# BAB 4

# IMPLEMENTASI DAN PENGUJIAN SISTEM

## 4.1 Implementasi Sistem

Tahap implementasi merupakan tahap pembangunan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi, dimana aplikasi siap untuk dioperasikan. Aplikasi ini diimplementasikan dengan nama *The Survival Setup* pada *file .exe* (*executable file*).

### 4.1.1 Implementasi Perangkat Keras

Perangkat keras yang dapat digunakan untuk menjalankan aplikasi *game* ini tidak harus memiliki spesifikasi yang tinggi. Perangkat keras minimum agar dapat menjalankan *game The Survival* adalah sebagai berikut:

1. *Processor Dual Core* 1,3 Ghz
2. *RAM* 1024 MB
3. Slot *Harddisk* 100 MB
4. Kartu grafik dengan DirectX 9 *compatible sound card*
5. LCD monitor dengan resolusi 1366 x 768 pixel

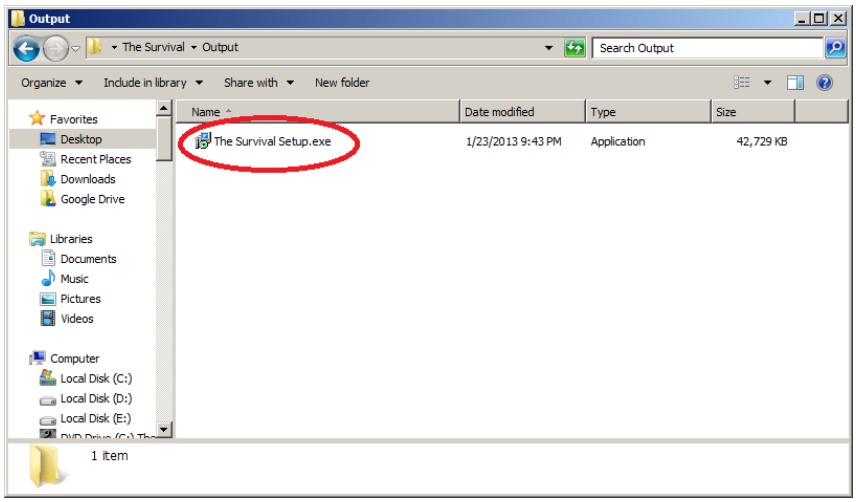
### 4.1.2 Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *The Survival* adalah sistem operasi *Windows XP* 32 bit ke atas dengan *graphic accelerator*.

### 4.1.3 Implementasi Instalasi Aplikasi

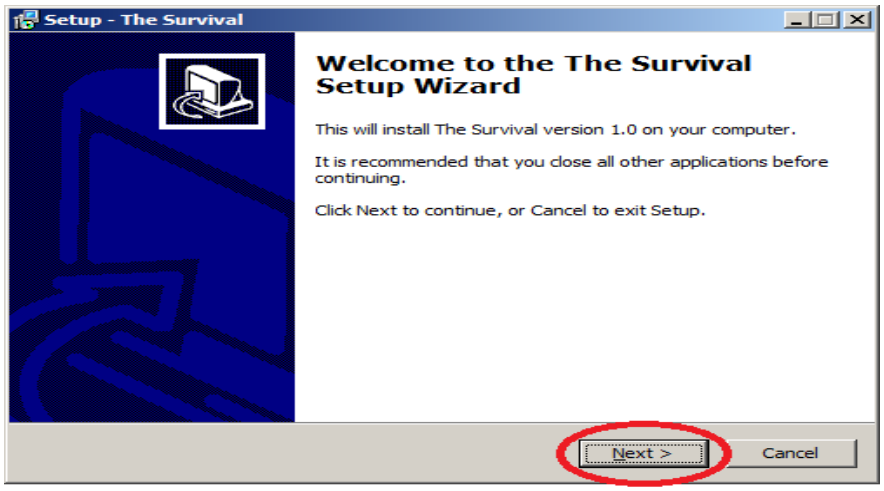
Sebelum dapat memainkan *game The Survival*, aplikasi harus terinstal pada *device* computer. Petunjuk dalam instalasi aplikasi *game* ini adalah sebagai berikut:

1. Klik dua kali *file The Survival Setup.exe* yang ada pada *harddisk* atau *compact disc*



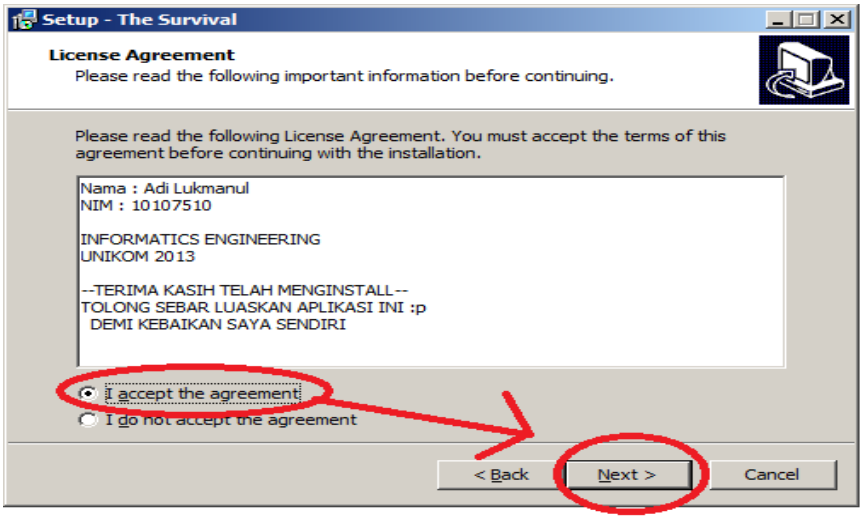

Gambar 4.1 *The Survival File Setup*

2. Setelah muncul dialog *The Survival Setup Wizard*, kemudian klik *Next*



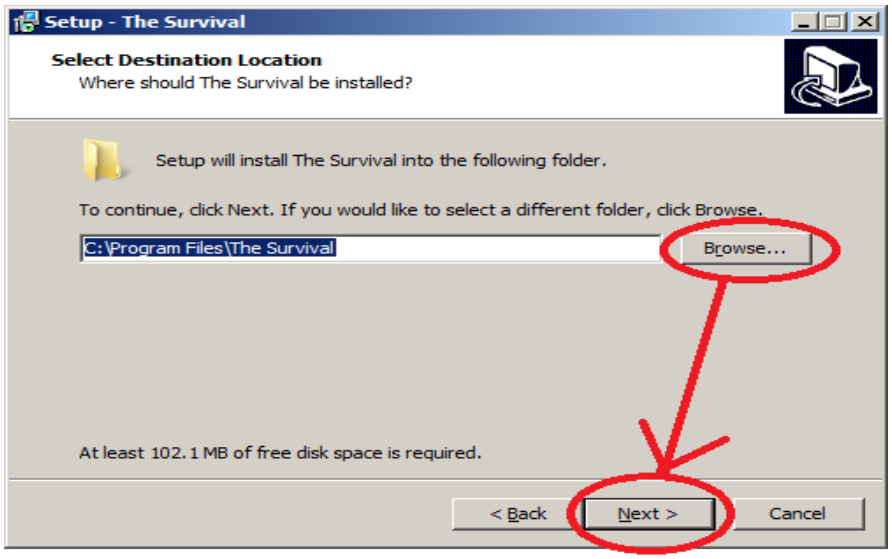

Gambar 4.2 *The Survival Setup Wizard*

3. Pada dialog *License Agreement* pilih opsi *"I accept the agreement"* kemudian klik *next*



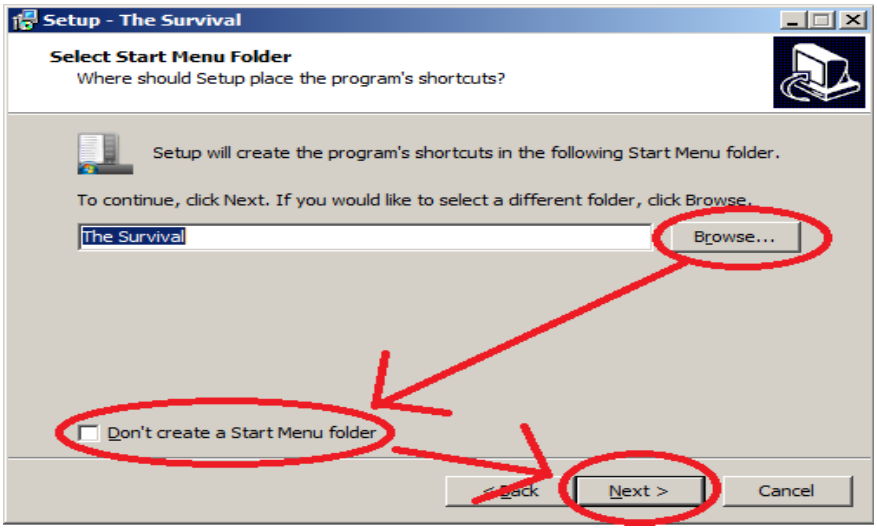

Gambar 4.3 *The Survival License Agreement*

4. Pada dialog *destination* pilih *folder* tempat aplikasi akan disimpan dengan klik *browse*, kemudian pilih tombol *next*



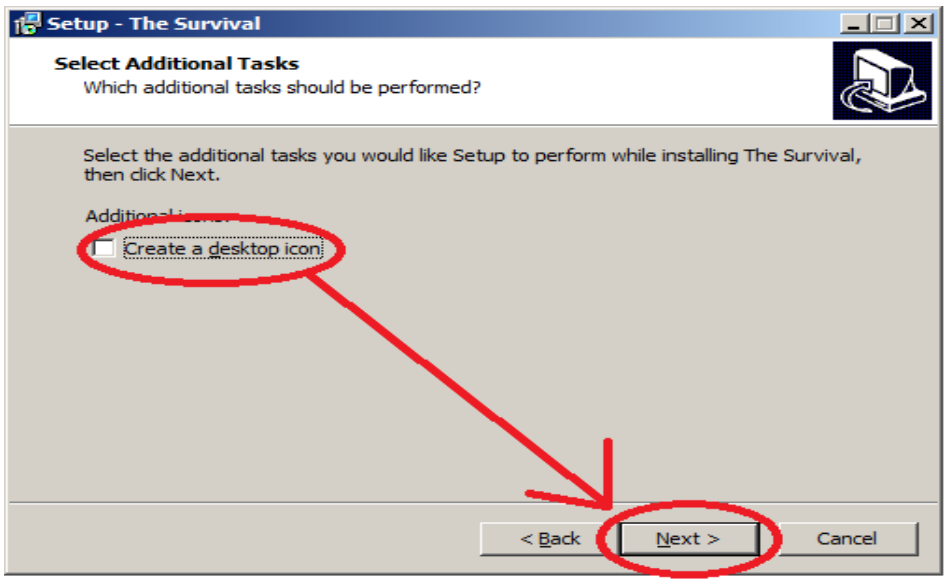

Gambar 4.4 *The Survival Destination Folder*

5. Pada dialog *program's shortcuts* pilih *shortcuts* aplikasi akan disimpan dengan klik *browse*, apabila tidak ingin menambahkan *shotcuts* pilih opsi *"Don't create a Start Menu folder"*, kemudian klik *next*



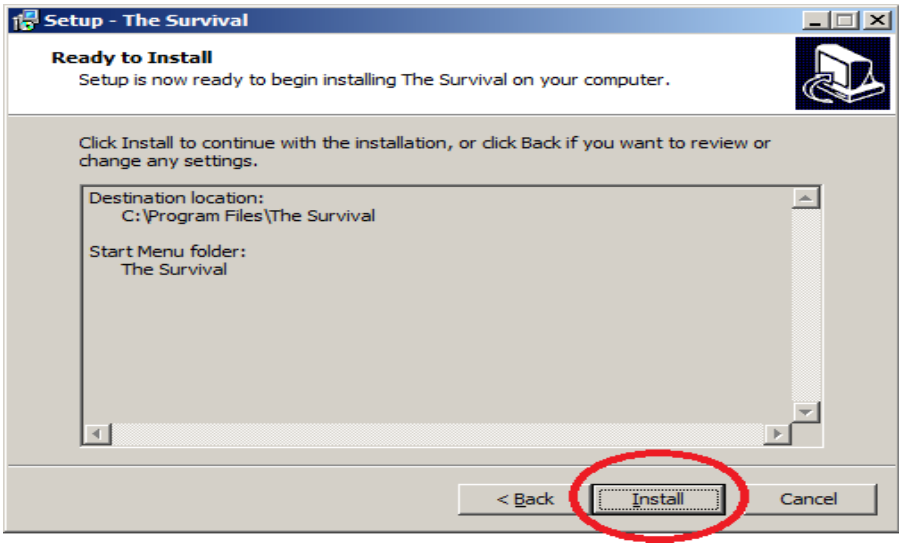

Gambar 4.5 *The Survival Destination Shortcuts*

6. Pada dialog *additional tasks*, pilih opsi "*Create a desktop icon*" untuk membuat ikon pada *desktop* kemudian klik *next*



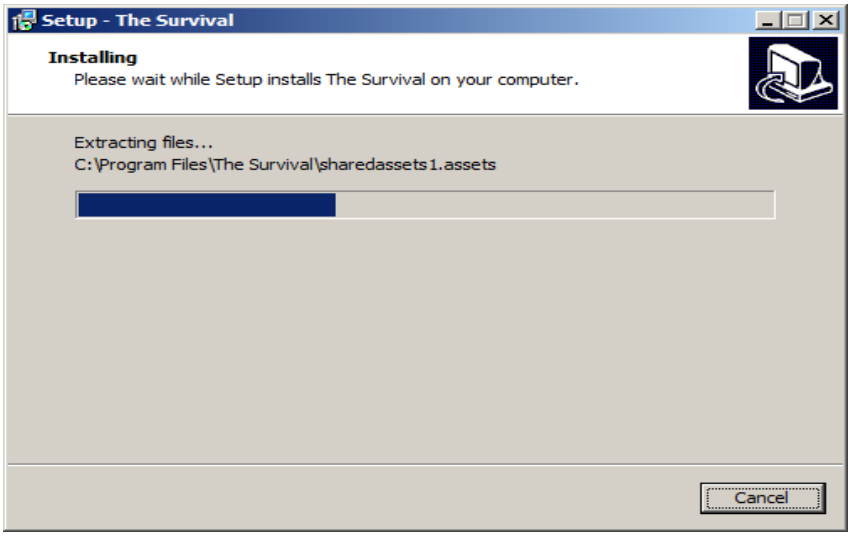

Gambar 4.6 *The Survival Additional Tasks*

7. Pada dialog *installation*, klik *install* untuk melakukan instalasi aplikasi



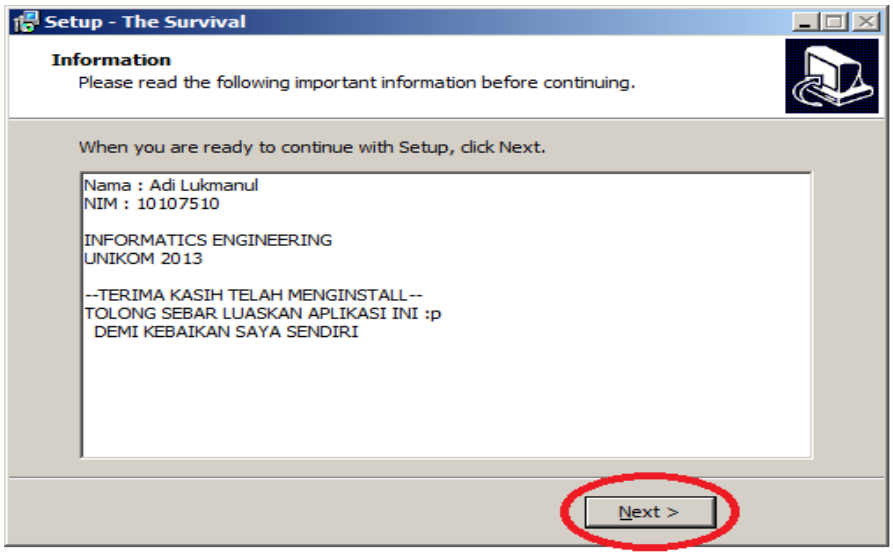

Gambar 4.7 *Begin Installing The Survival*

8. Pada dialog *Installing*, tunggu sampai *loading* selesai dan aplikasi terinstal, apabila ingin membatalkan instalasi klik tombol *cancel*




Gambar 4.8 *The Survival Load Installation*

9. Pada dialog *information*, baca informasi mengenai aplikasi, kemudian klik tombol *next* untuk melanjutkan




Gambar 4.9 *The Survival Important Information*

10. Pada dialog *completing installation*, pilih opsi *"Launch The Survival"* untuk memulai aplikasi setelah instalasi selesai, kemudian klik tombol *finish*




Gambar 4.10 *Completing Installation*

### 4.1.4 Implementasi Antarmuka

Berikut ini merupakan antarmuka yang ada pada aplikasi *game The Survival*:

1. Antarmuka Menu Utama

   Saat pengguna membuka aplikasi pertama kali, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka menu utama. Berikut ini merupakan tampilan dari antarmuka menu utama yang diperlihatkan pada Gambar 4.11.




Gambar 4.11 Tampilan Antarmuka Menu Utama

Terdapat 4 (empat) sub menu di dalam menu utama pada *game the survival*, yaitu menu mulai yang berfungsi untuk memulai permainan, menu skor tertinggi yang berfungsi untuk melihat pencapaian skor tertinggi, menu cara bermain yang berfungsi untuk melihat petunjuk bermain, dan menu keluar untuk mengeluarkan aplikasi.

2. Antarmuka Menu Skor Tertinggi

   Saat pemain memilih menu skor tertinggi, maka antarmuka yang ditampilkan oleh sistem adalah antarmuka skor tertinggi dengan informasi pencapaian skor tertinggi oleh pemain. Berikut ini merupakan tampilan dari antarmuka menu skor tertinggi yang diperlihatkan pada Gambar 4.12.




Gambar 4.12 Tampilan Antarmuka Menu Skor Tertinggi

3. Antarmuka Menu Cara Bermain

Saat pemain memilih menu cara bermain, maka antarmuka yang ditampilkan oleh sistem adalah informasi mengenai petunjuk-petunjuk cara bermain dalam *game the survival*. Berikut ini merupakan tampilan dari antarmuka menu cara bermain yang diperlihatkan pada Gambar 4.13.




Gambar 4.13 Tampilan Antarmuka Menu Cara Bermain

## 4.2 Pengujian Sistem

Pada umunya ada 2 (dua) macam teknik pengujian yaitu *white box testing* dan *black box testing*. Pengujian yang menggunakan teknik *white box testing* lebih menitikberatkan pada struktur internal (*source code*) program, di mana setiap baris kode yang akan di *compile* dicek satu persatu untuk mengetahui apakah masih terjadi *error*. Pengujian pada *black box testing* tidak terlalu memperhatikan struktur internal (*source code*) dari program, tetapi lebih kepada menemukan kesalahan program. Pada *the survival* digunakan pengujian *black box* dan *white box testing*. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian *alpha* (fungsional) dan pengujian *beta*.

### 4.2.1 Pengujian *Alpha*

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Pengujian *alpha* berfokus pada persyaratan fungsional perangkat lunak.

#### 4.2.1.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game the survival.* Skenario pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada table 4.1 di bawah ini.

Tabel 4.1 Skenario Pengujian Aplikasi

| No | Komponen yang di uji | Skenario Pengujian | Jenis Pengujian |
|---|---|---|---|
| 1. | Menu Utama | Memilih menu mulai | *Black box* |
| | | Memilih menu skor tertinggi | *Black box* |
| | | Memilih menu cara bermain | *Black box* |
| 2. | Menu Mulai | Memilih menu mulai | *Black box* |
| | | Dunia Permainan | *White box* |
| 3. | Menu Skor Tertinggi | Memilih Menu skor tertinggi | *Black box* |
| | | Memilih tombol kembali | *Black box* |
| 4. | Menu Cara Bermain | Memilih menu cara bermain | *Black box* |
| | | Memilih tombol kembali | *Black box* |
| 5. | Menggerakan karakter | Menekan tombol W | *Black box* |
| | | Menekan tombol A | *Black box* |
| | | Menekan tombol S | *Black box* |
| | | Menekan tombol D | *Black box* |
| 6. | Menembak | Menekan klik kiri | *Black box* |

#### 4.2.1.2 Kasus dan Hasil Pengujian (*Black Box Testing*)

Berikut ini merupakan kasus untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box testing* berdasarkan skenario pengujian aplikasi yang terdapat pada tabel 4.1 yang sudah dijabarkan. Hasil pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada tabel 4.2.

Tabel 4.2 Pengujian *Black Box*

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | Skenario Uji | Hasil yang Diharapkan | Kesimpulan |
| 1. | Menu Utama | Memilih Menu Mulai | Sistem menampilkan dunia permainan | [√] Berhasil<br>[ ] Tidak Berhasil |

Tabel 4.2 Pengujian *Black Box* (Lanjutan)

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | **Skenario Uji** | **Hasil yang Diharapkan** | **Kesimpulan** |
| | | Memilih Menu Skor Tertinggi | Sistem menampilkan halaman menu skor tertinggi | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih Menu Cara Bermain | Sistem menampilkan halaman menu cara bermain | [√] Berhasil<br>[ ] Tidak Berhasil |
| 2. | Menu Mulai | Memilih Menu Mulai | Sistem menampilkan dunia permainan | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Dunia Permainan | Dapat dilihat pada sub bab 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*) | |
| 3. | Menu Skor Tertinggi | Memilih Menu Skor Tertinggi | Sistem menampilkan halaman skor tertinggi | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih tombol kembali | Sistem menampilkan menu utama | [√] Berhasil<br>[ ] Tidak Berhasil |
| 4. | Menu Cara bermain | Memilih Menu Cara Bermain | Sistem menampilkan halaman cara bermain | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Memilih tombol kembali | Sistem menampilkan menu utama | [√] Berhasil<br>[ ] Tidak Berhasil |
| 5. | Menggerakkan Karakter | Menekan tombol W | Sistem menggerakkan karakter ke atas layar | [√] Berhasil<br>[ ] Tidak Berhasil |

Tabel 4.2 Pengujian *Black Box* (Lanjutan)

| No | Kasus yang akan di uji | Skenario dan Hasil Uji | | |
|---|---|---|---|---|
| | | **Skenario Uji** | **Hasil yang Diharapkan** | **Kesimpulan** |
| 5. | Menggerakkan karakter | Menekan tombol A | Sistem menggerakkan karakter ke kiri layar | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Menekan tombol S | Sistem menggerakkan karakter ke bawah layar | [√] Berhasil<br>[ ] Tidak Berhasil |
| | | Menekan tombol D | Sistem menggerakkan karakter ke kanan layar | [√] Berhasil<br>[ ] Tidak Berhasil |
| 6. | Menembak | Menekan klik kiri | Sistem mengeluarkan peluru dari karakter | [√] Berhasil<br>[ ] Tidak Berhasil |

### 4.2.1.3 Kasus dan Hasil Pengujian (*White box Testing*)

Berikut ini merupakan kasus menu mulai untuk menguji permainan di dalam perangkat lunak yang sudah dibangun dengan menggunakan metode *white box testing* berdasarkan skenario pengujian yang terdapat pada tabel 4.1. pengujian yang dilakukan adalah sebagai berikut:

a. Algoritma Yang Akan di Uji

Berikut ini merupakan algoritma dari metode *procedural content generation* yang penerapannya menggunakan bahasa pemrograman C#:

```
1.  if (numberOfUnits < totalUnits && totalSpawnedUnits >=
    totalUnits){
2.  spawn = false;}else
```

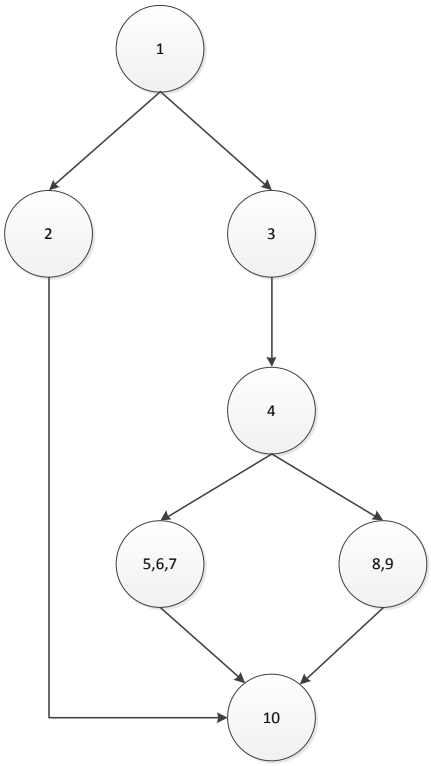
```
3.  {yield return new WaitForSeconds(timeBetweenSpawns);
4.  if (unitList[(int)unitLevel] != null){
5.  Transform unit = InstanceManager.Spawn(unitList[(int)
    unitLevel].transform,spawnLocation.position,Quaternion.ide
    ntity);
6.  numberOfUnits++;
7.  totalSpawnedUnits++;}else
8.  {Debug.LogError("Error trying to spawn unit of level " +
    unitLevel.ToString() + " on spawner " + spawnID + " - No
    unit set");
9.  spawn = false;}}
10. break;
```

Berikut ini merupakan algoritma dari algoritma A* yang penerapannya menggunakan bahasa pemrograman C#:

```
1.  if (connections == null) {
2.  return;}
3.  else {nodeR.g = nodeR.parent.g+nodeR.cost+penalty +
    nodeRunData.path.GetTagPenalty(tags);}
4.  if (heuristic == Heuristic.None) {
5.  nodeR.h = 0;}
6.  else if (heuristic == Heuristic.Euclidean) {
7.  nodeR.h = (uint)Mathfx.RoundToInt ((position-target
    Position).magnitude*scale);}
8.  else if (heuristic == Heuristic.Manhattan) {
9.  nodeR.h = (uint)Mathfx.RoundToInt  ((Abs (position.x-
    targetPosition.x) + Abs (position.y-targetPosition.y) +
    Abs (position.z-targetPosition.z))* scale);}
10. else if (heuristic == Heuristic.DiagonalManhattan) {
11. int xDistance = Abs (position.x-targetPosition.x);
12. int zDistance = Abs (position.z-targetPosition.z);
13. if (xDistance > zDistance) {
14. nodeR.h = (uint)(14*zDistance + 10*(xDistance-
    zDistance))/10;}
15. else {nodeR.h = (uint)(14*xDistance + 10*(zDistance-
    xDistance))/10;}
16. nodeR.h = (uint)Mathfx.RoundToInt (nodeR.h * scale);}
17. get {return nodeR.g+nodeR.h;}
```

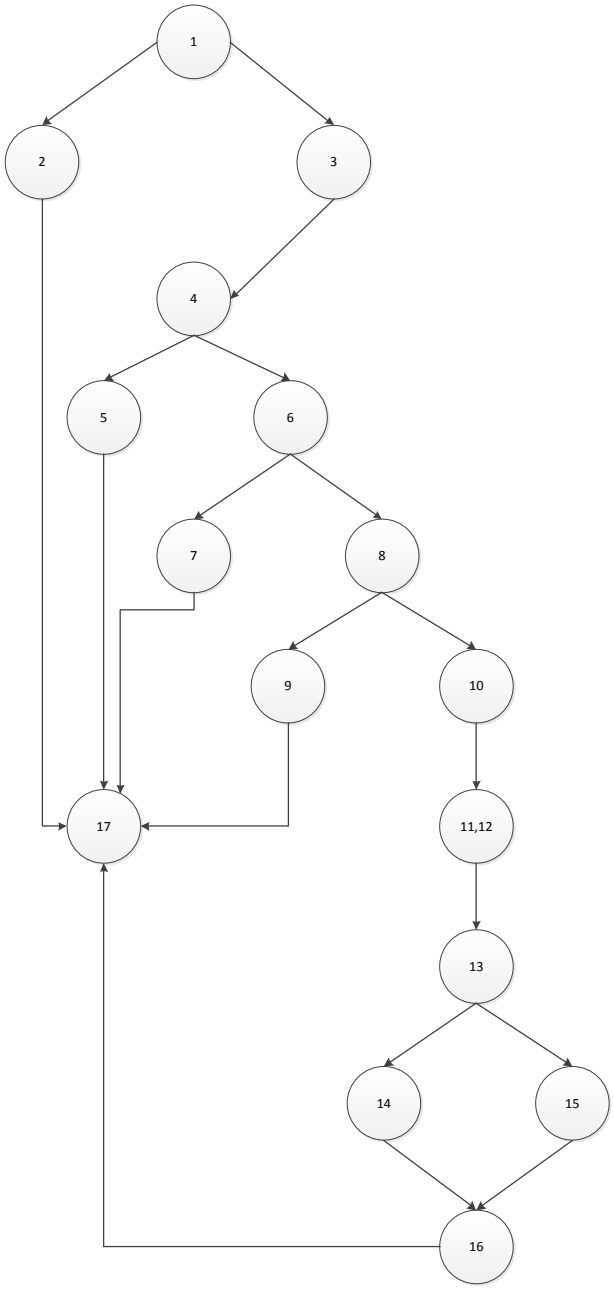
b. *Flow Graph* Permainan

*Flow graph* metode *procedural content generation* pada *game the survival* dapat dilihat pada Gambar 4.14 di bawah ini.




Gambar 4.14 *Flow Graph* Metode *Procedural Content Generation*

*Flow graph* algoritma A* pada *game the survival* dapat dilihat pada Gambar 4.15 di bawah ini.




Gambar 4.15 *Flow Graph* Algoritma A*

Keterangan :

= Menggambarkan Kondisi

= Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

*Cyclomatic complexity* untuk *flow graph* metode *procedural content generation* adalah sebagai berikut:

E = 8, N = 7

V(G) = E – N + 2P

V(G) = 8 – 7 +2(1)

V(G) = 1 + 2 = 3

*Cyclomatic complexity* untuk *flow graph* algoritma A* adalah sebagai berikut:

E = 20, N = 16

V(G) = E – N + 2P

V(G) = 20 – 16 +2(1)

V(G) = 4 + 2 = 6

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

P = Jumlah bagian dari *graph* yang tidak terkoneksi

d. *Independent Path*

*Independent path* untuk metode *procedural content generation* adalah sebagai berikut:

*Path* 1 = (1) – (3) – (4) – (5,6,7) – (10)

*Path* 2 = (1) – (3) – (4) – (8,9) – (10)

*Path* 3 = (1) – (2) – (10)

*Independent path* untuk algoritma A* adalah sebagai berikut:

*Path* 1 = (1) – (2) – (17)

*Path* 2 = (1) – (3) – (4) – (5) – (17)

*Path* 3 = (1) – (3) – (4) – (6) – (7) – (17)

*Path* 4 = (1) – (3) – (4) – (6) – (8) – (9) – (17)

*Path* 5 = (1) – (3) – (4) – (6) – (8) – (10) – (11,12) – (13) – (15) – (16) – (17)

*Path* 6 = (1) – (3) – (4) – (6) – (8) – (10) – (11,12) – (13) – (14) – (16) – (17)

e. *Graph Matriks*

*Graph matriks* untuk metode *procedural content generation* dapat dilihat pada Tabel 4.3 di bawah ini.

Tabel 4.3 *Graph Matriks* Metode *Procedural Content Generation*

| | (1) | (2) | (3) | (4) | (5,6,7) | (8,9) | (10) | E |
|---|---|---|---|---|---|---|---|---|
| (1) | | 1 | 1 | | | | | 1 |
| (2) | | | | | | | 1 | 0 |
| (3) | | | | 1 | | | | 0 |
| (4) | | | | | 1 | 1 | | 1 |
| (5,6,7) | | | | | | | 1 | 0 |
| (8,9) | | | | | | | 1 | 0 |
| (10) | | | | | | | | 0 |
| SUM ( E )+1 | | | | | | | | 3 |

*Graph matriks* untuk algoritma A* dapat dilihat pada Tabel 4.4 di bawah ini.

Tabel 4.4 *Graph Matriks* Algoritma A*

| | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) | (11,12 | (13) | (14) | (15) | (16) | (17) | E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1) | | 1 | 1 | | | | | | | | | | | | | | 1 |
| (2) | | | | | | | | | | | | | | | | 1 | 0 |
| (3) | | | | 1 | | | | | | | | | | | | | 0 |
| (4) | | | | | 1 | 1 | | | | | | | | | | | 1 |
| (5) | | | | | | | | | | | | | | | | 1 | 0 |
| (6) | | | | | | | 1 | 1 | | | | | | | | | 1 |
| (7) | | | | | | | | | | | | | | | | 1 | 0 |
| (8) | | | | | | | | | 1 | 1 | | | | | | | 1 |
| (9) | | | | | | | | | | | | | | | | 1 | 0 |
| (10) | | | | | | | | | | | 1 | | | | | | 0 |
| (11,12) | | | | | | | | | | | | 1 | | | | | 0 |
| (13) | | | | | | | | | | | | | 1 | 1 | | | 1 |
| (14) | | | | | | | | | | | | | | | 1 | | 0 |
| (15) | | | | | | | | | | | | | | | 1 | | 0 |
| (16) | | | | | | | | | | | | | | | | 1 | 0 |
| (17) | | | | | | | | | | | | | | | | | 0 |
| SUM ( E )+1 | | | | | | | | | | | | | | | | | 6 |

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Hasil Uji

Tabel 4.5 Hasil Uji Metode *Procedural Conten Generation*

| **Kasus dan Hasil Uji** | | | | |
|---|---|---|---|---|
| **No. *path*** | **Kasus Uji** | **Hasil yang Diharapkan** | **Hasil Sesuai Kasus Uji** | **Keterangan** |
| 1. | Objek yang dimunculkan belum diinisialisasi | Menghentikan pemunculan musuh | Menghentikan pemunculan musuh | [√] Dikerjakan<br>[ ] Tidak dikerjakan |
| 2. | Objek yang dimunculkan diinisialisasi, total objek dimunculkan ≤ total unit, jumlah objek yang dimunculkan >total unit | Musuh dimunculkan | Musuh dimunculkan | [√] Dikerjakan<br>[ ] Tidak dikerjakan |

Tabel 4.5 Hasil Uji Metode *Procedural Conten Generation* (Lanjutan)

| Kasus dan Hasil Uji | | | | |
|---|---|---|---|---|
| **No. *path*** | **Kasus Uji** | **Hasil yang Diharapkan** | **Hasil Sesuai Kasus Uji** | **Keterangan** |
| 3. | Objek yang dimunculkan diinisialisasi, total objek yang dimunculkan ≥ total unit, jumlah objek yang dimunculkan <total unit | Menghentikan pemunculan musuh sementara | Menghentikan pemunculan musuh sementara | [√] Dikerjakan<br>[ ] Tidak dikerjakan |

Tabel 4.6 Hasil Uji Algoritma A*

| Kasus dan Hasil Uji | | | | |
|---|---|---|---|---|
| **No. *path*** | **Kasus Uji** | **Hasil yang Diharapkan** | **Hasil Sesuai Kasus Uji** | **Keterangan** |
| 1. | `connections == null` (Posisi simpul tetangga berada diluar *path*) | Tidak dilakukan pencarian F | Tidak dilakukan pencarian F | [√] Dikerjakan<br>[ ] Tidak dikerjakan |
| 2. | `connections == null` dan `heuristic == Heuristic.None` (Posisi simpul tetangga tepat di tujuan) | Mencari F dengan nilai heuristik = 0 | Mencari F dengan nilai heuristik = 0 | [√] Dikerjakan<br>[ ] Tidak dikerjakan |
| 3. | `connections ≠ null` dan `heuristic == Heuristic.Euclidean` (Posisi simpul tetangga berada dalam | Mencari F dengan nilai H menggunakan *eucledian* | Mencari F dengan nilai H menggunakan *eucledian* | [√] Dikerjakan<br>[ ] Tidak dikerjakan |

| | | | | |
|---|---|---|---|---|
| | *path*, tetapi tidak menyentuh garis *path*) | | | |
| 4. | `connections ≠ null` dan `heuristic == Heuristic.Manhattan` (Posisi *node* tetangga tegak lurus terhadap simpul terpilih) | Mencari F dengan nilai H menggunakan *manhattan distance* | Mencari F dengan nilai H menggunakan *manhattan distance* | [√] Dikerjakan<br>[ ] Tidak dikerjakan |
| 5. | `connections ≠ null`, `heuristic == Heuristic.DiagonalManhattan` dan `xDistance > zDistance` (Posisi *node* tetangga *diagonal* terhadap simpul terpilih, dengan nilai x > z) | Mencari F dengan nilai H menggunakan *diagonal manhattan* | Mencari F dengan nilai H menggunakan *diagonal manhattan* | [√] Dikerjakan<br>[ ] Tidak dikerjakan |
| 6. | `connections ≠ null`, `heuristic == Heuristic.DiagonalManhattan` dan `xDistance <= zDistance` (Posisi *node* tetangga *diagonal* terhadap simpul terpilih, dengan nilai x ≤z) | Mencari F dengan nilai H menggunakan *diagonal manhattan* | Mencari F dengan nilai H menggunakan *diagonal manhattan* | [√] Dikerjakan<br>[ ] Tidak dikerjakan |

g. Kesimpulan

Dari hasil pengujian pada metode *procedural content generation* dan algoritma A*, maka didapat kesimpulan bahwa seluruh *node* yang terdapat pada setiap *independent path* telah dikerjakan serta tidak terjadi pengulangan tak berhingga.

#### 4.2.1.4 Kesimpulan Pengujian *Alpha*

Dari hasil pengujian yang dilakukan dapat disimpulkan bahwa sistem sudah dapat berjalan sesuai dengan yang diharapkan. Secara fungsionalitas, sistem sudah dapat menghasilkan keluaran yang diharapkan.

### 4.2.2 Pengujian *Beta*

Pengujian *beta* merupakan pengujian yang dilakukan secara objektif, dimana dilakukan pengujian secara langsung terhadap pengguna dengan menggunakan kuesioner mengenai kepuasan pengguna atas aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

### 4.2.3 Kuesioner

Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi. Kuesioner disebarkan kepada 30 orang dari umur 14 tahun ke atas yang sering bermain *game*, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian

penerapan sistem yang telah dibangun. Kuesioner terdiri dari 7 pernyataan dengan menggunakan skala 1 sampai 5 (skala *likert*).

Untuk mencari nilai presentase dari masing-masing jawaban kuesioner digunakan rumus skala *likert* sebagai berikut:

$$P = \frac{S}{Skorideal} \times 100 \text{ ……(4.1)}$$

Keterangan:

P = Nilai Presentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai yang ditetapkan jawaban

Skorideal = Nilai tertinggi dikalikan dengan jumlah sampel

Berikut ini adalah hasil presentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus di atas.

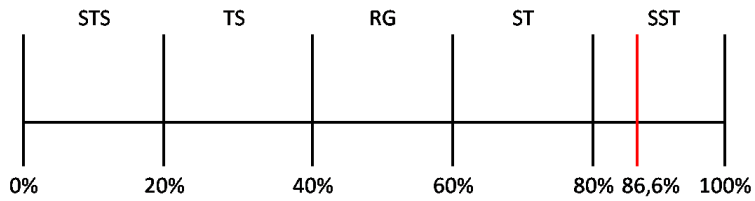
1. *Gameplay* pada *game the survival* menghibur pemain

Tabel 4.7 Tabel Perhitungan Presentase Pertanyaan Ke-1

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 1 | Sangat Setuju | 5 | 10 | 50 |
| | Setuju | 4 | 20 | 80 |
| | Ragu-ragu | 3 | - | - |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 130 |

$$P = \frac{130}{150} \times 100 = 86{,}67\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap menghibur atau tidaknya *game the survival* adalah 86,67% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju *game the survival* menghibur. Berikut kriteria interpretasi skor :




Gambar 4.16 Interpretasi Skor Pertanyaan Ke-1

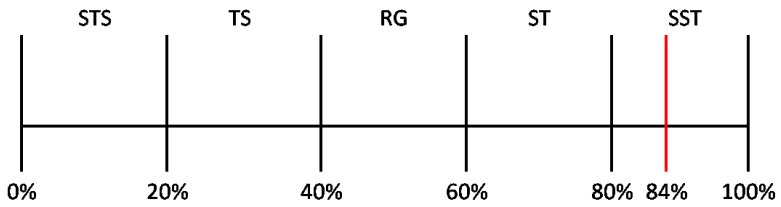
2. Tampilan grafik 2.5D pada *game the survival* sudah bagus

Tabel 4.8 Tabel Perhitungan Presentase Pertanyaan Ke-2

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 2 | Sangat Setuju | 5 | 11 | 55 |
| | Setuju | 4 | 14 | 56 |
| | Ragu-ragu | 3 | 5 | 15 |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 126 |

$$P = \frac{126}{150} \times 100 = 84\%$$

Berdasarkan hasil presentase nilai di atas, dapat disimpulkan bahwa penilaian terhadap sudah bagus atau tidaknya tampilan grafik 2.5D pada *game the survival* adalah 84% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju grafiknya sudah bagus. Berikut kriteria interpretasi skor :




Gambar 4.17 Interpretasi Skor Pertanyaan Ke-2

3. *Interface* (antarmuka) dalam *game the survival* nyaman untuk digunakan

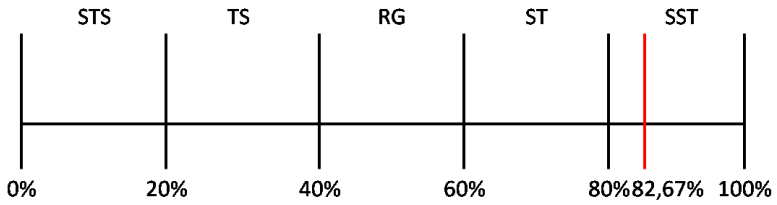
Tabel 4.9 Tabel Perhitungan Presentase Pertanyaan Ke-3

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 3 | Sangat Setuju | 5 | 7 | 35 |
| | Setuju | 4 | 20 | 80 |
| | Ragu-ragu | 3 | 3 | 9 |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 124 |

$$P = \frac{124}{150} \times 100 = 82{,}67\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap kenyamanan *interface* pada *game the survival* adalah

82,67% dari 100%, atau dapat dikategorikan bahwa pemain setuju. Berikut kriteria interpretasi skor :




Gambar 4.18 Interpretasi Skor Pertanyaan Ke-3

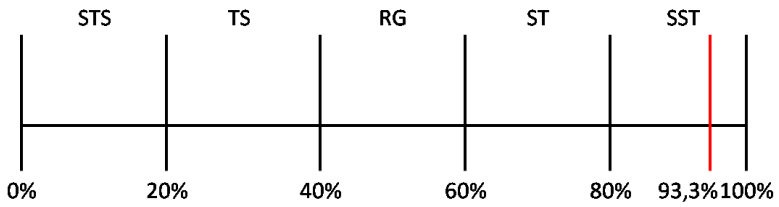
4. Mudah untuk mengontrol karakter pemain dengan menggunakan *keyboard* dan *mouse*

Tabel 4.10 Tabel Perhitungan Presentase Pertanyaan Ke-4

| **Pertanyaan** | **Keterangan** | **Skor** | **Frekuensi** | **S** |
|---|---|---|---|---|
| 4 | Sangat Setuju | 5 | 20 | 100 |
| | Setuju | 4 | 10 | 40 |
| | Ragu-ragu | 3 | - | - |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | | |
| Jumlah | | | 30 | 140 |

$$P = \frac{140}{150} \times 100 = 93{,}3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap kemudahan mengontrol karakter *game the survival* dengan menggunakan *keyboard* dan *mouse* adalah 93,3% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju. Berikut kriteria interpretasi skor :




Gambar 4.19 Interpretasi Skor Pertanyaan Ke-4

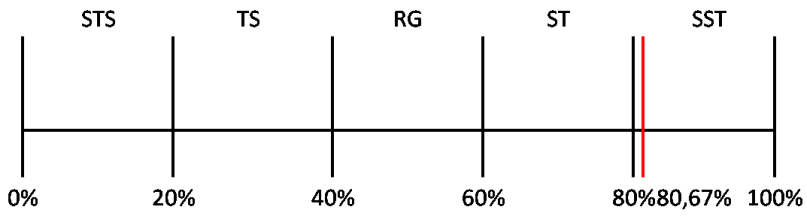
5. Mudah untuk menginstalasi *game The Survival*

Tabel 4.11 Tabel Perhitungan Presentase Pertanyaan Ke-5

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 7 | Sangat Setuju | 5 | 5 | 25 |
| | Setuju | 4 | 21 | 84 |
| | Ragu-ragu | 3 | 4 | 12 |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 121 |

$$P = \frac{121}{150} \times 100 = 80{,}67\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap mudah atau tidaknya untuk menginstalasi *game The Survival* adalah 80,67% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju bahwa proses instalasinya mudah. Berikut kriteria interpretasi skor :




Gambar 4.20 Interpretasi Skor Pertanyaan Ke-5

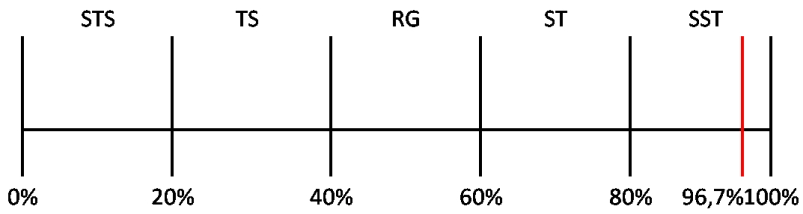
6. *Game The Survival* menarik sehingga perlu untuk menambah *level* dengan tingkat kesulitan yang lebih variatif

Tabel 4.12 Tabel Perhitungan Presentase Pertanyaan Ke-6

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 5 | Sangat Setuju | 5 | 25 | 125 |
| | Setuju | 4 | 5 | 20 |
| | Ragu-ragu | 3 | - | - |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 145 |

$$P = \frac{145}{150} \times 100 = 96{,}7\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap penambahan *level* pada *game the survival* adalah 96,7% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju. Berikut kriteria interpretasi skor :




Gambar 4.21 Interpretasi Skor Pertanyaan Ke-6

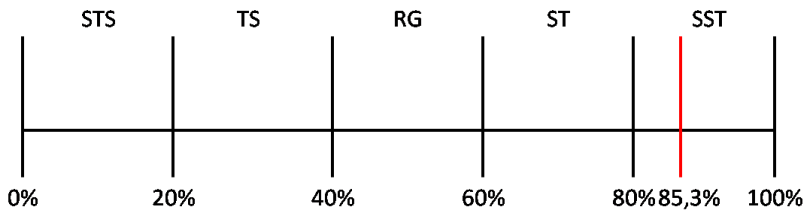
7. Supa tidak bosan perlu untuk menambahkan karakter pemain pada *game The Survival*

Tabel 4.13 Tabel Perhitungan Presentase Pertanyaan Ke-7

| Pertanyaan | Keterangan | Skor | Frekuensi | S |
|---|---|---|---|---|
| 6 | Sangat Setuju | 5 | 11 | 55 |
| | Setuju | 4 | 16 | 64 |
| | Ragu-ragu | 3 | 3 | 9 |
| | Tidak Setuju | 2 | - | - |
| | Sangat Tidak Setuju | 1 | - | - |
| Jumlah | | | 30 | 128 |

$$P = \frac{128}{150} \times 100 = 85{,}3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap penambahan karakter yang bisa dimainkan pada *game the survival* adalah 85,3% dari 100%, atau dapat dikategorikan bahwa pemain sangat setuju. Berikut kriteria interpretasi skor :




Gambar 4.22 Interpretasi Skor Pertanyaan Ke-7