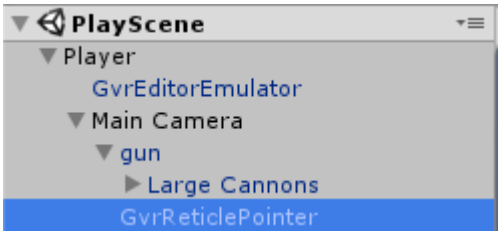
Gambar 3.17 NavMeshAgent

Kemudian penulis membuat object Player. Pembuatan object Player ini dengan cara menambahkan empty object kemudian object Main Camera dan GvrEditorEmulator di *drag and drop* ke dalamnya, hingga seperti pada gambar 3.18.



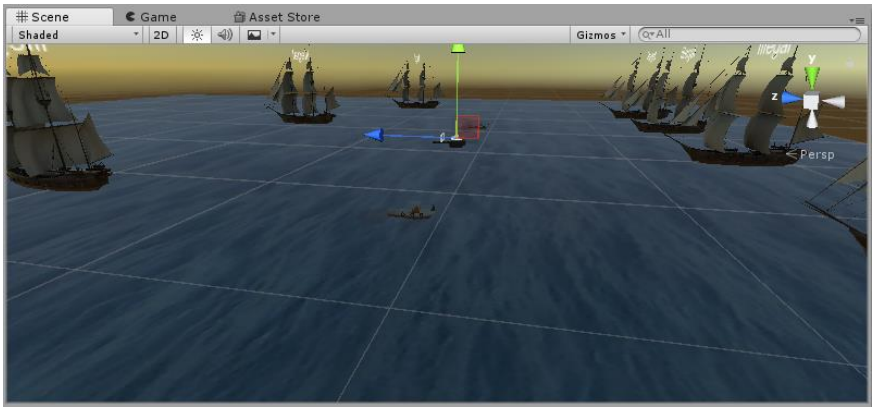
Gambar 3.18 Player Object

Setelah membuat object Player, penulis menambahkan tembakkan dan GvrReticlePointer pada Main Camera, untuk GvrReticlePointer berada pada folder GoogleVR, sub-folder Prefabs, sub-folder Cardboard, sedangkan untuk object gun merupakan object Large Turret yang berada pada folder Simple Warships. Kemudian object Large Turret penulis ganti nama menjadi gun maka object Player menjadi seperti pada Gambar 3.19.



Gambar 3.19 Player Object Final

Setelah membuat Player Object, penulis meletakkan beberapa object pendukung yaitu object pada Asset USSC Brig Sloop sebagai penjahat dan baik, kemudian object fish boat pada Asset fish_boat sebagai target/nelayan, kemudian pada object baik dan penjahat dibedakan dengan memberi petunjuk berupa text pada bagian atas kapal. Gambar 3.20 adalah hasil penambahan dan peletakkan object pada PlayScene.

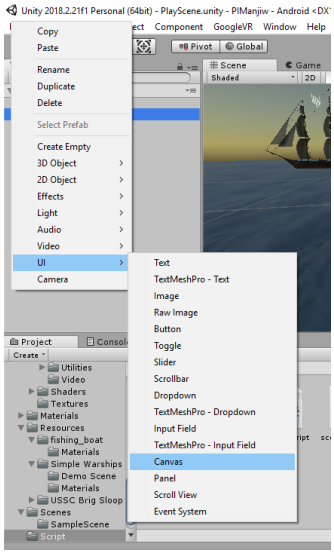


Gambar 3.20 Complete Scene Object

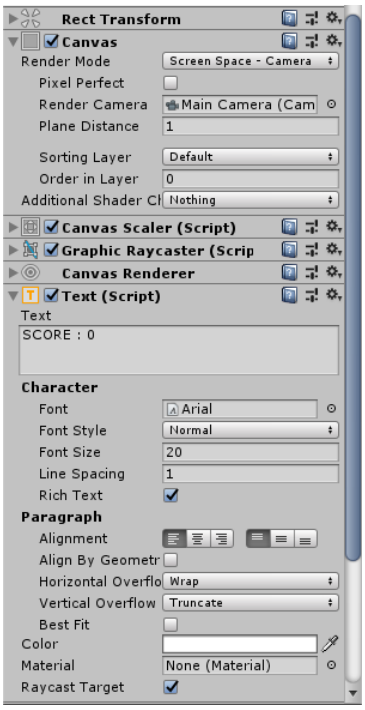
Setelah meletakkan berbagai object pada scene, penulis membuat spawnPoint pada object Player sebagai tempat untuk keluarnya peluru, dengan cara menambahkan *object sphere* pada Large Canon yang berada pada child dari Player. Selanjutnya penulis membuat sphere pada bagian luar Player atau public, dengan menyesuaikan bentuk yang tidak terlalu besar dan mengganti namanya menjadi bullet, setelah itu *drag and drop* pada asset, lalu hapus object yang berada pada scene. Untuk memudahkan peletakkan, penulis membuat folder Resources pada Asset untuk meletakkan bullet tersebut.

Setelah semua selesai, penulis melakukan pemberian *script*. *Script* yang digunakan yaitu diberi nama PlayerScript karena script ini akan

ditambahkan pada object Player. Untuk penambahan script pada object dengan cara *drag and drop* ke object gun. Selanjutnya penambahan UI untuk scoring yaitu seperti pada Gambar 3.21. Pada object canvas, penulis menambahkan *component Text* kemudian secara keseluruhan tab Inspector pada object canvas seperti pada Gambar 3.22. Sedangkan untuk EventSystem tidak perlu dilakukan perubahan.

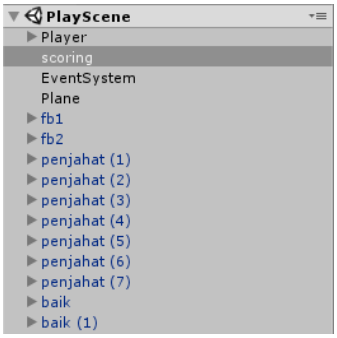


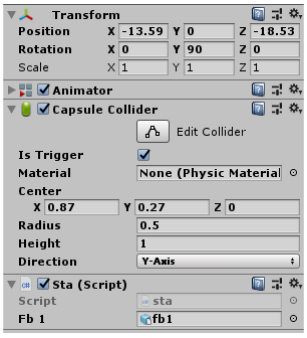
Gambar 3.21 adding UI



Gambar 3.22 Inspector Canvas

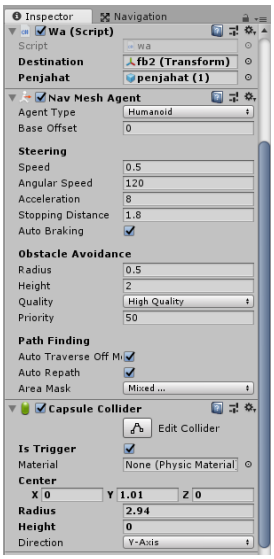
Gambar 3.23 adalah seluruh object yang penulis gunakan pada PlayScene.

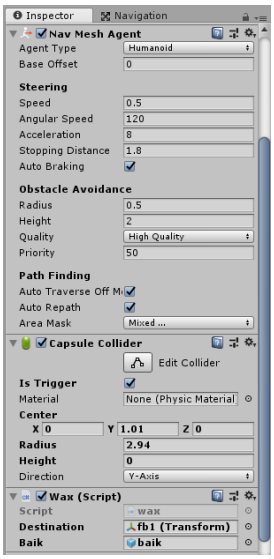




Gambar 3.23 PlayScene Object Gambar 3.24 Inspector fb1

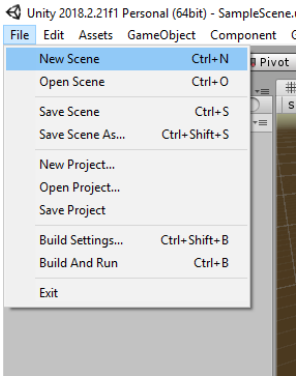
Selanjutnya, penulis menambahkan script pada object penjahat, baik dan fb. Untuk object yang duplikasi seperti penjahat (1) dengan penjahat (2), maka cukup dikenal dengan nama object yang sama yaitu penjahat. Penambahan script baiknya dilakukan sebelum duplikasi object, agar tidak perlu *drag and drop* pada tiap object. Untuk object penjahat, penulis menambahkan script bernama wa, untuk object baik, penulis menambahkan script bernama wax dan untuk object fb1 dan fb2 yaitu script dengan nama sta dan stb. Selanjutnya, penulis melakukan perubahan pada Inspector tiap-tiap object, seperti pada gambar 3.24 untuk object fb1, gambar 3.25 untuk Inspector object baik, gambar 3.26 untuk Inspector object penjahat.





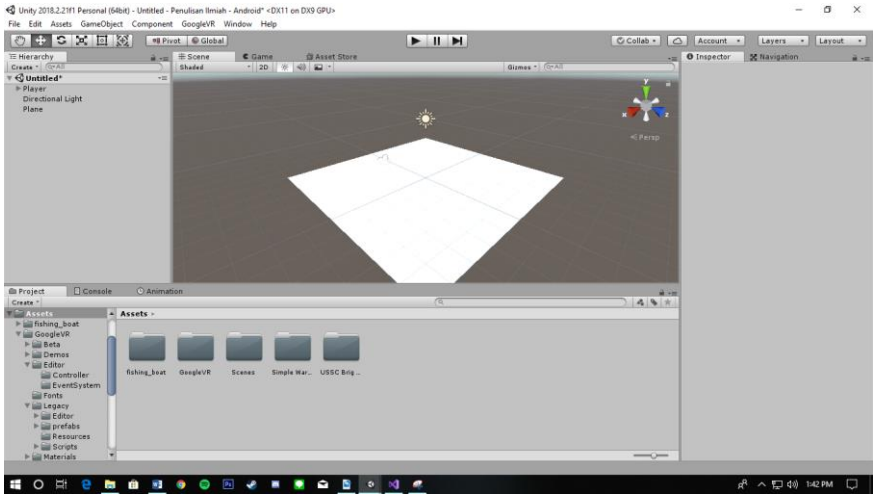
Gambar 3.25 Inspector Baik Gambar 3.26 Inspector Penjahat

Setelah membuat PlayScene, selanjutnya penulis melakukan pembuatan MenuScene. Untuk pembuatan scene baru, penulis memilih New Scene, seperti pada Gambar 3.27.



Gambar 3.27 New Scene

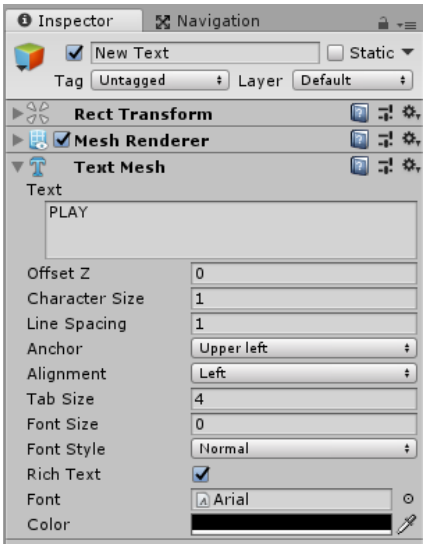
Setelah membuat scene baru, penulis mengganti nama scene dengan cara save as, dan diberi nama MenuScene. Penulis kemudian melakukan copy object Player dari PlayScene ke MenuScene. Penulis juga menambahkan *plane* dan menghapus Main Camera yang berada pada luar object Player, maka hierarchy dan viewer akan seperti Gambar 3.28.



Gambar 3.28 Scene Menu Awal

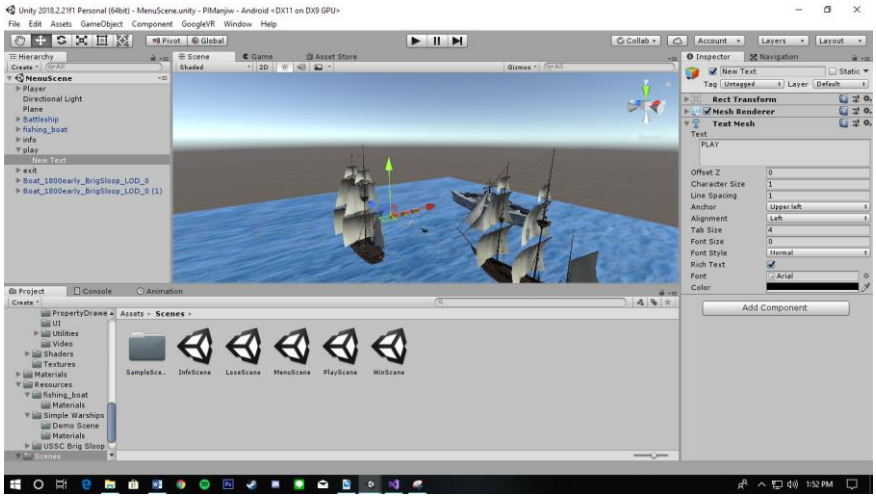
Pada scene ini tidak perlu dilakukan *bake* untuk NavMeshAgent, karena pada scene ini yang bergerak hanya *Pointer* untuk mengarahkan ke *box*. Selanjutnya pembuatan *box* sebagai *object trigger* untuk pindah *scene*.

Penulis menambahkan *3d object cube*, lalu menambahkan 3d *object text*, kemudian pada *tab Inspector*-nya seperti pada Gambar 3.29.



Gambar 3.29 Inspector Text pada Cube

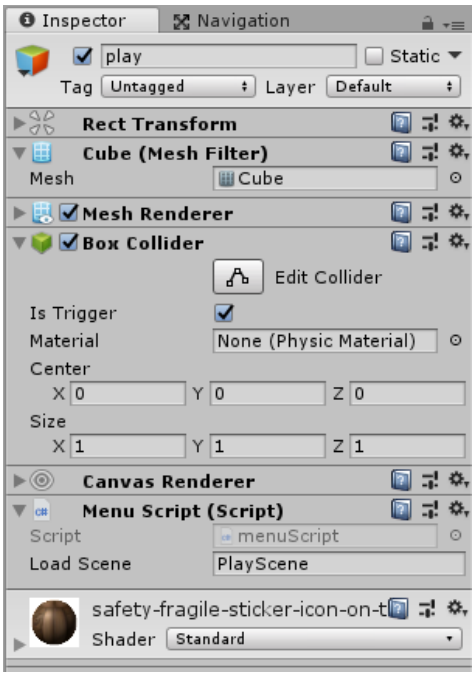
Setelah itu, penulis membuat 2 *cube* lainnya dengan *Text*-nya masing-masing yaitu INFO dan EXIT, terakhir Penulis menambahkan *texture* pada *box* dan *plane* serta beberapa *object* dari *asset*. Untuk keseluruhan MenuScene dapat dilihat pada gambar 3.30.



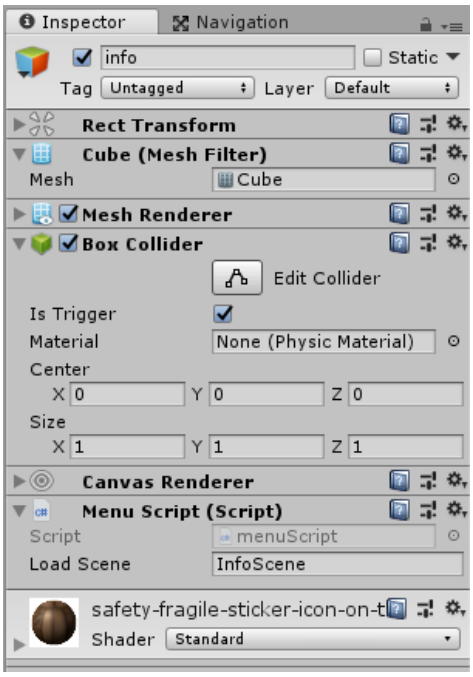
Gambar 3.30 Viewer dan Hierarchy MenuScene

Langkah terakhir pada scene ini yaitu pemberian script pada *box* Play, *box* Info dan *box* Exit. *Script* yang digunakan yaitu *script* dengan nama menuScript, *script* ini digunakan untuk kedua *object box* yang mana dari

script ini digunakan serializefield sehingga *loadscene* bisa ditentukan, sedangkan untuk *box* Exit menggunakan script yang bernama keluar, berikut inspector script pada tiap-tiap box yaitu Gambar 3.31 Inspector Box Play dan Gambar 3.32 Inspector Box Info.



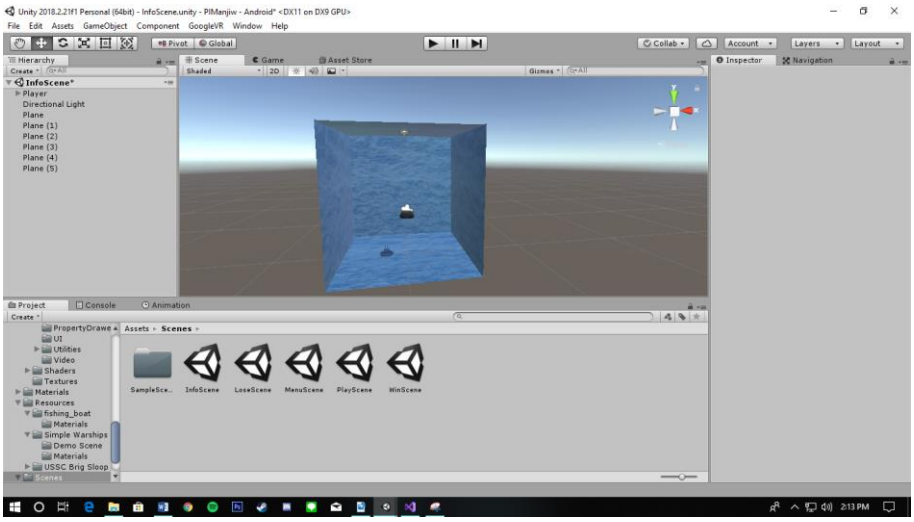
Gambar 3.31 Inspector Box Play



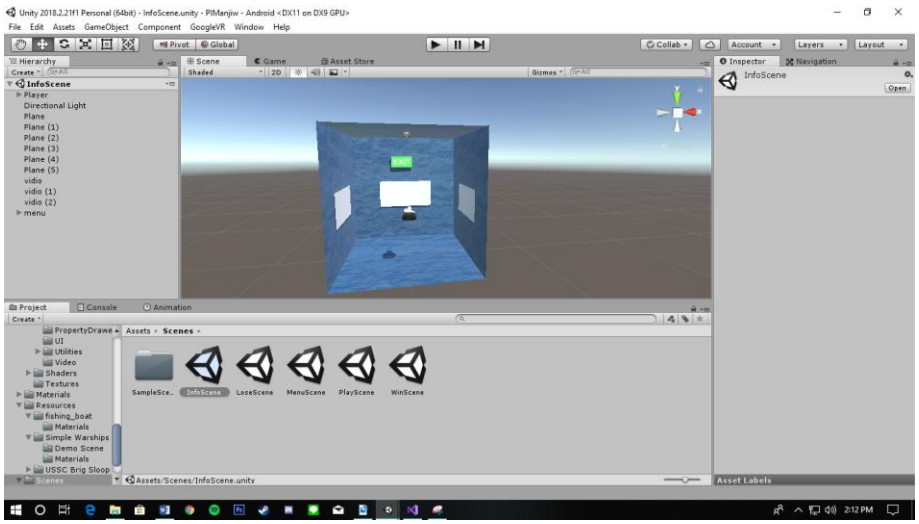
Gambar 3.32 Inspector Box Info

Selanjutnya adalah pembuatan InfoScene, yaitu buat scene baru dan gunakan object Player yang sudah dibuat pada PlayScene. Pada scene ini, penulis juga melakukan *copy object* dari MenuScene untuk *object box* Exit namun scriptnya diganti menjadi menuScript lalu pilih *loadscene*-nya menjadi MenuScene.

Kemudian, penulis membuat ruang dengan menggunakan plane, hingga seperti pada Gambar 3.33. Untuk layar pemutar *video* animasi, penulis menambahkan *plane* baru dan pada bagian atas layar pemutar *video* animasi, letakkan *box* Exit di atasnya hingga seperti Gambar 3.34. *Box* Exit ini bisa diambil dari MenuScene sehingga tidak perlu membuat ulang karena inspectornya sama dan scriptnya sama.

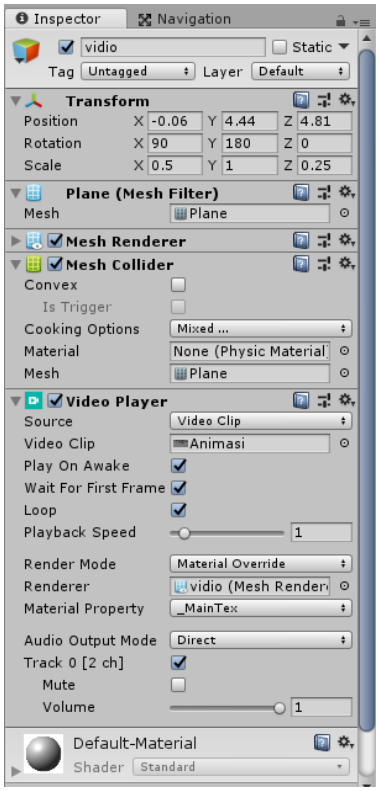


Gambar 3.33 Buat Ruang InfoScene



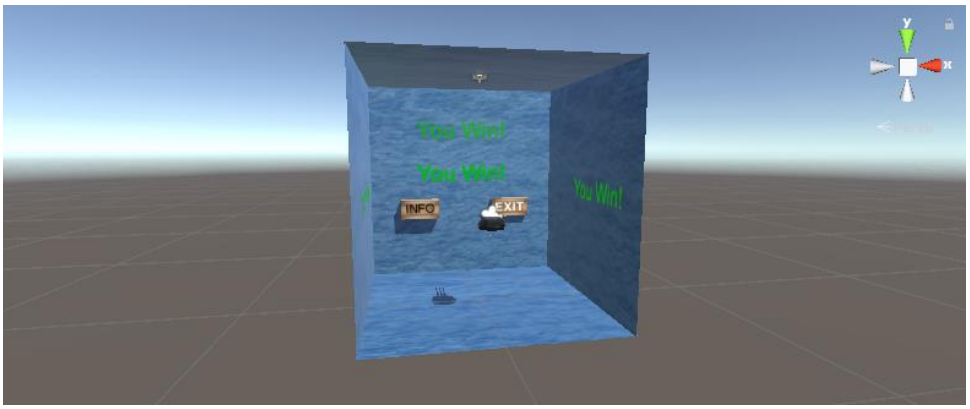
Gambar 3.34 Complete InfoScene

Untuk *plane* yang memutarkan vidio yaitu *object* vidio, inspectornya dapat dilihat pada Gambar 3.35. Untuk menambahkan component *Video Player*, penulis melakukan *add component* dengan mengetik *Video Player*, dan letakkan animasi *Video Clip*. Sebelumnya vidio yang akan diputarkan telah di-*import* ke *assets*. Untuk melakukan *import* pada *asset* unity cukup dengan *drag and drop*.



Gambar 3.35 Inspector Plane Vidio

Selanjutnya, pembuatan *scene* WinScene. Pada scene ini, penulis menggunakan ruang yang sudah dibuat pada InfoScene, hapus beberapa object hingga seperti Gambar 3.36, kemudian *save as* dengan memberi nama WinScene. Untuk WinScene *object* yang digunakan yaitu *box* Info, *box* Exit dan *Text*. *Box* Info yang digunakan, menggunakan *box* Info pada MenuScene, sedangkan *box* Exit yang digunakan, menggunakan *box* Exit yang pada *scene* InfoScene. Pada *Text* penulis mengganti dengan "*You Win!*".



Gambar 3.36 Complete WinScene

Scene terakhir yang penulis buat adalah LoseScene, *scene* ini menggunakan ruang dan object pada WinScene dengan melakukan sedikit perubahan yaitu pada scene ini *box* Info diganti dengan *box* Play, dan *Text* diganti menjadi "*You Lose!*". Untuk *Box* Play, penulis me-*copy* dari MenuScene. Gambar 3.37 adalah keseluruhan *scene* LoseScene.

Gambar 3.37 Complete LoseScene

3.8 Penjelasan Script

Pada pembuatan aplikasi ini digunakan 7 script yaitu playerScript, menuScript, scoreManager, wa, wax, sta, keluar. Berikut penjelasan tiap-tiap scriptnya. Untuk wa dan wax, *script* ini digunakan untuk pindahnya *object* penjahat dan baik, sedangkan sta untuk *object fishing boat.*

3.8.1 PlayerScript

Penggunaan playerScript pada aplikasi ini yaitu dikhususkan untuk *object* Player. Adapun fungsi utamanya untuk mengaktifkan *trigger* pada *collider* sehingga pada saat Pointer yang berada pada object player saat mengenai *object* lain seperti kapal ataupun *box* memicu suatu aksi.

```
using UnityEngine;
using System.Collections;

public class playerScript : MonoBehaviour
{
    private GameObject gun;
    private GameObject spawnPoint;
    private bool isShooting;
    void Start()
    {
        gun =
gameObject.transform.GetChild(0).gameObject;
        spawnPoint =
gun.transform.GetChild(0).gameObject;
        isShooting = false;
    }
```

Code yang diawali dengan kata “using” digunakan untuk *import library* yang sudah tersedia di Unity. Kemudian setelah melakukan *import* langkah selanjutnya adalah inisiasi kelas yaitu “public class playerScript”, sedangkan “MonoBehaviour” akan otomatis tercipta dan berfungsi sebagai kendali ke *inspector object*. Setelah melakukan inisiasi kelas, *line* dibawahnya adalah inisiasi *object* yaitu apabila dengan *private* maka hanya akan dikenali pada kelas tersebut, sedangkan jika *public* dikenali diseluruh kelas. Inisiasi *gameobject* ini dilakukan untuk menghubungkan dengan 3d *gameobject* yang telah dibuat.

Selanjutnya setalah void Start adalah *line function* utama, yaitu “gun=gameObject.transform.GetChild(0).gameObject;”, gun merupakan nama gameObject yang kita buat, sehingga pada code ini digunakan untuk mencari si gameObject tersebut yang mana gameObject tersebut sedangkan GetChild(0) untuk mencari object gun berdasar index dari gameobject script tersebut digunakan. Kemudian line dibawahnya sama tetapi untuk gameObject spawnPoint, dan indexnya berdasar lokasi gun. Selanjutnya function isShooting dijadikan false agar tidak langsung aktif, function ini bertujuan untuk menembakkan peluru yaitu gameObject bullet.

```
IEnumerator Shoot()
    {
        isShooting = true;
        GameObject bullet = Instantiate(Resources.Load("bullet",
typeof(GameObject))) as GameObject;
        Rigidbody rb = bullet.GetComponent<Rigidbody>();
        bullet.transform.rotation = spawnPoint.transform.rotation;
        bullet.transform.position = spawnPoint.transform.position;
        rb.AddForce(spawnPoint.transform.forward * 1000f);
        Destroy(bullet, 1);
        yield return new WaitForSeconds(1f);
        isShooting = false;
    }
```

IEnumerator merupakan function dari Coroutine, yang mana fungsinya untuk mengembalikan nilai sedangkan coroutine sendiri berfungsi untuk memberikan transparansi pada frame. Pada function diatas, apabila kondisinya true, maka gameObject bullet akan keluar dari

gameObject spawnPoint, resources disini merupakan tempat bullet tersebut disimpan.

Kemudian, pada bullet yang sudah dibuat ditambahkan component Rigidbody, Rigidboy disini sendiri membuat reaksi fisika yaitu jika diberi aksi maka akan menghasilkan reaksi. Apabila rigidbody tidak ditambahkan pada bullet, bullet tersebut tidak akan bergerak, namun karena bullet tersebut diberikan rigidbody maka apabila diberikan aksi dia akan menghasilkan reaksi.

Dua line setelahnya merupakan penyesuaian location dan rotation pada spawnPoint dan Bullet agar sesuai. Line dibawahnya yaitu rb addforce, merupakan pemberian aksi, yang mana benda tersebut diberikan 1000f, yaitu f disini adalah frame dan mengarah forward, arah forward ini sendiri diatur melalui Scene Viewer. Kemudian line destroy untuk menghancurkan bullet pada saat mengenai collider lain. Kemudian line setelahnya menggunakan salah satu fungsi coroutine yaitu untuk memberikan transparansi frame sebanyak 1 kali, kemudian function itu bernilai false.

```
void Update()
    {
        RaycastHit hit;
        Debug.DrawRay(spawnPoint.transform.position,
spawnPoint.transform.forward, Color.green);
        if (Physics.Raycast(spawnPoint.transform.position,
spawnPoint.transform.forward, out hit, 5000))
        {
            if (hit.collider.name.Contains("penjahat"))
            {
                if (!isShooting)
                {
                    StartCoroutine("Shoot");
                }
            }
```

Void update digunakan untuk mengeksekusi perintah yang selalu berubah sesuai kondisi. Berhubung method ini akan terus mengeksekusi

```
else if (hit.collider.name.Contains("baik"))
            {
                if (!isShooting)
                {
                    StartCoroutine("Shoot");
                }
            }
```

program, maka variable didalamnya biasanya ikut berubah, misalnya pada coroutine, yang mana hanya akan menjalankan function isShoting pada saat Raycast mengarah ke gameObject yang bernama penjahat. Dalam penggunaan raycast float max distance perlu diperhitungkan, disini penulis menggunakan float 5000 agar bisa lebih jauh yang mana pada percobaan sebelumnya menggunakan 100, bullet yang keluar tidak mengenai collider gameobject lain. Untuk else if lainnya digunakan untuk trigger ke game object lainnya, sehingga terjadi collider dan menimbulkan suatu aksi tertentu.

Pada dasarnya script ini digunakan sebagai trigger saja, sehingga belum ada kondisi yang terjadi apabila terjadi collider, kecuali destroy bullet, tetapi jika script ini tidak ada, maka *game* tidak dapat berjalan dan hanya akan menjadi *game* yang tidak mempunya aksi.

3.8.2 MenuScript

Pada script ini digunakan untuk melakukan perpindahan scene, namun script ini tidak bisa berjalan sendiri karena memerlukan trigger dari PlayerScript.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class menuScript : MonoBehaviour {
    [SerializeField] private string loadScene;
       void OnTriggerEnter(Collider col)
    {
        SceneManager.LoadScene(loadScene);
    }
}
```

Pada script ini diperlukan import library SceneMangement untuk mengatur perpindahan scene. Pada script ini terdapat method Serializefield yang mana fungsinya untuk menentukan ke suatu gameObject ataupun scene. Pada script ini tidak tertulis secara implisit kemana scene ini akan berpindah bila terkena trigger, karena perpindahannya dapat ditentukan

melalui Inspector, yang mana kendali pada Inspector gameObject ini merupakan fungsi dari MonoBehaviour. Script ini digunakan pada box Play dan box Info pada MenuScene, serta box Exit pada InfoScene sedangkan box Exit pada MenuScene menggunakan script yang berbeda.

3.8.3 Script: keluar

Script ini hanya digunakan pada box Exit yang ada pada MenuScene, sedangkan box Exit pada InfoScene dan Win/Lose Scene adalah yang berbeda. Untuk penggunaan *library*-nya tidak perlu menambahkan *SceneManagement*.

```
public class keluar : MonoBehaviour {
    void OnTriggerEnter(Collider col)
    {
        Application.Quit();
    }
}
```

Script ini digunakan untuk keluar dari *game* atau menutup aplikasi yang sedang dijalankan.

3.8.4 Script: scoreManager

Pada script ini berfungsi sebagai parameter menang atau kalah pada PlayScene, dan juga sebagai SceneManagement untuk perpindahan ke WinScene apabila kondisi menang terpenuhi atau perpindah ke LoseScene apabila kondisi kalah terpenuhi. Script ini dikaitkan dengan object canvas yang bernama scoring pada PlayScene.

```
public class scoreManager : MonoBehaviour {
    public static int score;
    Text text;
       void Awake () {
        text = GetComponent<Text>();
        score = 0;}
       void Update () {
        text.text = "SCORE : " + score;
        if(score >= 250)
        {
            SceneManager.LoadScene("WinScene");}
        else if(score < 0)
        {
            SceneManager.LoadScene("LoseScene");}}}
```

Berhubung script ini digunakan untuk perpindahan scene, maka diperlukan import library SceneManagement. Inisiasi variable score dan text sebagai text. Penggunaan method awake yaitu sebagai method yang akan dieksekusi terlebih dahulu dari method lainnya. Inisiasi score awal yaitu 0, dan variable text menggunakan function text. Selanjutnya method update yang mengeksekusi secara terus menerus, berisikan perubahan pada score dan kondisi, yang mana apabila score mencapai lebih dari sama dengan 250 maka akan pindah ke WinScene, sedangkan apabila score kurang dari 0 maka akan pindah ke LoseScene.

3.8.5 Script: wa

Penamaan wa sendiri merupakan kepanjangan dari walk agent. Script ini digunakan oleh object kapal yang bernama penjahat. Pada script ini apabila terkena trigger dari script Player maka akan memberikan perubahan score pada script scoreManager.

```
public class wa : MonoBehaviour {
    [SerializeField]
    Transform _destination;
    NavMeshAgent _navMeshAgent;
    public GameObject penjahat;
    public static int scoreValue = 50;
       void Start () {
        _navMeshAgent = this.GetComponent<NavMeshAgent>();
        if(_navMeshAgent == null)
        {
            Debug.LogError("The nav mesh agent component is not
attached to " + gameObject.name); }
        else
        {
            SetDestination(); }  }
    private void SetDestination()
    {
        if(_destination != null)
        {
            Vector3 targetVector =
_destination.transform.position;
            _navMeshAgent.SetDestination(targetVector); } }
    void OnTriggerEnter(Collider col)
    {
        GetComponent<CapsuleCollider>().enabled = false;
        Destroy(col.gameObject);
        scoreManager.score += scoreValue;
        Destroy(gameObject, 1);}}
```

Penggunaan serializefield pada script ini untuk menentukan tujuan object ini berjalan. Inisiasi gameobject penjahat dan scorevalue pada object tersebut. Method start untuk mengeksekusi navmesh yang sudah dibuat, kemudian dengan kondisi mengarah ke tujuan yang ditentukan menggunakan serializefield. Kemudian method dari SetDestination sendiri berfungsi untuk mengarahkan object ke arah tujuan. Terakhir tambahkan method trigger, yang mana apabila pointer Player mengarah ke object akan terjadi trigger dan mengaktifkan method tersebut. Method trigger ini akan mematikan capsule collider pada object, melakukan terminate object kapal dan bullet serta penambahan skor pada object score di sript scoreManager.

3.8.6 Script: wax

Pada script ini berfungsi sama seperti script wa pada sub bab 3.7.5, hanya saja yang jadi pembeda adalah scorevalue-nya, apabila di scrip wa score akan bertambah sedangkan pada script ini akan berkurang. Untuk *library* yang digunakan sama dengan *script* wa.

```
    void OnTriggerEnter(Collider col)
    {  GetComponent<CapsuleCollider>().enabled = false;
        Destroy(col.gameObject);
        scoreManager.score -= scoreValue;
        Destroy(gameObject, 1); }}
```

Gameobject pada script ini bernama baik, pembeda dari script wa terlihat pada method trigger yang mana scoreManager.score akan berkurang sedangkan di wa akan bertambah.

3.8.7 Script: sta

Pada script ini akan dikaitkan pada gameobject fb1 yang merupakan sebagai tujuan dari navmesh gameobject penjahat ataupun baik.

```
public class sta : MonoBehaviour
{
    public GameObject fb1;
    public static int scoreValue = 500;
    void OnTriggerEnter(Collider col)
```

Script ini pada dasarnya sama seperti wax pada sub bab 3.7.6, hanya saja gameobject ini bersifat *stationary*. Pengurangan jika pointer mengenai

gameobject ini adalah 500, sehingga pasti akan kalah jika pointer mengenai gameobject ini.

3.9 Implementasi Aplikasi

Pada bagian ini tahap yang dilakukan adalah mengimplementasikan aplikasi yang telah berekstensi .apk.

Berikut adalah hasil dari Uji Coba Aplikasi Tenggelamkan berbasis Android dengan menggunakan perangkat *smartphone*:

Tabel 3.1 Hasil Uji Coba Aplikasi di *Smartphone* yang Berbeda

No	Tipe *Smartphone*	Spesifikasi	*Installasi*	Sesuai Gerakan	Error
1	Redmi Note 4x	**Android OS** 6.0 (Marshmallow) **Resolusi Layar** 1080 x 1920 piksel **Processor** Octa core 2.0 GHz Cortex A53	Berhasil	Sesuai	Tidak Ada
2	OPPO A3S	**Android OS** 8.1 (Oreo) **Resolusi Layar** 1520 x 720 piksel **Processor** Octa core 1.8 GHz Cortex A53	Berhasil	Sesuai	Tidak Ada

Tabel 3.1 Hasil Uji Coba Aplikasi di *Smartphone* yang Berbeda

No	Tipe *Smartphone*	Spesifikasi	*Installasi*	Sesuai Gerakan	Error
3	Lenovo Vibe K5	**Android OS** 5.1 (Lollipop) **Resolusi Layar** 720 x 1280 piksel **Processor** Octa core (4x1.4 GHz Cortex A53 & 4x1 GHz Cortex A53)	Berhasil	Sesuai	Tidak Ada
4	IMO B6S	**Android OS** 5.1 (Lollipop) **Resolusi Layar** piksel **Processor** Quad core 1.3 GHz Cortex A7	Berhasil	Tidak	Tidak memiliki *Gyroscope*
5	Redmi 5A	**Android OS** 7.1 (Nougat) **Resolusi Layar** 1280 x 720 piksel **Processor** Quad core 1.4 GHz Cortex A53	Berhasil	Tidak	Tidak memiliki *Gyroscope*

Dapat dilihat dari Tabel 3.1, beberapa *smartphone* yang diujikan berhasil melalui tahap *installasi*, namun terdapat *error*, *error* yang didapat

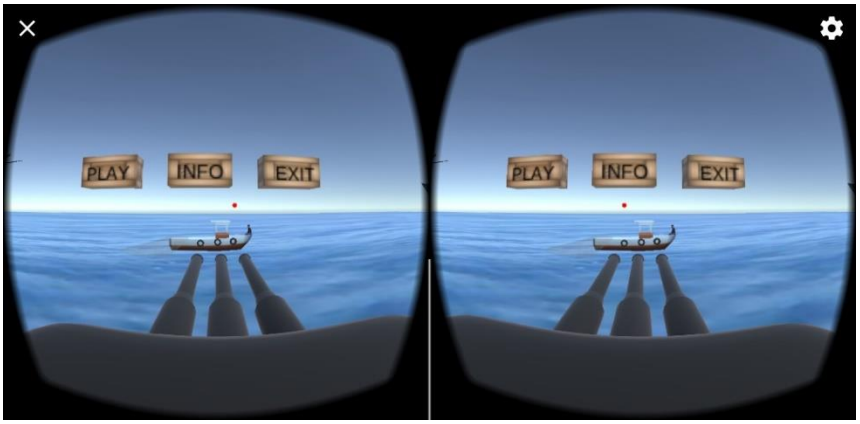
sangat fatal yaitu pada saat dijalankan aplikasi hanya diam tidak ada reaksi saat adanya gerakan, hal ini diduga karena *smartphone* tersebut tidak mendukung *gyroscope* sehingga tidak memproses sensor gerakan, meskipun aplikasi tersebut dapat ter-*install* dengan benar.

3.10 Uji Coba Aplikasi

Setelah program di*build* hingga menghasilkan ekstensi file .apk, selanjutnya aplikasi bisa di*install* pada *smartphone* dan dijalankan. Pada saat menjalankan aplikasi, maka pengguna aplikasi akan diperlihatkan tampilan *splashscreen* terlebih dahulu seperti pada Gambar 3.38.

Gambar 3.38 Splash Screen Apk

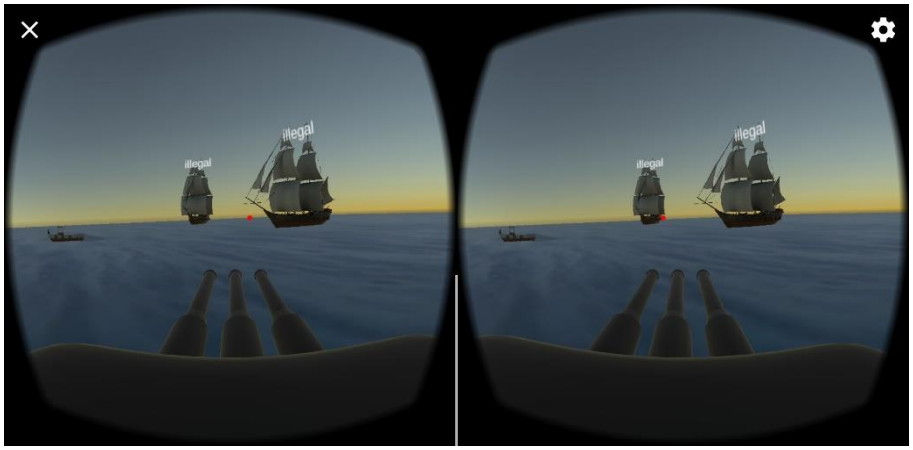
Tampilan *splashscreen* akan berlangsung selama 2 detik dan berisikan informasi nama aplikasi yaitu "*Tenggelamkan"*. Setelah tampilan *splashscreen,* selanjutnya pengguna akan masuk ke *scene* utama yaitu MenuScene seperti pada Gambar 3.39.



Gambar 3.39 MenuScene Apk

Pada MenuScene, pengguna aplikasi dihadapkan dengan 3 pilihan yaitu Play untuk bermain, Info untuk menonton vidio animasi, dan Exit untuk keluar. Pada Gambar 3.40 adalah tampilan pada PlayScene jika pengguna memilih Play pada MenuScene.

Pada Gambar 3.40, pengguna aplikasi masuk ke dalam PlayScene, yang mana pada aplikasi ini pengguna aplikasi harus mendapatkan skor 250 untuk memenangkan permainan atau mendapat skor dibawah 0 dan kalah.

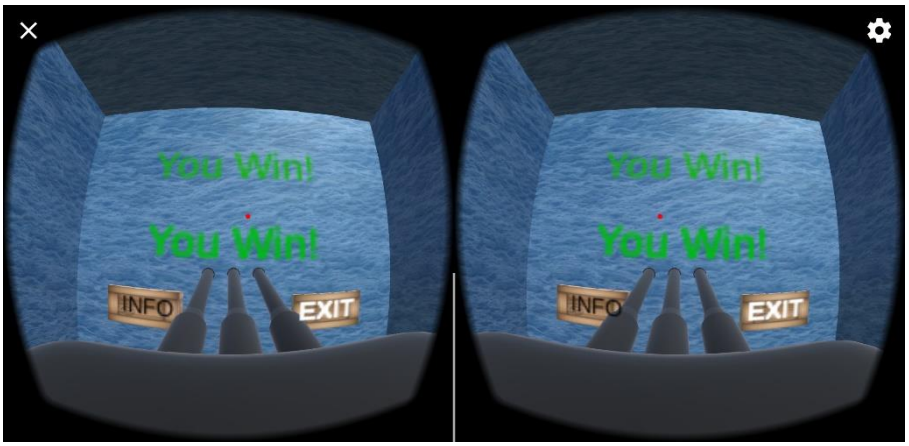


Gambar 3.40 PlayScene Apk

Pada Gambar 3.41 adalah tampilan LoseScene jika pengguna aplikasi mendapat skor dibawah 0. Pada *scene* LoseScene pengguna aplikasi akan dihadapkan pada dua pilihan yaitu Play untuk bermain kembali dan Exit untuk kembali ke menu utama.

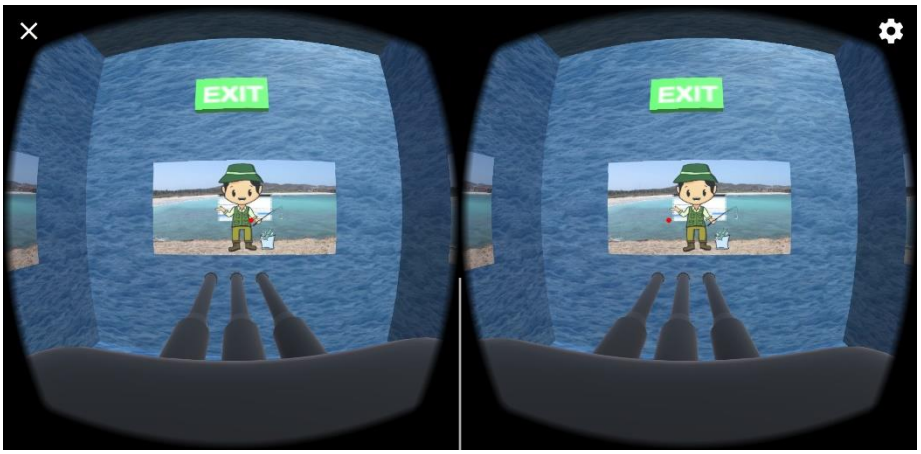
Gambar 3.41 LoseScene Apk

Pada Gambar 3.42 adalah tampilan WinScene jika pengguna aplikasi mendapat skor 250. Pada *scene* WinScene pengguna aplikasi akan dihadapkan pada dua pilihan yaitu Info untuk menonton *video* animasi dan Exit untuk kembali ke menu utama.



Gambar 3.42 WinScene Apk

Pada Gambar 3.43 adalah tampilan InfoScene, dimana pada tampilan ini pengguna aplikasi akan disajikan *video* animasi terkait informasi proses penenggelaman kapal dan diakhir terdapat *video* cara bermain.



Gambar 3.43 InfoScene Apk

Pada Gambar 3.43, *video* yang ditampilkan bersifat *looping* sehingga apabila pengguna sudah melihatnya dapat memilih Exit, untuk kembali ke menu utama. Pada menu utama jika memilih exit maka aplikasi akan tertutup.

3.11 Olah Data Kuesioner

3.11.1 Karakteristik Responden

Pada bagian ini dilakukan analisis deskriptif terhadap karakteristik responden yang digunakan dalam penelitian ini yaitu masyarakat umum, yang meliputi jenis pekerjaan dan usia. Karakteristik tersebut diharapkan dapat memberi gambaran tentang keadaan responden. Untuk lebih jelasnya dapat disajikan sebagai berikut :

1. Jenis Pekerjaan/Profesi

Tabel 3.2 Pekerjaan/Profesi Responden

No.	Pekerjaan/Profesi	Jumlah	Persen
1	Mahasiswa/i	26	86.67%
2	Siswa/i	2	6.67%
3	Pegawai Swasta	1	3.33%
4	Admin	1	3.33%
Total		30	100%

Berdasarkan tabel 3.2 di atas, diketahui pekerjaan/profesi yang mencoba aplikasi dan mengisi kuesioner untuk aplikasi penulis kebanyakan adalah Mahasiswa/i (86.67 %).

2. Usia

Tabel 3.3 Usia Responden

No.	Usia	Jumlah	Persen
1	16 Tahun	1	3.33%
2	18 Tahun	7	23.33%
3	19 Tahun	9	30%
4	20 Tahun	6	20%
5	21 Tahun	4	13.33%
6	22 Tahun	2	6.67%
7	26 Tahun	1	3.33%
Total		30	100%

Berdasarkan tabel 3.3, diketahui responden kebanyakan adalah berusia 19 tahun (30%).

3.11.2 Kuesioner Uji Coba Pengguna

Hasil uji coba pada sisi pengguna menggunakan kuesioner dengan jumlah 30 responden sebagai penilaian terhadap aplikasi ini. Dalam kuisioner ini memberikan 10 pertanyaan dan 4 pilihan jawaban yakni Tidak Baik (TB), Cukup Baik (CB), Baik (B), dan Sangat Baik (SB). Berikut adalah pertanyaan kuisioner dan jawaban dari 30 responden seperti pada tabel 3.4. :

Tabel 3.4 Jawaban Responden

No	Pertanyaan	Jawaban			
		Tidak Baik	Cukup Baik	Baik	Sangat Baik
1	Bagaimana kesesuaian penggunaan warna dan desain latar belakang aplikasi ini?	0%	56.7%	36.6%	6.7%
2	Bagaimana ketepatan ukuran tulisan pada aplikasi ini?	0%	26.7%	60%	13.3%
3	Apakah permainan mudah dimengerti?	0%	16.7%	40%	43.3%
4	Bagaimana ketepatan fungsi tombol dengan tujuan yang diinginkan?	0%	30%	53.3%	16.7%
5	Apakah tampilan animasi jelas?	0%	40%	53.3%	6.7%
6	Bagaimana kesesuaian ilustrasi musik dalam mendukung aplikasi?	3.3%	30%	56.7%	10%

Tabel 3.4 Jawaban Responden

No	Pertanyaan	Jawaban			
		Tidak Baik	**Cukup Baik**	**Baik**	**Sangat Baik**
7	Bagaimana ketepatan penyampaian materi video animasi terkait proses penenggelaman kapal?	0%	23.3%	66.7%	10%
8	Bagaimana kemudahan dalam pengoperasian aplikasi?	0%	26.7%	50%	23.3%
9	Apakah anda dapat mengerti mengenai kampanye "Stop Illegal Fishing"?	0%	16.7%	46.7%	36.7%
10	Apakah anda tertarik untuk mempelajari/aktif dalam kampanye "stop illegal fishing"?	3.3%	26.7%	33.3%	36.7%

3.11.3 Skala Likert

Dari hasil nilai yang didapat, maka penelitian terhadap 30 responden dapat dikelompokkan dengan cara berikut :

• Nilai tertinggi = Total responden x Bobot terbesar

= 30 x 4

= 120

• Nilai terendah = Total responden x Bobot terkecil

= 30 x 1

= 30

• Jarak = Nilai tertinggi – Nilai terendah

= 120 – 30

= 90

• Interval = Jarak / Banyaknya kelas

= 90 / 4

= 22.5

Kelas interval didapat dengan cara berikut:

Nilai terendah dimasukkan pada interval terakhir yaitu tidak baik sebesar 30 dengan interval 22.5 yang diambil dari jarak dibagi dengan banyaknya kelas menjadi 30 + 22.5 = 52.5, sampai pada nilai tertinggi yang didapat sebesar 120 dengan penilaian sangat baik. Maka dapat ditentukan pengelompokkan penilaian yang bisa di lihat pada tabel 3.5.

Tabel 3.5. Penilaian dan Kelas Interval

Kategori	Interval
Tidak Baik (TB)	30 – 52,4
Cukup Baik (CB)	52,5 – 74,9
Baik (B)	75 – 97,4
Sangat Baik (SB)	97,5 - 120

3.11.4 Hasil Jawaban Responden

Setelah dilakukan penelitian dengan menyebarkan kuesioner kepada 30 responden, maka didapatkan hasil jawaban responden yaitu sebagai berikut:

3.11.4.1 Pertanyaan (X1)

Mengenai kesesuaian penggunaan warna dan desain latar belakang tampilan pada aplikasi ini, berdasarkan tabel 3.6 di bawah, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 17 orang yang menjawab cukup baik dengan persentase 56.7%, 11 orang menjawab baik dengan persentase 36.6%, 2 orang menjawab

sangat baik dengan persentase 6.7%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 75 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan kesesuaian penggunaan warna dan desain latar belakang tampilan aplikasi ini baik.

Tabel 3.6. Frekuensi Hasil Jawaban (X1)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	17	11	2	30
Persentase	0%	56.7%	36.6%	6.7%	100%
Nilai	0	34	33	8	75

3.11.4.2 Pertanyaan (X2)

Tabel 3.7. Frekuensi Hasil Jawaban (X2)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	8	18	4	30
Persentase	0%	26.7%	60%	13.3%	100%
Nilai	0	16	54	16	84

Mengenai ketepatan ukuran tulisan pada tampilan aplikasi ini, berdasarkan tabel 3.7 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 18 orang yang menjawab baik dengan persentase 60%, 8 orang menjawab cukup baik dengan persentase 26.7%, 4 orang menjawab sangat baik dengan persentase 13.3%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 84 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan ketepatan ukuran tulisan pada tampilan aplikasi ini baik.

3.11.4.3 Pertanyaan (X3)

Mengenai permainan pada aplikasi mudah dimengerti, berdasarkan tabel 3.8 di bawah, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 13 orang yang menjawab sangat baik dengan persentase 43.3%, 12 orang menjawab baik dengan persentase

40%, 5 orang menjawab sangat baik dengan persentase 16.7%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 98 yang termasuk dalam interval 97.5 – 120 yaitu sangat baik, maka responden menyatakan permainan pada aplikasi mudah dimengerti sangat baik.

Tabel 3.8. Frekuensi Hasil Jawaban (X3)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	5	12	13	30
Persentase	0%	16.7%	40%	43.3%	100%
Nilai	0	10	36	52	98

3.11.4.4 Pertanyaan (X4)

Tabel 3.9. Frekuensi Hasil Jawaban (X4)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	9	16	5	30
Persentase	0%	30%	53.3%	16.7%	100%
Nilai	0	18	48	20	86

Mengenai ketepatan fungsi tombol dan tujuan yang diinginkan pada tampilan aplikasi ini, berdasarkan tabel 3.9 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 16 orang yang menjawab baik dengan persentase 53.3%, 9 orang menjawab cukup baik dengan persentase 30%, 5 orang menjawab sangat baik dengan persentase 16.7%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 86 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan fungsi tombol dan tujuan yang diinginkan pada tampilan aplikasi ini baik.

3.11.4.5 Pertanyaan (X5)

Mengenai kejelasan tampilan animasi pada aplikasi ini, berdasarkan tabel 3.10 di bawah, terlihat dari 30 responden yang

ditanya dengan pernyataan tersebut, ada 16 orang yang menjawab baik dengan persentase 53.3%, 12 orang menjawab cukup baik dengan persentase 40%, 2 orang menjawab sangat baik dengan persentase 6.7%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 80 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan kejelasan tampilan animasi pada aplikasi ini baik.

Tabel 3.10. Frekuensi Hasil Jawaban (X5)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	12	16	2	30
Persentase	0%	40%	53.3%	6.7%	100%
Nilai	0	24	48	8	80

3.11.4.6 Pertanyaan (X6)

Tabel 3.11. Frekuensi Hasil Jawaban (X6)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	1	9	17	3	30
Persentase	3.3%	30%	56.7%	10%	100%
Nilai	1	18	51	12	82

Mengenai kesesuaian ilustrasi musik dalam mendukung aplikasi ini, berdasarkan tabel 3.11 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 17 orang yang menjawab baik dengan persentase 56.7%, 9 orang menjawab cukup baik dengan persentase 30%, 3 orang menjawab sangat baik dengan persentase 10%, dan 1 orang menjawab tidak baik dengan persentase 3.3%. Dengan jumlah nilai 82 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan kesesuaian ilustrasi music dalam mendukung aplikasi ini baik.

3.11.4.7 Pertanyaan (X7)

Tabel 3.12. Frekuensi Hasil Jawaban (X7)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	7	20	3	30
Persentase	0%	23.3%	66.7%	10%	100%
Nilai	0	14	60	12	86

Mengenai ketepatan penyampaian materi video animasi terkait proses penenggelaman kapal pada aplikasi ini, berdasarkan tabel 3.12 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 20 orang yang menjawab baik dengan persentase 66.7%, 7 orang menjawab cukup baik dengan persentase 23.3%, 3 orang menjawab sangat baik dengan persentase 10%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 86 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan ketepatan penyampaian materi video animasi terkait proses penenggelaman kapal pada aplikasi ini baik.

3.11.4.8 Pertanyaan (X8)

Tabel 3.13. Frekuensi Hasil Jawaban (X8)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	8	7	15	30
Persentase	0%	26.7%	23.3%	50%	100%
Nilai	0	16	21	60	97

Mengenai kemudahan pengoperasian aplikasi pada aplikasi ini, berdasarkan tabel 3.13 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 15 orang yang menjawab sangat baik dengan persentase 50%, 8 orang menjawab cukup baik dengan persentase 26.7%, 7 orang menjawab baik dengan persentase 23.3%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 97 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka

responden menyatakan kemudahan pengoperasian aplikasi pada aplikasi ini baik.

3.11.4.9 Pertanyaan (X9)

Tabel 3.14. Frekuensi Hasil Jawaban (X9)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	0	5	14	11	30
Persentase	0%	16.7%	46.6%	36.7%	100%
Nilai	0	10	42	44	96

Mengenai responden dapat mengerti mengenai kampanye "Stop Illegal Fishing" pada aplikasi ini, berdasarkan tabel 3.14 di atas, terlihat dari 30 responden yang ditanya dengan pernyataan tersebut, ada 14 orang yang menjawab baik dengan persentase 46.6%, 11 orang menjawab sangat baik dengan persentase 36.7%, 5 orang menjawab cukup baik dengan persentase 16.7%, dan 0 orang menjawab tidak baik dengan persentase 0%. Dengan jumlah nilai 96 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan responden dapat mengerti mengenai kampanye "Stop Illegal Fishing" pada aplikasi ini baik.

3.11.4.10 Pertanyaan (X10)

Tabel 3.15. Frekuensi Hasil Jawaban (X10)

Penilaian	TB	CB	B	SB	Jumlah
Bobot Nilai	1	2	3	4	
Frekuensi	1	8	10	11	30
Persentase	3.3%	26.7%	33.3%	36.7%	100%
Nilai	1	16	30	44	91

Mengenai responden tertarik untuk mempelajari/aktif dalam kampanye "Stop Illegal Fishing", berdasarkan tabel 3.15 di atas, terlihat

dari 30 responden yang ditanya dengan pernyataan tersebut, ada 11 orang yang menjawab sangat baik dengan persentase 36.7%, 10 orang menjawab sangat baik dengan persentase 33.3%, 8 orang menjawab cukup baik dengan persentase 26.7%, dan 1 orang menjawab tidak baik dengan persentase 3.3%. Dengan jumlah nilai 91 yang termasuk dalam interval 75 – 97.4 yaitu baik, maka responden menyatakan responden tertarik untuk mempelajari/aktif dalam kampanye "Stop Illegal Fishing".

3.11.5 Rata-rata dan Persentase dari Pertanyaan

Setelah penulis menghitung nilai yang di dapat dari masing -masing pertanyaan, penulis melakukan perhitungan rata – rata dan persentase yang akan digunakan untuk menyimpulkan rata rata dan persentase hasil jawaban dari responden. Hasil tersebut dapat dilihat pada tabel 3.16.

Tabel 3.16 Rata-rata dan Persentase Responden

No.	Pertanyaan	Rata-rata	Persentase
1	Bagaimana kesesuaian penggunaan warna dan desain latar belakang aplikasi ini?	0.63	63%
2	Bagaimana ketepatan ukuran tulisan pada aplikasi ini?	0.7	70%
3	Apakah permainan mudah dimengerti?	0.82	82%
4	Bagaimana ketepatan fungsi tombol dengan tujuan yang diinginkan?	0.72	72%
5	Apakah tampilan animasi jelas?	0.67	67%

Tabel 3.16 Rata-rata dan Persentase Responden

No.	Pertanyaan	Rata-rata	Persentase
6	Bagaimana kesesuaian ilustrasi musik dalam mendukung aplikasi?	0.68	68%
7	Bagaimana ketepatan penyampaian materi video animasi terkait proses penenggelaman kapal?	0.72	72%
8	Bagaimana kemudahan dalam pengoperasian aplikasi?	0.81	81%
9	Apakah anda dapat mengerti mengenai kampanye "Stop Illegal Fishing"?	0.8	80%
10	Apakah anda tertarik untuk mempelajari/aktif dalam kampanye "stop illegal fishing"?	0.76	76%

Dari hasil rata – rata seluruh pertanyaan dapat diperoleh data jumlah nilai, rata - rata keseluru dan presentase keseluruhan sebagai berikut:

1. Jumlah nilai total dari pertanyaan adalah 875 yang diperoleh dari rumus:

$$\Sigma nilai$$

2. Jumlah rata – rata total dari pertanyaan adalah 0.73 yang diperoleh dari rumus:

$$\frac{\Sigma \text{nilai}}{120 \times 10}$$

3. Jumlah persentase total dari pertanyaan adalah 73% yang diperoleh dari rumus:

$$\frac{\Sigma \text{nilai}}{120 \times 10} x100\%$$

Dari hasil perhitungan kuesioner, presentasi total dari pertanyaan diperoleh sebesar 73%. Dengan 30 responden dari kalangan mahasiswa