BAB 3

METODE PENELITIAN

3.1 Analisis Kebutuhan

Game yang akan menjadi media implementasi dari teknologi PUN adalah game *genre survival* dengan sudut pandang orang ketiga. *Game* dimulai dengan masuknya seorang *player* kemudian sistem mengecek apakah terdapat *room* yang tersedia, jika tidak *player* pertama akan membuat *room* sendiri termasuk NPC di dalamnya dan menunggu *player* lain untuk datang, namun jika sudah ada *room* yang tersedia maka dia akan langsung masuk ke dalam *room* tersebut. Setelah itu proses selanjutnya adalah proses sinkronisasi data antar *player* selama *game* berlangsung. Berdasarkan alur tersebut maka dapat dispesifikasikan kebutuhan-kebutuhan sistem sebagai berikut :

3.1.1 Kebutuhan *Input*

Kebutuhan input yang digunakan dalam sistem ini adalah :

1. Nama *player* berupa *string* yang dibatasi hingga 7 karakter.
2. *Event* dan *status player* yang diambil pada saat bermain.
3. *Event* dan *status* NPC yang diambil pada saat bermain.
4. Data *room* yang disinkronisasikan pada saat *game* dimulai.
5. Objek 3D sebagai *player*, NPC dan *Room*.
6. Input navigasi dari player yang diambil dari pergerakan *virtual joystick*.
7. Aksi player yang diambil dari beberapa tombol aksi.

3.1.2 Kebutuhan Proses

Kebutuhan proses yang terdapat pada sistem ini adalah sebagai berikut :

1. Proses pengacakan *player* yang ikut serta dalam satu *room*.
2. Proses pendistribusian *event* dan *status* dari NPC ke semua *player*.
3. Proses sinkronisasi *event* dan *status* antar *player*.

3.1.3 Kebutuhan *Output*

Aplikasi dapat menampilkan keluaran sebagai berikut :

1. *Game* mampu melakukan sinkronisasi *event* dan *status* antar *player* secara *realtime*.
2. Terbentuknya *room* yang akan menjadi area permainan.
3. Terbentuknya NPC yang akan menjadi musuh dari *player*.
4. Menampilkan keterangan akhir.

3.2 Kebutuhan perangkat lunak

Analisis kebutuhan perangkat lunak terdiri dari sistem operasi dan *software* pembangun. Berikut adalah perangkat lunak yang digunakan untuk membangun aplikasi:

1. Sistem Operasi menggunakan *Windows 7*
2. UNITY 3D versi 2018.3.1
3. Visual Studio Code
4. *Android OS 4.1 or later*

3.3 Kebutuhan perangkat keras

Kebutuhan perangkat keras yang diperlukan untuk membangun Aplikasi ini berupa perangkat dengan spesifikasi sebagai berikut :

a. Spesifikasi Komputer

1. *GPU : DX10 (shader model 4.0) capabilities. Supports a minimum display resolution of 720p (1280 by 720);*
2. *CPU : SSE2 instruction set support and 1.8 GHz or faster processor. Quad-core or better recommended.*
3. *Minimal Disk Space : 5,9 GB.*
4. *RAM : 8 GB.*

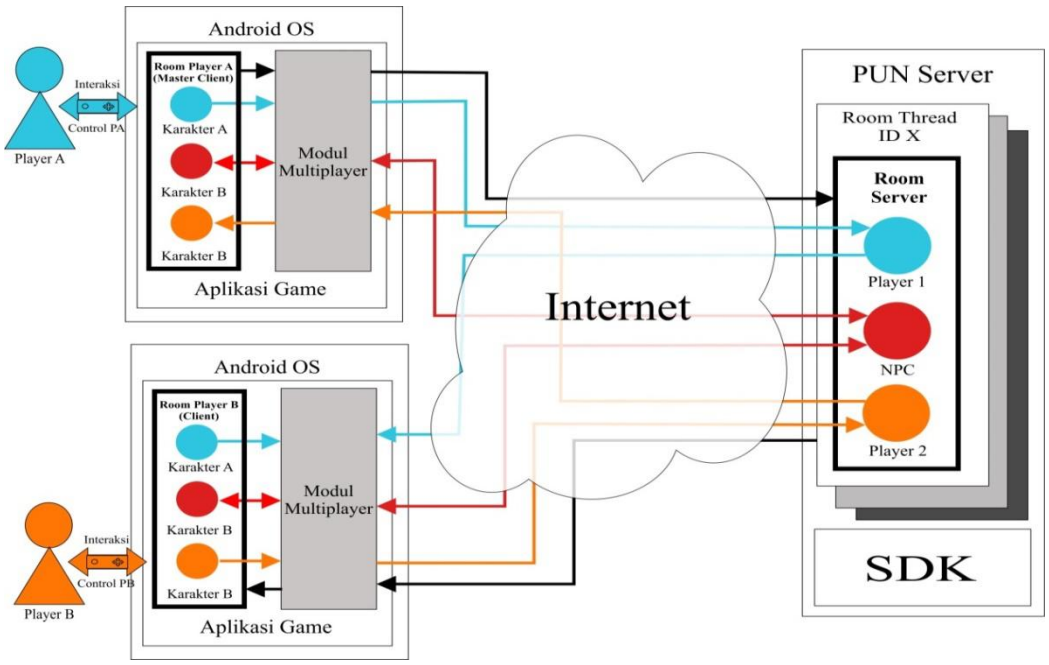
b. Spesifikasi *Smartphone*

Smartphone ARMv7 CPU with NEON support or Atom CPU; OpenGL ES 2.0 or later

3.4 Pemodelan yang Digunakan

3.4.1 Skema Garis Besar Game Multiplayer

Skema garis besar multiplayer untuk game ini digambarkan sebagai berikut :



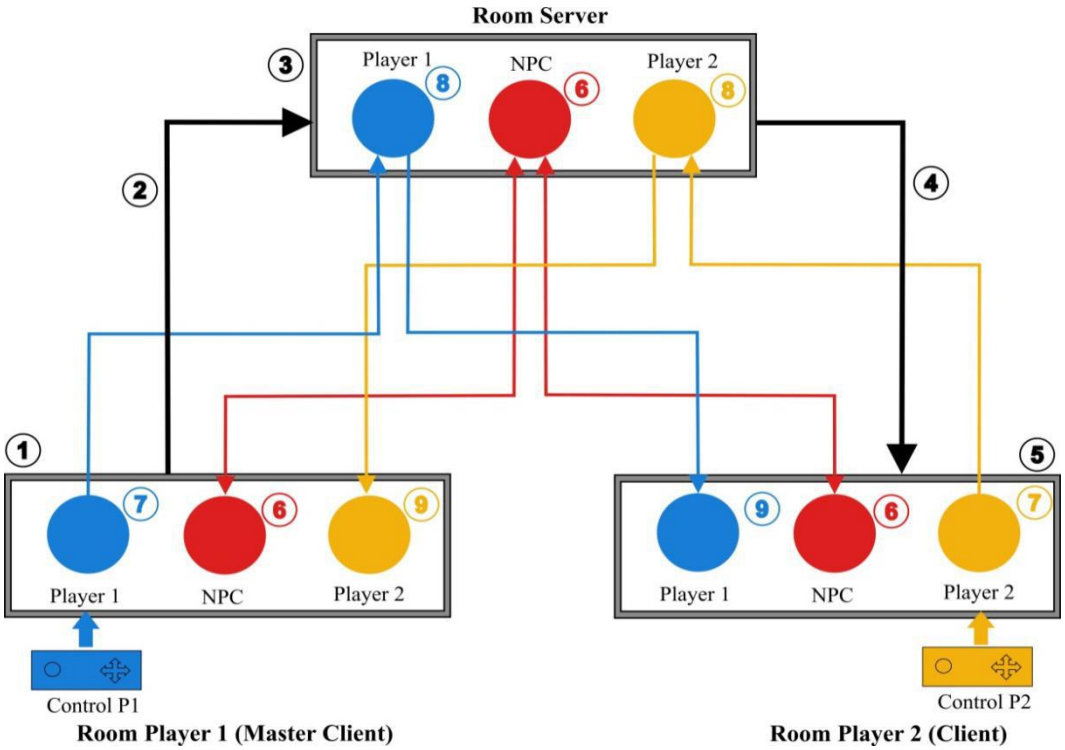
Gambar 3.1 *Layer Multiplayer*

Pada gambar 3.1 adalah gambar yang mempresentasikan hubungan *client-server* dari PUN. Game berdiri di atas sistem operasi android, kemudian dengan menggunakan modul *multiplayer* data dari setiap *player* disinkronisasikan melalui jaringan internet. Modul *multiplayer* menghubungkan game dengan *server* PUN.

Satu sesi permaian (permaianan yang berlangsung dalam satu *room*) akan dilayani oleh sebuah thread. *Thread* tersebut diciptakan oleh program *server* yang berjalan di atas SDK *Photon Server* untuk melayani sinkronisasi antar *player*. *Thread* yang diciptakan dapat diberi nama sesuai keinginan ataupun nama acak. *Thread* memiliki batasan jumlah *player*, pada kasus ini batasan klien yang ditangani oleh satu *thread* adalah jumlah maksimal player dalam satu *room*.

3.4.2 Sinkronisasi *Multiplayer*

Sinkronisasi *multiplayer* dapat digambarkan sebagai berikut :



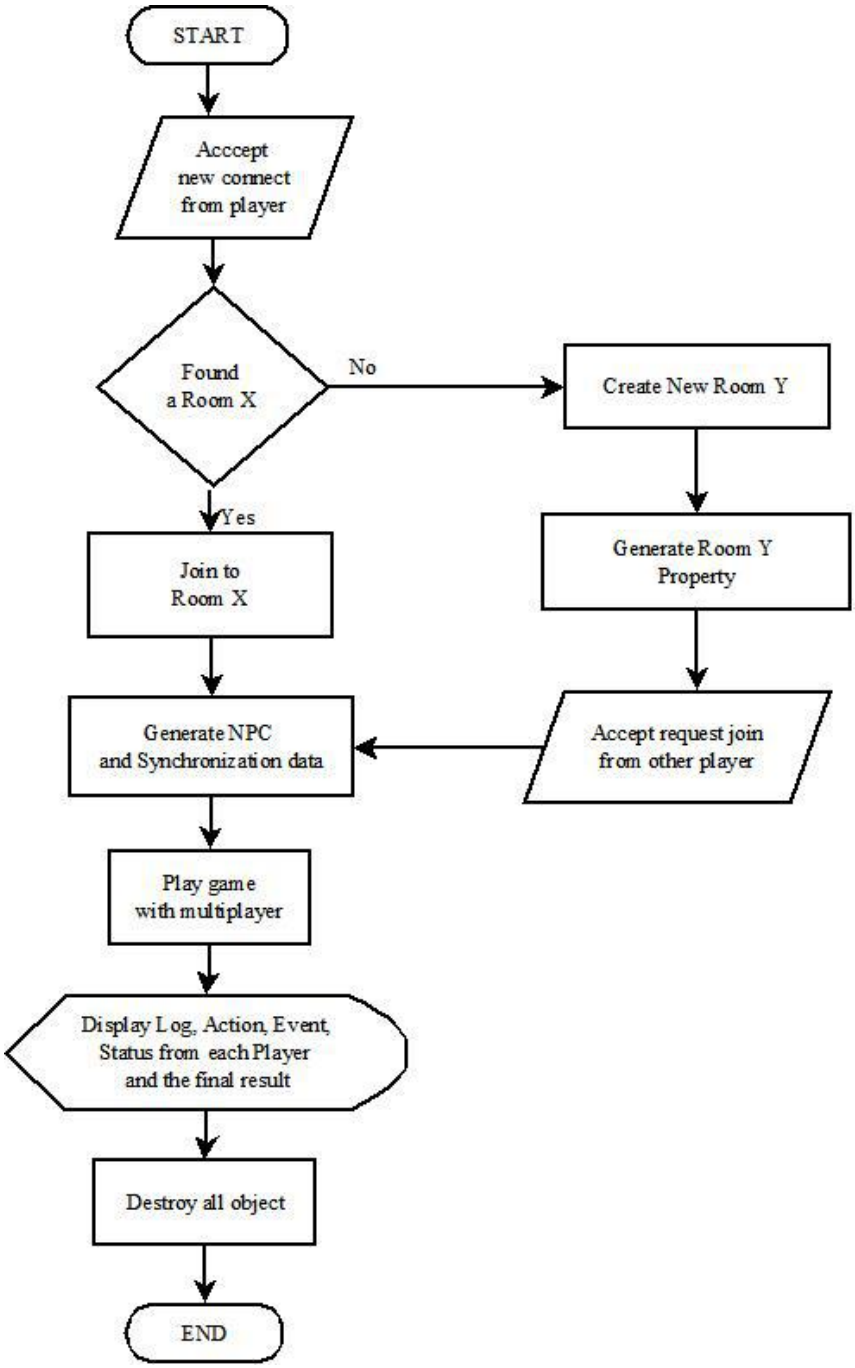
Gambar 3.2 Sinkronisasi *Multiplayer*

Pada gambar 3.2 adalah grafik yang menggambarkan sinkronisasi antar *player* melalui jaringan yang dibuat *server*. Pada grafik ini menggunakan permisalan 2 player. Urutan proses diuraikan sebagai berikut :

1. Ketika permainan dimulai *player* 1 akan berperan sebagai *Master Client* dan membuat ruangan labirin, karakter dan NPC. Kecuali karakter *player* 2 dibuat ketika ada *player* 2 memasuki labirin.
2. Data yang telah dibuat dikirimkan ke *server* PUN.
3. Data tersimpan di *server* PUN, dan siap didistribusikan ke *player* lain.
4. Ketika terjadi koneksi dari *player* 2 saat itu data dikirimkan ke *player* 2. *Player* 2 juga membuat karakter yang disinkronisasikan ke *player* 1.
5. Sehingga pada *player* 2 akan memiliki data ruang labirin, karakter dam NPC yang kondisinya sama seperti pada *player* 1. Begitupun pada *room player* 1 akan terdapat karakter dari *player* 2.
6. Sedangakan untuk NPC, sejak pertama diciptakan dia sudah dapat menerima input data dari semua player yang terhubung dan mensinkronisasikan data yang dia miliki. Data tersebut digunakan untuk proses dalam NPC.
7. Pada saat permainan berlangsung, *player* 1 maupun 2 akan mengendalikan karakternya masing–masing.
8. Data kendali *player* 1 dan 2 dikirimkan ke *server* PUN untuk didistibusikan satu sama lain.
9. Karakter player 1 berjalan di *room player* 2 dikendalikan dari kontrol perangkat player 1, begitu pula sebaliknya.

3.4.3 Flowchart Game

Flowchart untuk proses alur game adalah sebagai berikut :



Gambar 3.3 Flowchart

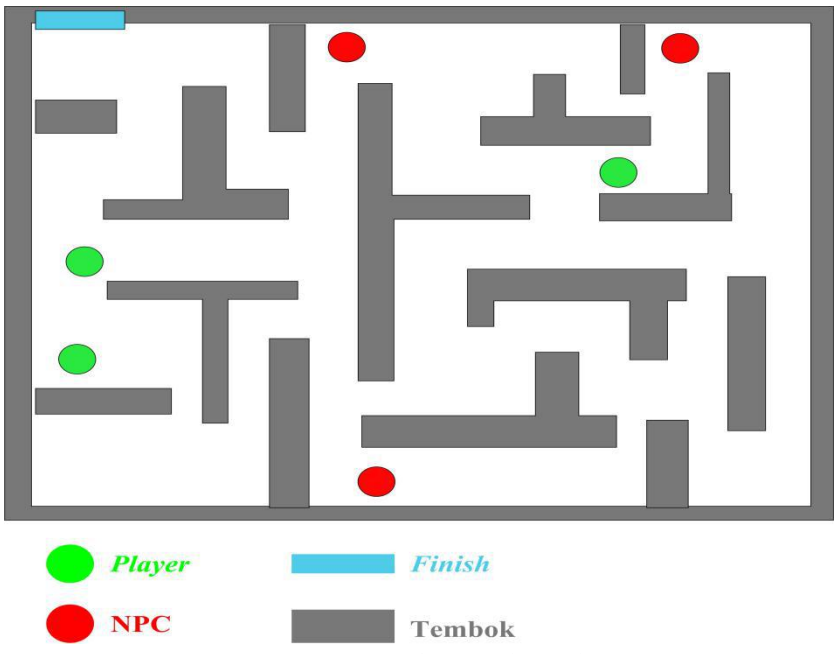
Pada gambar 3.3 adalah *flowchart* yang menggambarkan algoritma pada *server* PUN. Algoritma yang digambarkan merupakan algoritma dari sebuah thread yang dijalankan di *server Photon*. Jika dijabarkan algoritmanya akan sebagai berikut :

1. Pertama setelah program pada *server* aktif, program akan menunggu adanya *request* dari klien (*request* dari klien terjadi ketikap *player* membuka dan memulai game). *Request* ini berupa permintaan membuka koneksi untuk aplikasi klien sebagai *player* baru yang akan bermain.
2. Kemudian program memeriksa apakah tersedia *room* yang kosong (misal *room* X), disini terdapat dua kemungkinan yaitu, pertama *room* tersedia artinya terdapat suatu *room* yang memiliki slot yang masih dapat diisi, atau yang kedua *room* tidak tersedia, baik tidak ada *room* yang memili slot yang dapat diisi ataupun memang tidak ada *room* yang aktif pada saat itu.
3. Jika *room* tersedia maka algoritma akan berlanjut ke tahap penggabungan *player* terhadap *room* tersebut. Disini *player* akan berperan sebagai klien biasa.
4. Jika *room* tidak tersedia maka klien ini akan meminta kepada *server* untuk dibuatkan *room* baru (dengan kata lain *room* Y). Setelah dibuatkan *room*, lanjut ke proses *generate* berbagai *object* yang dibutuhkan untuk bermain seperti ruangan labirin dan termasuk *object player* itu sendiri. Disini *player* akan berperan sebagai *Master Client*. Selanjutnya program akan menunggu *request* dari player lain (algoritma 2 bagi player lain) dengan batasan waktu tertentu. Ketika kondisi dianggap cukup untuk memulai

permainan permainan akan langsung dimulai. Saat permainan dimulai *player* akan dipindahkan ke dalam area labirin secara terpisah saat itu juga NPC mulai diaktifkan untuk mengejar *player.*

5. Kemudian saat bermain, terjadi sinkronisasi antar *player* yang terdapat dalam satu *room*, mereka berbagi informasi, aksi, *event* dan status mereka. Pada tahap ini adalah momen ketika para *player* harus menyelesaikan misi mereka dengan kondisi tertentu. Sampai permainan diselesaikan akan muncul keterangan apakah misi berhasil atau gagal kemudian permainan berakhir.
6. Terakhir setelah permainan berakhir program *server* akan memusnahkan semua *object* yang telah dibuat dalam room tersebut dan pada akhirnya memusnahkan *room* guna membersihkan sumber daya untuk permainan selanjutnya.

3.4.4 Desain *Storyboard*



Gambar 3.4 *Storyboard*

Pada gambar 3.4 adalah *storyboard* saat game dimainkan dengan asumsi Player lengkap berjumlah 3 orang (lingkaran berwarna hijau), dengan lawan 3 NPC (lingkaran berwarna merah). Ruangan labirin dan NPC yang menjadi arena dan lawan para Player adalah merupakan Object yang dibuat secara otomatis oleh *Master Client* dengan menggunakan algoritma tertentu.

Saat memasuki permainan pertama kali para Player akan ditempatkan secara bersamaan dalam suatu bagian ruangan. Misi mereka adalah keluar dari dalam labirin dengan selamat, mereka harus bertahan hidup melawan NPC dan Player lain. *Player* yang menjadi pemenang adalah *Player* yang berhasil bertahan hidup atau keluar dari dalam labirin terlebih dahulu.

Aturan dalam game ini adalah palyer yang ikut serta harus melawan player lain dan menghindari 3 NPC untuk keluar dari labirin dengan selamat. NPC akan mengejar siapapun yang paling dekat dengannya, seorang player dapat dikejar oleh 1 atau lebih NPC. Perlu diperhatikan bahwa berapapun jumlah player yang ikut serta jumlah NPC akan tetap 3. Balok biru pada gambar adalah contoh pintu keluar dari labirin.

3.4.5 Skema Pengujian

Pada penelitian ini, untuk memastikan berhasil atau tidaknya penerapan teknologi PUN, untuk menjadi sarana *backend multiplayer* dari sebuah *game survival* yang akan dibuat, maka dilakukan skema pengujian dengan parameter antara lain :

1. Jaringan, yaitu jaringan yang digunakan oleh perangkat mobile untuk terhubung ke internet berupa jaringan wi-fi atau selular.

2. Jumlah NPC, adalah banyaknya NPC yang akan menjadi musuh player dalam room selama bermain.
3. Jenis Labirin statis atau dinamis (*auto-generate*), adalah labirin yang digunakan sebagai *environment,* perbedaan kedua jenis labirin tersebut adalah; labirin statis berupa labirin yang bentuknya tidak berubah dan source sudah terdapat dalam game yang terinstal di masing-masing perangkat player, sedangkan labirin dinamis bentuknya selalu berubah setiap kali mulai permainan, dan source tidak selalu terdapat dalam game yang terinstal di masing-masing perangkat player, namun di-*generate* oleh *Master Client* kemudian disinkronisasikan ke semua player.

Dengan menggunakan parameter-parameter tersebut hasil yang ingin dicapai dari kasus sederhana sampai kompleks adalah latency yang kurang dari 100 ms antar player. Untuk memperoleh satu nilai latency dari satu kondisi, setiap kondisi akan diuji sebanyak beberapa kali, kemudian diambil nilai rata-rata dari satu kondisi tersebut.

Skema Pengujian ditunjukan dengan table sebagai berikut :

Tabel 3.3 Tabel Pengujian

No.	Kondisi yang diujikan			Latency /ms
	Jaringan	**Jumlah NPC**	**Jenis Labirin**	
1.	*Wi-fi*	1	Statis	
2.	Selular	1	Statis	
3.	*Wi-fi* dan salular	1	Statis	
4.	*Wi-fi*	2	Statis	
5.	Selular	2	Statis	
6.	*Wi-fi* dan salular	2	Statis	
7.	*Wi-fi*	3	Statis	
8.	Selular	3	Statis	
9.	*Wi-fi* dan salular	3	Statis	

10.	*Wi-fi*	1	Dinamis	
11.	Selular	1	Dinamis	
12.	*Wi-fi* dan salular	1	Dinamis	
13.	*Wi-fi*	2	Dinamis	
14.	Selular	2	Dinamis	
15.	*Wi-fi* dan salular	2	Dinamis	
16.	*Wi-fi*	3	Dinamis	
17.	Selular	3	Dinamis	
18.	*Wi-fi* dan salular	3	Dinamis	