PENERAPAN *DYNAMIC LIGHTING* PADA *2D ENDLESS RUNNER GAME* MENGGUNAKAN *VISIBILITY POLYGON COMPUTATION*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Arda Satata Fitriajie
NIM: 155150200111216

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENERAPAN *DYNAMIC LIGHTING* PADA *2D ENDLESS RUNNER GAME*
MENGGUNAKAN *VISIBILITY POLYGON COMPUTATION*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Arda Satata Fitriajie
NIM : 155150200111216

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Februari 2019
Telahh diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Eriq Muhammad Adams Jonemaro, S.T, M.Kom	Tri Afirianto , S.T, M.T
NIP: 198504102012121001	NIK: 2013098512131001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 14 Januari 2019

Arda Satata Fitriajie
NIM: 155150207111078

PRAKATA

Puji dan syukur penulis penjatkan kepada Tuhan Yang Maha Esa atas berkat, tuntunan dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan *Dynamic Lighting* Pada *2D Endless Runner Game* Menggunakan *Visibility Polygon*" sebagai salah satu persyaratan untuk menyelesaikan studi di Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.

Penulis menyadari bahwa skripsi dapat terselesaikan berkat bantuan, petunjuk, bimbingan dan dukungan dari berbagai pihak yang telah banyak membantu proses penyelesaian tugas akhir ini. Oleh karena itu penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Tri Afirianto, S.T, M.T yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
2. Bapak Agus Wahyu Widodo, S.T, M.Cs. selaku Ketua Program Studi Teknik Informatika,
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika,
4. Orang tua ayahanda Ir. Imam Santoso, S.T, M.T dan ibunda Dr. Dina Poerwoningsih, S.T, M.T. serta keluarga yang tak henti hentinya memberikan dukungan moril dan materil,
5. Seluruh dosen Fakultas Ilmu Komputer Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis sehingga penulis dapat menyelesaikan skripsi ini,
6. Teman teman satu proyek game endless runner Ade, Yosua, dan Muaz yang selalu solid dalam menyelesaikan proyek game penelitian ini,
7. Kanjeng Mami selaku ibunda dari Andy yang selalu menyediakan tempat untuk mengerjakan penelitian ini dan menyediakan pesenkopi agar penelitian ini berjalan lancar,
8. Teman teman Slathem Studio yang selalu bersama dalam menyelesaikan penelitian ini,
9. Teman teman yang berpengaruh dalam penyelesaian studi dan skripsi ini Ade, Vio, Yosua, Muadz, Andik, Wira, Ricky, Ary, Rezka, Yohana dll. yang menjadi motivasi dan semangat dalam terselesaikannya penelitian ini,
10. Sahabat dari SMA Garda, Andrianto, dan Wina serta teman-teman berkumpul Aldy, Aldien, Aldino, Andik, Fariz, Ferdio, Satria, dan Nopret yang selalu ada untuk berkumpul minum kopi dan memberikan saran, dukungan, perhatian, dan nasihat untuk terus maju dalam menyelesaikan studi dan skripsi ini,
11. Teman dan partner CV. Cakralaksana Sejahtera Arman, Helmy, dll. yang selalu membimbing dan membagikan pengalaman berharga yang membantu penulis dalam menyelesaikan skripsi ini,

12. Teman teman seperjuangan Fakultas Ilmu Komputer angkatan 2015 yang telah memberikan bantuan selama masa studi hingga penyelesaian skripsi ini
13. Serta Semua pihak yang tidak dapat penulis sebutkan satu persatu yang terlibat secara langsung maupun tidak langsung dalam proses pengerjaan skripsi ini.

Penulis sadar bahwa dalam penyusunan skripsi in masih banyak kekurangan, sehingga saran dan kritik yang membangun dapat disampaikan secara langsung untuk pengembangan dan penelitian selanjutnya.

Malang, 14 Januari 2019

Arda Satata Fitriajie
ardasatata@gmail.com

ABSTRAK

Arda Satata Fitriajie, Penerapan *Dynamic Lighting Pada 2D Endless Runner Game* Menggunakan *Visibility Polygon Computation*

Pembimbing: Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Tri Afirianto, S.T., M.T

Penelitian ini bertujuan untuk mengembangkan modul *dynamic lighting* pada *video game* dengan lingkungan *2D* yang memiliki jenis *endless runner* yang menggunakan prinsip *visibility polygon*. Salah satu jenis efek grafis yang ada pada video game yaitu *dynamic lighting*, efek tersebut umumnya diterapkan pada *game 3D* yang mana efek tersebut dapat membawa kesan realistis dan *immersive* pada game yang dimainkan. Implementasi *dynamic lighting* pada game *2D* pada penelitian sebelumya diimplementasikan menggunakan metode yang digunakan game 3D yaitu *normal mapping*, teknik tersebut lebih umum digunakan pada game *top-down* dan game dengan *style isometric*, sedangkan pada game yang akan diimplementasikan dapat digunakan teknik *visibility polygon* karena perspektif game yang dilihat dari samping. Pada penelitian ini akan diimplementasikan *dynamic lighting* menggunakan konsep *visibility polygon* karena sifat cahaya sama dengan konsep pengelihatan yang ada pada konsep *visibility polygon* pada bidang datar dua dimensi *visibility polygon* pada penelitian ini diimplementasikan menggunakan *raycasting* yang ditembakan dari suatu titik dan menyebar seperti sifat cahaya yang mana cahaya direpresentasikan dengan satu objek *ray* yang dipancarkan oleh modul. Berdasarkan kebutuhan diatas maka akan diimplementasikan modul *dynamic lighting* pada game *2D endless runner* yang menghasilkan modul *dynamic lighting* yang diimplementasikan menggunakan *raycasting* yang memvisualisasikan *visibility polygon* dan didapatkan hasil pengujian fungsional 100% valid dan pengujian performa fps 2 kasus uji yang mendapat hasil *fps* diatas 24.

Kata kunci: *Dynamic Lighting, 2D Game, Endless Runner, Visibility Polygon, Raycasting*

ABSTRACT

Arda Satata Fitriajie, *Dynamic Lighting Implementation in 2D Endless Runner Game Using Visibility Polygon Computation*

Supervisors: Eriq Muhammad Adams Jonemaro, S.T, M.Kom dan Tri Afirianto, S.T., M.T

This study aims to develop dynamic lighting modules in video games with 2D environments that have endless runner types that use the principle of polygon visibility. One type of graphic effect that exists in video games is dynamic lighting, this effect is generally applied to 3D games where the effect can bring a realistic and immersive impression on the game being played. The implementation of dynamic lighting in 2D games in the previous study was implemented using a method used in 3D games, namely normal mapping, the technique is more commonly used in top-down games and game with isometric style, whereas in the game to be implemented polygon visibility techniques due to game perspective which is seen from the side. In this study, dynamic lighting will be implemented using the concept of polygon visibility because the light nature is the same as the vision concept that exists in the concept of polygon visibility in the flat two-dimensional polygon visibility in this study implemented using raycasting which is shot from a point and spreads like light where light represented by one ray object emitted by the module. Based on the above requirements, dynamic lighting modules will be implemented in 2D endless runner games that produce dynamic lighting modules that are implemented using raycasting that visualize polygon visibility and obtain 100% valid functional test results and fps performance testing of 2 test cases that get fps above 24.

Keywords: *Dynamic Lighting, 2D Game, Endless Runner, Visibility Polygon, Raycasting*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Permainan *Video* atau *Video Game* saat ini adalah salah satu hiburan yang saat ini tidak hanya dinikmati oleh anak-anak namun juga orang dewasa,menurut artikel dari huftingpost video game sangat populer di Amerika bahkan penjualan konten dari *video game* di amerika melebihi penjualan dari konten musik dan film *box office* digabungkan keduanya (Rick Taylor, 2014). *Video game* saat ini tidak semua disajikan dalam *3D*, namun game *2D* juga cukup popular karena pada awalnya game disajikan dalam 2D, bahkan menurut *the verge* game bertema *pixel art* tidak lagi dianggap *retro* namun menjadi bagian dari tren dari tema game 2D saat ini (Sam Byford, 2014), salah satu *subgenre* dari *game 2D* yaitu adalah *endless runner* yang mana *endless runner* ini secara prinsip adalah game *2D platformer*, Secara umum *game* dengan *genre* tersebut memiliki ciri khas grafis yang sederhana dan tidak menampilkan visual yang realistis (Keo, 2017), Namun menurut survei dari *cinemablend* saat ini konsumen video game mengatakan bahwa grafis pada video game adalah aspek penting saat membeli game, grafis pada game yaitu mencakup seberapa *immersive* grafis dan efek grafis game tersebut agar pengalaman dan emosi pemain saat sedang bermain(William Usher, 2014).

Salah satu jenis efek grafis pada video game yaitu *dynamic lighting*, efek tersebut umumnya diterapkan pada game 3D yang mana efek tersebut dapat membawa kesan realistis pada lingkungan game yang dimainkan. DOOM 3 (2004) adalah salah satu game 3D yang mengimplementasikan *dynamic lighting* pada masanya, parameter lighting yang digunakan meliputi posisi, orientasi, saturasi, kecerahan dan temperature warna. Penggunaan *dynamic lighting* dengan parameter tersebut agar pemain dapat merasakan tensi dan tekanan pada saat muncul monster tertentu sehingga kesan keterlibatan pemain dengan lingkungan dalam game menjadi kuat dan pemain merasa benar-benar ada pada didalam permainan (El-Nasr, 2005). Implementasi *dynamic lighting* pada game 2D pada penelitian sebelumya diimplementasikan menggunakan metode yang digunakan game 3D yaitu normal mapping, teknik tersebut lebih umum digunakan pada game top-down dan game dengan style isometric (Konferencia, 2013), karena penelitian ini fokus pada game *2D endless runner* yang berbasis *side scroll platformer* maka akan dibahas bagaimana mengimplementasikan *dynamic lighting* yang secara spesifik diimplementasikan pada *2D endless runner* menggunakan metode *visibility polygon* karena metode *normal mapping* tidak dapat menjadi bentuk polygon cahaya karena metode tersebut hanya dapat digunakan untuk memberi kontur/kedalaman pada tekstur 2D sehingga berbeda tujuan pengaplikasiannya dengan *visibility polygon* yang mana ia dapat membentuk polygon cahaya untuk diimplementasikan pada game 2D sebagai modul *dynamic lighting* yang menjadi aspek penting pada gameplay game 2D yang akan dibuat. *Immersive* yg akan dicapai pada penelitian ini yaitu modul

dapat membentuk bentuk polygon yang dinamis layaknya visualisasi pengelihatan pada lingkungan *game 2D*.

Visibility polygon adalah konsep dasar dari komputasi geometri yang menjelaskan bagaimana membentuk *polygon* yang menjadi daerah pada bidang 2D yang menjadi acuan apakah suatu titik pada bidang tersebut dapat terlihat dari titik yang diinginkan (Ghosh, 2007). Pada penelitian ini akan diimplementasikan *dynamic lighting* menggunakan konsep *visibility polygon* karena sifat cahaya sama dengan konsep pengelihatan yang ada pada konsep *visibility polygon* pada bidang datar dua dimensi. Penggunaan konsep *visibility polygon* pada penelitian ini karena *visibility polygon* adalah konsep yang digunakan pada bidang datar yang sesuai dengan lingkungan game 2D yang basisnya adalah geometri bidang datar.

Berdasarkan kebutuhan diatas maka akan diimplementasikan modul *dynamic lighting* menggunakan *visibility polygon* pada game 2D endless runner yang akan dikembangkan pada skripsi ini.

1.2 Rumusan masalah

Rumusan masalah pada penelitian ini dijelaskan sebagai berikut.

1. Bagaimana mengimplementasikan *Dynamic Lighting* menggunakan *visbility polygon*.
2. Bagaimana pengujian dan kinerja modul *dynamic lighting* pada 2D endless runner game.

1.3 Tujuan

Penelitian ini bertujuan membangun modul *dynamic lighting* dengan *dynamic lighting* pada *2D endless runner game* menggunakan *visibility polygon*.

1.4 Manfaat

Implementasi penelitian ini bermanfaat untuk:

1. Bagi Peneliti menerapkan ilmu yang telah dipelajari untuk membuat modul yang digunakan untuk mengembangkan *2D endless runner game*.

2. Bagi Pengembang *Video Game 2D* Memberi alternatif metode untuk mengimplementasikan dynamic lighting pada lingkungan game 2D menggunakan *visibility polygon*.

1.5 Batasan masalah

Skripsi ini menekankan pada pembuatan modul yang berguna untuk mengimplementasi *dynamic lighting* dengan ketentuan:

1. Modul dibuat untuk game pada lingkungan 2D
2. Menggunakan *visibility polygon computation* sebagai konsep dasar pembuatan modul

1.6 Sistematika pembahasan

Sistematika penulisan yang disusun pada tugas akhir ini diharapkan dapat menunjang tujuan yang diharapkan. Adapun sistematika pada penulisan tugas akhir ini, sebagai berikut :

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang penulisan penelitian, rumusan masalah, tujuan penelitian, manfaat penelitian, sistematika penulisan yang digunakan untuk implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon*

BAB II LANDASAN KEPUSTAKAAN

Bab ini membahas tentang langkah-langkah yang digunakan untuk implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon*

BAB III METODOLOGI

Bab ini membahasa tentang langkah-langkah yang digunakan untuk implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon* dan langkah kerja yang dilakukan dalam penelitian

BAB IV PERANCANGAN MODUL

Bab ini membahas tentang bagaimana rancangan implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon*

BAB V IMPLEMENTASI

Bab ini membahas tentang implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon*

BAB VI PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian serta analisis terhadap implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon*

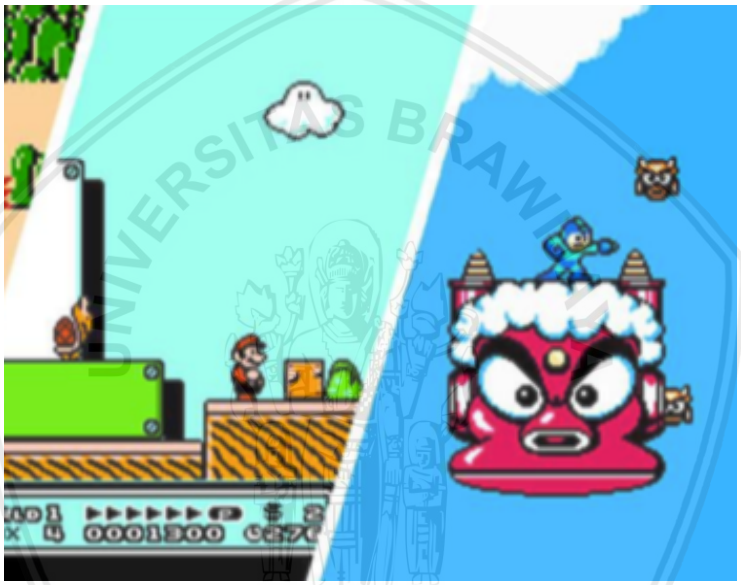
BAB VII PENUTUP

Bab ini berisi tentang kesimpulan yang diperolah dari pembuatan dan pengujian implementasi Dynamic Lighting pada 2D endless runner game menggunakan *visibility polygon* yang dikembangakan dalam skrpsi ini serta saran-saran untuk pengembangan sistem selanjutnya

BAB 2 LANDASAN KEPUSTAKAAN

2.1 2D Platformer

2D Platformer adalah suatu *sub-genre* dari *video game* dan dapat dikatergorikan sebagai *game puzzle* ataupun *game action*. Konsep utama dari game *2D platformer* adalah *player* mengontrol karakter pada *game* yang melewati *platform* yang terdapat di dasar ataupun yang *platform* melayang yang tergantung pada desain dari *level* yang dimainkan. Tujuan dasar game *platformer* yang umum adalah mengkoleksi item seperti *coin* atau berlian dan menghindari/melawan karakter musuh (Bhosale, Kulkarni dan Patankar, 2018).



Gambar 2.1 *Super Mario Bros 3 dan Mega Man 2*

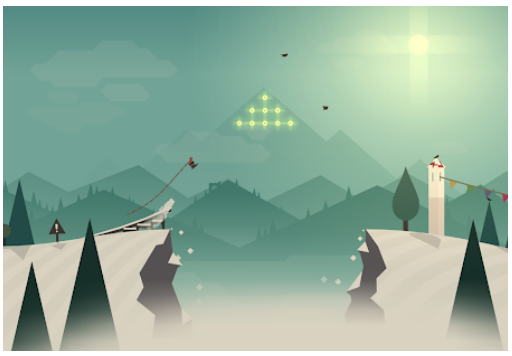
Sumber : Graphical Style in Video Games, 2017

2D Platformer adalah *genre* game yang umum pada era game 8-bit yang dirilis sekitar tahun 1980, dikarenakan keterbatasan kemampuan *hardware console* pada tahun tersebut game yang rilis pada tahun tersebut memiliki gaya grafis yang datar dan tidak memiliki efek yang special karena keterbatasan *hardware* tadi (Keo, 2017), *Super Mario Bros 3* (1988) dan *Mega Man 2* (1988) pada gambar 2.1 adalah contoh game *platformer 2D* yang populer pada masanya.

2.2 Endless Runner

Endless Runner game memiliki konsep dasar yaitu player dapat mendapatkan skor/point tanpa batas. Tujuan dari game ini yaitu mendapat point sebanyak-banyaknya. Salah satu contoh adalah game "*Temple Run*" pada gambar 2.3, yang mana player terus bermain sampai karakter menabarak *obstacle*. *Game Endless* dapat diimplementasikan dalam berbagai macam cara:

endless runner, endless flyer, endless avoidance secara dasar adalah interaksi pada game yang dapat diulang secara tak terbatas (Cao et al., 2014).



Gambar 2.2 Alto's Adventures

Sumber : techcrunch.com

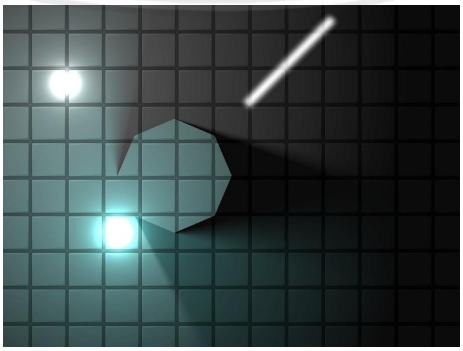
Gambar 2.3 Temple Run

Sumber : kotaku.com

Sebagai contoh game *Alto's Adventure* pada gambar 2.2 adalah implementasi *2D endless runner* pada platform *mobile*, game ini banyak mendapat nominasi karena banyak yang mengatakan game ini memiliki gaya grafis yang unik dan mekanisme yang menarik. Game ini memiliki gaya flat dan efek yang digunakan adalah efek cahaya dan partikel yang menjadikan game ini menarik untuk dimainkan (Federico Viticci, 2015).

2.3 Dynamic Lighting

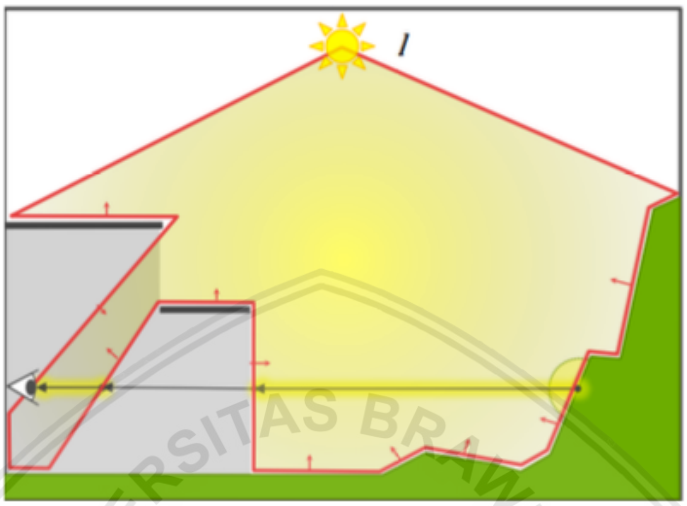
Dynamic Lighting adalah simulasi pencahayaan yang komputasinya dilakukan secara *real-time*. Karena itu penggunaan *dynamic lighting* memberikan efek cahaya yang dikalkulasi secara *realtime* bergantung pada kondisi yang terjadi pada lingkungan video game seperti narasi, posisi player ataupun pergerakan kamera. Efek ini dapat memperkuat interaksi, emosi dan dramatisasi yang terjadi pada game dibandingkan menggunakan metode *static lighting*. *Dynamic lighting* awalnya umum digunakan pada lingkungan game 3D namun saat ini game 2D juga mulai menggunakan Teknik *dynamic lignting* (Konferencia, 2013), contoh implementasi pada lingkungan 2D seperti pada gambar 2.4.



Gambar 2.4 *Contoh Dynamic Lighting pada bidang 2D*

Sumber : github.com/eXpl0it3r/LTBL

Penggunaan *Dynamic Lighting* pada video game dapat memperkuat pengalaman player pada video game dari sisi player, apabila dari sisi pengembang penggunaan *dynamic lighting* dapat memberi ruang berkreasi kepada desainer *game* dan memberi pengaruh positif terhadap game *aesthetics* (El-Nasr et al., 2006).

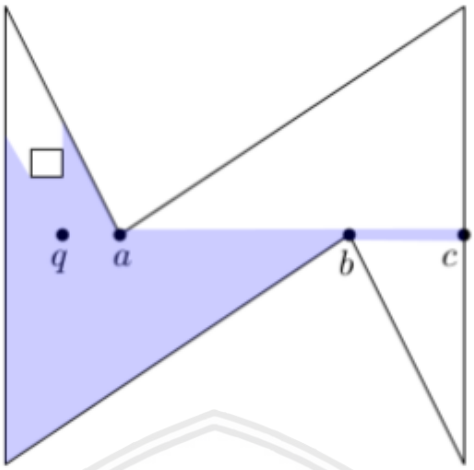


Gambar 2.5 *Ilustrasi Polygonal Volumetric Lighting*

Sumber : gamework.nvidia.com

Salah satu cara implementasi *dynamic lighting* adalah dengan menggunakan metode *Polygonal Volumetric Lighting* seperti pada gambar 2.5, cara dikenalkan oleh hobobler pada *GDC 2016*, dan dikembangkan untuk *game Fallout 4*. Metode ini diklaim sangat cepat, fleksibel dan mudah Diimplementasikan pada *engine* yang sudah ada. Implementasinya secara sederhana adalah dengan seperti bangun ruang yang memiliki sifat cahaya di dalamnya, dan bentuk dari bangun tersebut mengisi bagian kosong seperti sifat cahaya (Osipovic, 2017).

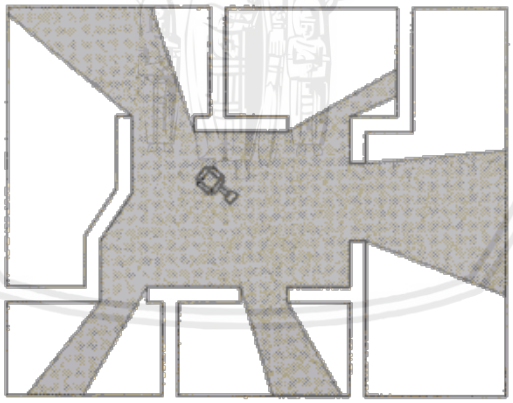
2.4 Visibility Polygon



Gambar 2.4 *Visibility Polygon*

Sumber : doc.cgal.org/latest/Visibility_2

Visibility Polygon adalah konsep dasar pada komputasi geometri yang mana digambarkan pada Gambar 2.4 yaitu dari titik mana saja yang dapat terlihat dari titik q apabila ditarik garis menuju 2 titik tersebut yang mana garis tersebut masih didalam bidang *polygon* yang terbentuk. Polygon yang menggambarkan area *polygon* yang dapat dilihat/diobservasi dari titik q inilah yang disebut *Visibility Polygon* (Bungiu, 2014).



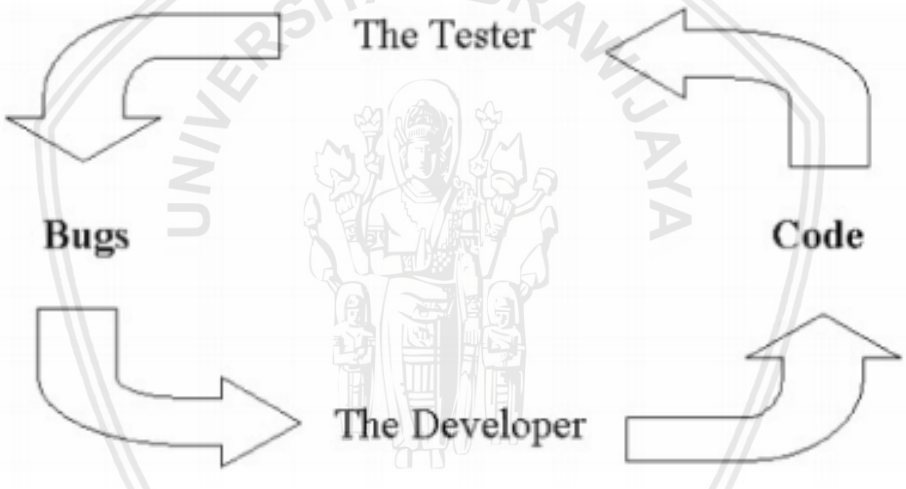
Gambar 2.5 *Art Gallery Problem*

Sumber : personal.kent.edu/~rmuhamma/Research/ArtGallery/artTrans.html

Visibility Polygon juga erat hubungannya dengan *art gallery problem*, yang mana *art gallery problem* adalah bagaimana mencari solusi optimal untuk menempatkan berapa jumlah pengawas pada suatu gallery/museum. Kemudian visualisasi pengelihatan atau area yang dapat diawasi dari suatu titik yang digambarkan menjadi polygon apabila dilihat dari atas (Ghosh, 2007).

2.5 Game Testing

Game Testing adalah suatu langkah penting dalam pengembangan game. Sebuah game di tes dengan berbagai perbedaan level pada tahapan pengembangannya. Secara umum pada bidang rekayasa perangkat lunak, *testing* perangkat lunak adalah perencanaan dokumen yang mengandung segala informasi tentang *testing* software tersebut. Namun *testing* pada game berbeda dengan *testing* pada software atau perangkat lunak. Ada banyak tahapan dalam melakukan *testing* pada game dikarenakan hampir semua *game testing* menggunakan metode black box *testing*. Para programmer game biasanya tidak melakukan *testing* pada pekerjaan mereka, umumnya mereka hanya melakukan sebagian kecil *testing* pada kode mereka untuk kemudian dilakukan integrasi pada suatu game. Pada gambar 2.6 dijelaskan siklus dari *game testing* untuk kemudian *testing* pada game dilakukan 6 langkah (Schultz, Bryant dan Langdell, 2005).



Gambar 2.6 Siklus *Testing Game*

Sumber*: Game Testing: All in One*

Rancangan dan uji desain: Meskipun banyak dari rencana ini dan yang dilakukan sebelumnya dalam dokumen rencana uji perangkat lunak, tetapi dengan setiap prototipe permainan yang baru, dokumen ini perlu direvisi untuk memperbarui setiap perubahan dalam spesifikasi, kasus uji baru, dukungan konfigurasi baru. Penguji harus memastikan bahwa tidak ada masalah baru yang diperkenalkan.

Mempersiapkan pengujian: Semua tim harus memperbarui kode mereka, tes, dokumen dan menguji lingkungan dan menyelaraskannya dengan satu sama lain. Tim pengembangan tes harus menandai bug yang diperbaiki, dan waktu pengujian harus memverifikasinya.

Lakukan pengujian: Jalankan setelan uji lagi. Jika ada kecacatan ditemukan, uji sekitar cacat untuk memastikan bug itu ada.

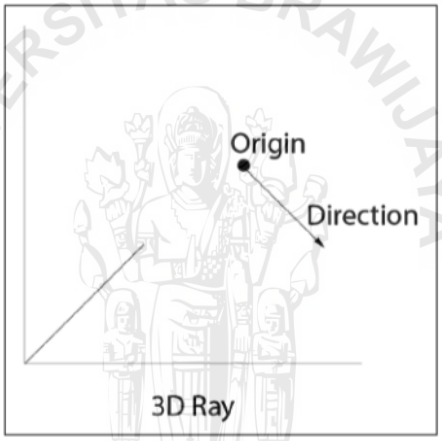
Melaporkan hasil tes: Detail lengkap tentang bug dilaporkan.

Memperbaiki Bug: Tim penguji berpartisipasi dalam langkah ini dengan menjelaskan bug untuk mengembangkan tim dan memberikan pengujian langsung untuk melacak bug.

Kembali ke langkah 1 dan dilakukan uji ulang: Versi baru dihasilkan setelah satu siklus.

2.6 Raycast

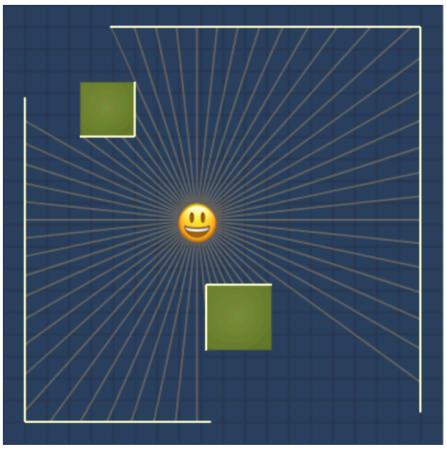
Raycast adalah suatu Teknik pada *computer graphic* yang merepresentasikan bagaimana fisika suatu cahaya pada komputasi grafis pada computer. Penggunaan nama tersebut pada awalnya digunakan pada paper penelitian tahun 1982 oleh Scott Roth untuk mendeskripsikan metode untuk merender mode solid konstruktif geometri (Roth, 1982).



Gambar 2.7 Visualisasi Cahaya Pada Komputer Grafis

Sumber : Game Physics Cookbook

Ray atau cahaya pada grafis computer direpresentasikan dengan titik pada suatu ruang/bidang dana rah. Cahaya mengarah secara tidak terbatas dari titik yang ditentukan menuju ke suatu arah yang telah ditentukan. Pada Gambar 2.7 diketahui cahaya berasal dari suatu titik menuju arah yang telah ditentukan dengan mengasumsikan arah cahaya telah dinormalisasi sebelumnya.



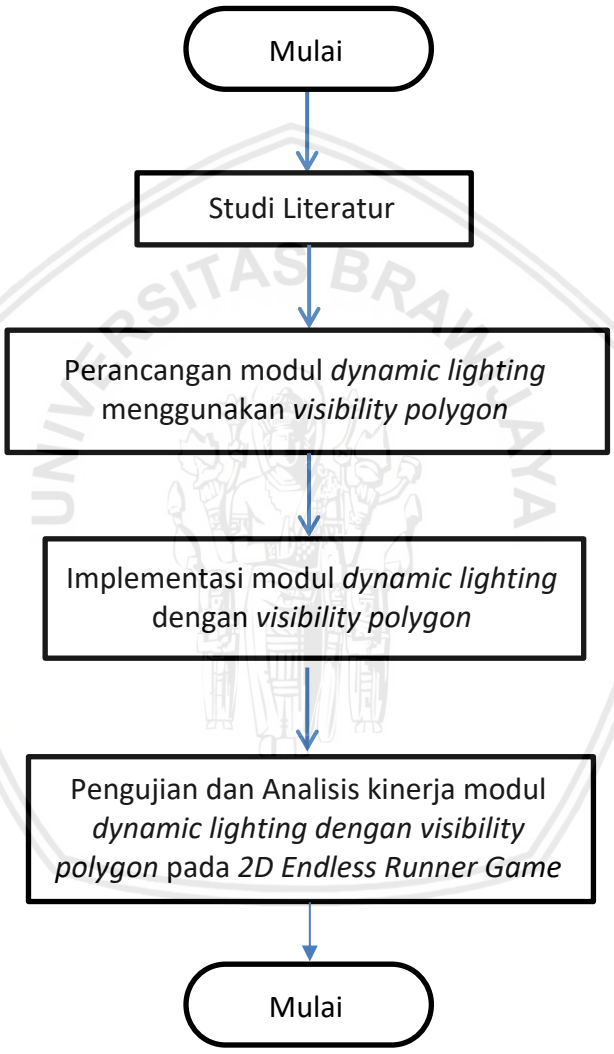
Gambar 2.8 *Visibility Polygon* Menggunakan *Raycast*

sumber : www.redblobgames.com/articles/visibility/

Gambar diatas adalah contoh implementasi *raycast* untuk masalah visibility pada game 2D topdown. Secara sistematis teori yang digunakan yaitu objek atau karakter menembakan cahaya dari arahnya secara melingkar 360 derajat dengan jarak antara cahaya yang ditentukan kemudian dapat terbentuk suatu polygon yang menggambarkan visibilitas dari objek tersebut dari arah atas.

BAB 3 METODOLOGI

Pada bab ini akan menjelaskan langkah langah yang akan dilakukan dalam penyususnan skripsi. Penelitian ini bersifat implementatif dengan megimplementasikan *Dynamic Lighting* menggunakan *visibility polygon*. Metodologi penelitian yang digunakan adalah:



Gambar 3.1 Diagram Alir Metodologi

3.1 Studi Literatur

Studi Literatur merupakan tahap pertama penelitian dalam rangka menyusun dasar teori yang digunakan sebagai dasar pengetahuan dalam penelitian ini. Penelursuran pengetahuan ini bersumber dari buku, jurnal dan internet (website resmi) yang berkaitan dengan informasi yang diperlukan dalam mengerjakan penelitian ini. Teori yang akan dijadikan acuan oleh penulis sebagai landasan dasar dalam melaksanakan penelitian ini yaitu: *Visbility Polygon* dan *Dynamic Lighting*

3.2 Perancangan Modul Dynamic Lighting Menggunakan Visibility Polygon

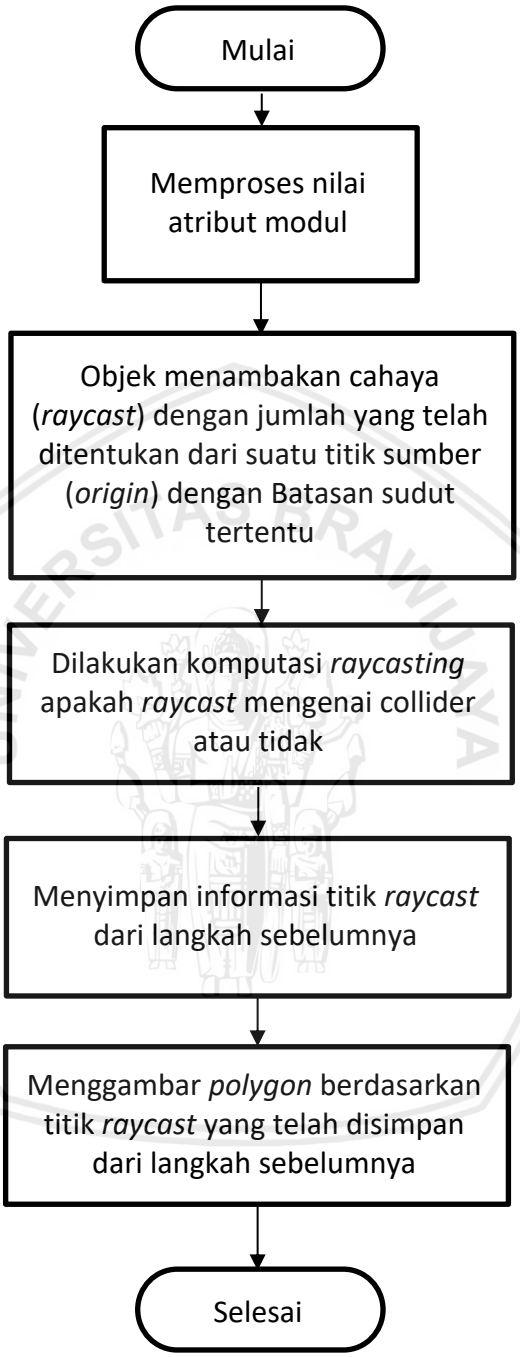
Tahap kedua penelitian dilakukan proses perancangan modul. Pada tahap ini dilakukan penjabaran kebutuhan modul *dynamic lighting* dari 2D endless runner game dan perancangan *pseudocode* dari algoritme yang digunakan modul yang menggunakan *visibility polygon*, kemudian rancangan tersebut digunakan untuk proses implementasi pada tahapan selanjutnya.

Pada table 3.1 dijelaskan parameter modul yang akan diimplementasikan menjadi modul.

Tabel 3.1 Parameter modul yang diusulkan

No	Atribut	Keterangan
1	Radius Cahaya	Jangkauan maksimum cahaya yang dipancarkan dari titik asal cahaya
2	Sudut Cahaya	Besar sudut cahaya dalam skala 1 putaran (360 derajat)
3	Resolusi Cahaya	Banyaknya *ray* (cahaya) dari *raycast* yang ditembakkan

Gambar 3.2 adalah rancangan algoritme *visibility polygon* secara umum yang diusulkan untuk modul *dynamic lighting* yang akan diterapkan.



Gambar 3.2 Diagram Alir Perancangan

3.3 Implementasi Modul Dynamic Lighting Menggunakan Visibility Polygon

Pada tahap implementasi, akan dilakukan proses implementasi dari rancangan subbab 3.2 menjadi *script* C# yang dapat digunakan *GameObject* pada *Unity3D*. Kemudian *Script* dipasangkan pada *GameObject* yang terkait agar dapat diimplementasikan menjadi *dynamic lighting* pada 2D Endless Runner game.

3.4 Pengujian dan Analisis Kinerja Modul Dynamic Lighting Menggunakan Visibility Polygon

Untuk tahap pengujian hasil dari implementasi akan dilakukan uji kinerja pengaruh FPS game dengan jumlah object *dynamic lighting* yang diimplementasikan yang bertujuan untuk memberikan saran konfigurasi parameter modul yang akan diimplementasikan nanti. Pengujian dilakukan dengan cara 5 kali pengujian tiap sesinya ditambahkan 2 object untuk menguji kinerja FPS game terhadap jumlah modul yang digunakan.

Pada Tabel 3.2 adalah contoh tabel uji kinerja yang akan menguji kinerja modul berdasarkan jumlah modul yang bertambah 2 tiap sesi pengujiannya, kemudian modul yang ditambahkan akan memiliki parameter yang sama karena pada pengujian ini akan diuji *Frame Per Second* berdasarkan jumlah modulnya.

Tabel 3.2 Contoh Tabel Uji Kinerja Berdasarkan Jumlah Modul

Tabel Uji Kinerja Berdasarkan Jumlah Modul		
Fase Uji	**Jumlah Modul Pada *Test Case***	***FPS***
1	2	…
2	4	…
3	6	…
4	8	…
5	10	…

Pada Tabel 3.3 adalah contoh table pengujian *Frame per Second* berdasarkan nilai resolusi cahaya yang diterapkan modul. Resolusi cahaya yang dimaksud adalah berapa jumlah *raycast* yang ditembakan dari sumber cahaya berdasarkan sudut yang telah ditentukan.

Tabel 3.3 Contoh Tabel Uji Kinerja Berdasarkan Resolusi Cahaya

Tabel Uji Kinerja Berdasarkan Resolusi Cahaya		
Fase Uji	**Resolusi Cahaya Modul**	**FPS**
1	5	…
2	10	…
3	15	…
4	20	…
5	25	…

Setelah uji kinerja, akan dilakukan pengujian fungsional menggunakan *black-box testing* untuk mengetahui apakah modul yang dibuat memiliki fungsional yang *valid*.

BAB 4 PERANCANGAN

Pada bab ini akan dijelaskan perancangan algoritme dan langkah-langkah perancangan lainnya yang dilakukan pada penelitian ini yaitu Perancangan *Collider*, Perancangan Agoritme *Dynamic Lighting* dan *Raycasting*.

4.1 Perancangan *Collider* Pada *Game*

Pada bab ini akan dijelaskan mengenai perancangan *collider* yang akan diimplementasikan pada *obstacle dan* karakter *player* yang akan menjadi target *raycasting* dari modul yang dibuat. *Collider* yang akan diimplementasikan adalah berbentuk persegi.

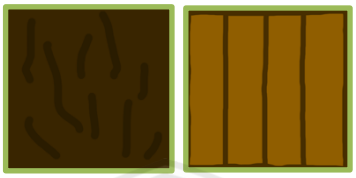
4.1.1 Perancangan *Collider Character Player*

Karakter player akan menjadi salah satu objek yang dapat terkena *raycasting* dari modul yang dibuat. Gambar 4.1 adalah rancangan *collider* yang akan diimplementasikan pada *character player* yang mana pemain akan menggerakan karakter tersebut untuk bergerak. *Collider* berbentuk persegi akan diimplementasikan pada tubuh karakter agar ia dapat terkena *raycast* dari modul yang akan dibuat. *Collider* pada Gambar 4.1 adalah persegi dengan garis berwarna hijau.

Gambar 4.1 Rancangan *Collider Character Player*

4.1.2 Perancangan *Collider* Obstacle

Objek *ground* dan *wall* berbentuk persegi pada game akan diimplementasikan juga *collider* berbentuk persegi sesuai dengan bentuknya. Gambar 4.2 adalah rancangan *collider* yang akan diimplementasikan pada objek *ground* dan *wall* yang mana *collider* ditunjukan dengan persegi dengan garis berwarna hijau.



Gambar 4.2 Rancangan *Collider Obstacle*

4.2 Perancangan Algoritme Dynamic Lighting

Pada sub bab ini akan dibahas mengenai rancangan algoritme yang akan diimplementasikan pada modul. Algoritme ini adalah rancangan modul secara umum yang dijelaskan pada Tabel 4.1.

Tabel 4.1 Pseudocode Modul *Dynamic Lighting* Secara Umum

No	Pseudocode Modul *Dynamic Lighting* Secara Umum
1	`Mulai`
2	`    Masukan parameter modul`
3	`    Proses nilai parameter (radius,sudut,resolusi)`
4	`    for i=0 i<resolusi i++`
5	`        Lakukan Komputasi Raycasting`
6	`        Simpan Informasi Raycasting`
7	`    Membentuk Polygon dari list hasil informasi raycasting`
8	`Selesai`

Secara umum tahap-tahap yang dilakukan adalah pertama modul menerima input parameter yang terdiri dari radius, sudut, dan resolusi. Radius pada modul ini adalah seberapa besar jarak tembak setiap unit *raycast*-nya. Kemudian sudut adalah besarnya sudut tembak cahaya dengan besaran derajat dari 0 sampai 360. Resolusi pada modul ini yaitu seberapa banyak cahaya/*raycast* yang ditembakan dari sumber cahaya.

4.3 Perancangan Algoritme *Raycasting*

Pada Tahap ini akan dijelaskan mengenai perancangan algoritme *raycasting* yang melakukan *handling* setiap tembakan cahaya/*raycast* yang dilakukan pada modul *dynamic lighting* pada subbab 4.2.

Tabel 4.2 Pseudocode Algoritme *Raycasting*

No	Pseudocode Algoritme *Raycasting*
1	`Memasukan nilai arah cahaya (ray) dan Radius Cahaya`
2	`If raycast mengenai collider`
3	`    Jarak Raycast = Titik Terkena Collider`
4	`Else`
5	`    Jarak Raycast = Jarak Maksimum Radius Cahaya`
6	`Kembalikan nilai jarak raycast`

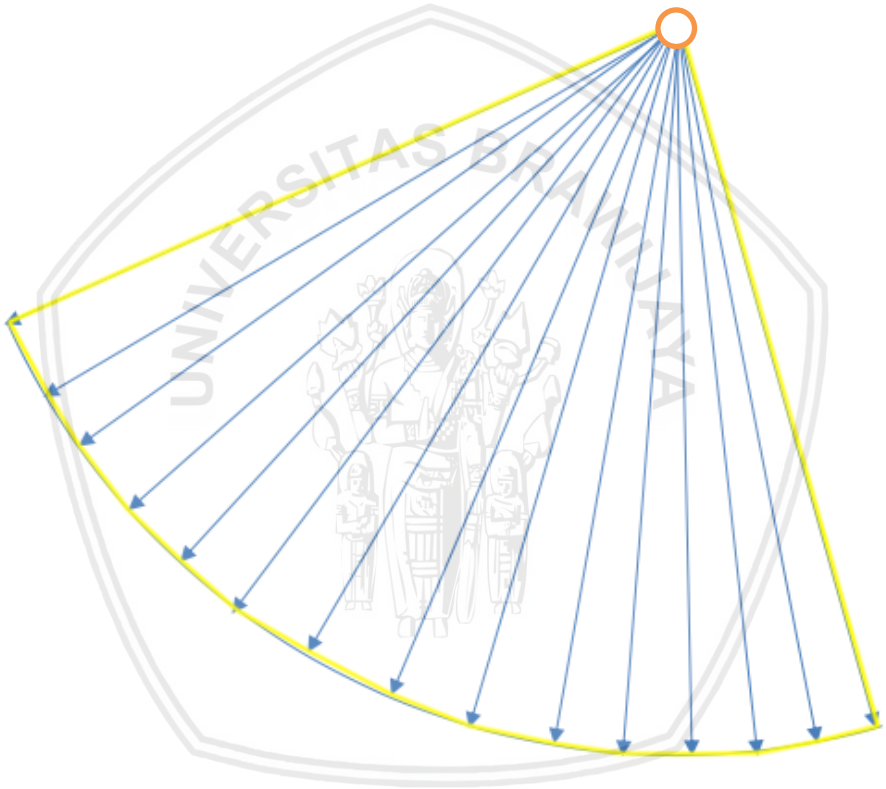
Tabel 4.2 menjelaskan mengenai algoritme tiap unit *raycast* yang ditembakan pada pada modul utama. Tujuan utamanya adalah menentukan titik jarak cahaya *raycast* yang ditembakan. Apabila *raycast* mengenai *collider* maka nilai yang disimpan adalah jarak dari sumber cahaya ke titik terkenanya *collider* sedangkan apabila *raycast* tidak mengenai *collider* maka jarak *raycast* adalah nilai maksimum radius cahaya yang telah ditentukan dari modul utama. Setelah nilai jarak *raycast* didapatkan maka fungsi ini akan mengembalikan nilai *raycast* kepada modul utama untuk kemudian nilai tersebut diproses agar dapat menjadi titik referensi untuk membentuk *polygon*.

4.4 Perancangan *Polygon Dynamic Lighting*

Pada tahap ini akan dijelaskan mengenai perancangan modul *dynamic lighting* berdasarkan perancangan algoritme pada subbab 4.2 dan 4.3. Setelah algoritme dirancang maka akan dilakukan perancangan modul menggunakan *raycasting*. Modul *dynamic lighting* yang nantinya akan menjadi objek dapat menghasilkan bentuk *polygon* yang dibentuk dari titik yang didapatkan dari hasil raycasting.

4.4.1 Visualisasi *Polygon* Apabila *Raycast* Tidak Mengenai *Collider*

Pada bagian ini akan dijelaskan bagaimana hasil visibility polygon dari modul *dynamic lighting* yang dibuat apabila *raycast* tidak mengenai *collider*.

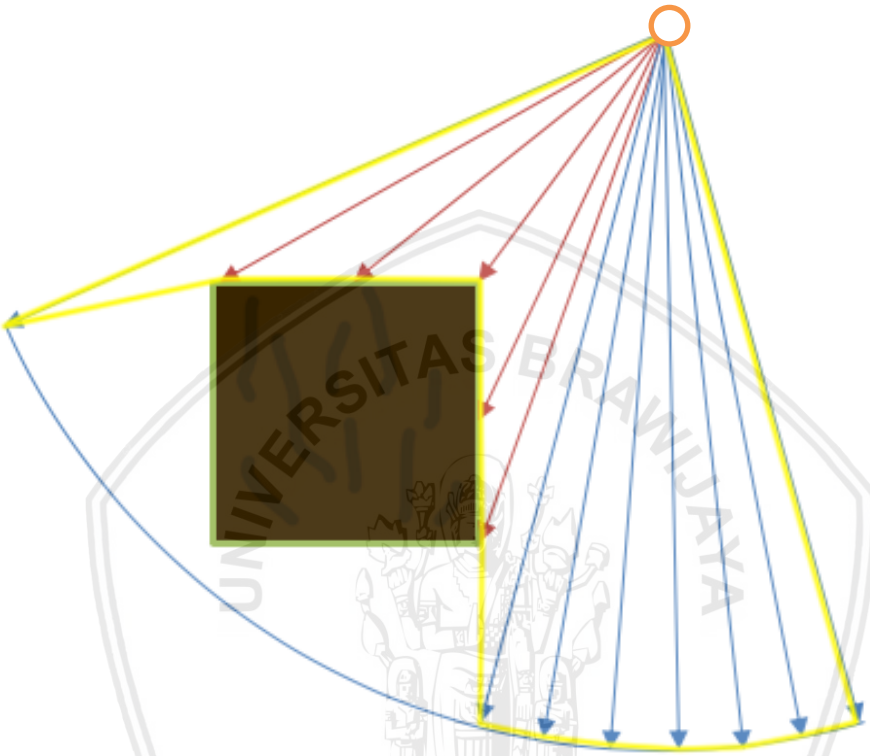


Gambar 4.3 Hasil *Polygon* Dari *Raycast* Yang Tidak Mengenai *Collider*

Pada Gambar 4.3 dapat dilihat rancangan modul yang mana sumber *raycast* adalah lingkaran oranye, dari sumber tersebut memancarkan raycast dengan radius tertentu. *Raycast* digambarkan dengan panah berwarna biru yang memancar dari sumber menuju jarak tertentu. Kemudian *polygon* digambarkan dengan *highlight* berwarna kuning yang menjadi *visibility polygon* dari titik sumber. *Polygon* ini lah yang menjadi *dynamic lighting* yang nantinya diimplementasikan.

4.4.2 Visualisasi *Polygon* Apabila Mengenai *Collider*

Pada bagian ini akan dijelaskan bagaimana hasil visibility polygon dari modul dynamic lighting yang dibuat apabila raycast mengenai collider yang terdapat pada object yang memiliki collider dari rancangan pada subbab 4.1.



Gambar 4.4 Hasil *Polygon* Dari *Raycast* Yang Mengenai *Collider*

Pada Gambar 4.4 dapat dilihat rancangan modul yang mana sumber *raycast* adalah lingkaran oranye, dari sumber tersebut memancarkan raycast dengan radius tertentu. *Raycast* yang tidak mengenai *collider* digambarkan dengan panah berwarna biru yang memancar dari sumber menuju jarak tertentu, sedangkan *raycast* yang mengenai *collider* digambarkan dengan anak panah merah. Kemudian *polygon* digambarkan dengan *highlight* berwarna kuning yang menjadi *visibility polygon* dari titik sumber. *Polygon* ini lah yang menjadi *dynamic lighting* yang nantinya diimplementasikan.

4.5 Perancangan Konfigurasi Parameter Pengujian Modul *Dynamic Lighting*

Pada bagian ini akan dijelaskan bagaimana perancangan konfigurasi parameter modul dynamic lighting yang akan dibuat untuk dilakukan pengujian pada Bab 6.

4.5.1 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Modul

Pada bagian ini akan dijelaskan bagaimana perancangan konfigurasi parameter untuk pengujian kinerja berdasarkan jumlah modul.

Tabel 4.3 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Modul

Parameter	Nilai
Besar Sudut	90.0
Besar Radius	40.0
Jumlah Raycast	25

Tabel 4.3 menjelaskan mengenai perancangan konfigurasi parameter yang nantinya akan digunakan sebagai batasan saat melakukan pengujian modul yang akan dibuat. Karena pengujian dilakukan berdasarkan jumlah modul maka parameter modul *dynamic lighting* ditentukan dengan jumlah yang sama berdasarkan Tabel 4.3.

4.5.2 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Raycast

Pada bagian ini akan dijelaskan bagaimana perancangan konfigurasi parameter untuk pengujian kinerja berdasarkan jumlah raycast pada satu modul *dynamic lighting*.

Tabel 4.4 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Raycast

Parameter	Nilai
Besar Sudut	90.0
Besar Radius	40.0

Tabel 4.4 menjelaskan mengenai perancangan konfigurasi parameter yang nantinya akan digunakan sebagai batasan saat melakukan pengujian modul yang akan dibuat. Karena pengujian dilakukan berdasarkan jumlah raycast maka ada 2 parameter modul *dynamic lighting* ditentukan dengan jumlah yang sama berdasarkan Tabel 4.3 dan yang akan bervariasi adalah parameter jumlah raycast pada modul *dynamic lighting* yang akan dibuat.

BAB 5 IMPLEMENTASI

Pada Bab ini akan dijelaskan mengenai implementasi berdasarkan perancangan yang telah dilakukan pada Bab 4.

5.1 Implementasi *Collider* Pada *Game*

Bagian ini akan menjelaskan implementasi *collider* yang telah dirancang pada bab 4. Implementasi game seluruhnya adalah menggunakan engine *Unity* telah disediakan *collider* yang dapat diimplementasikan langsung pada objek yang diinginkan.

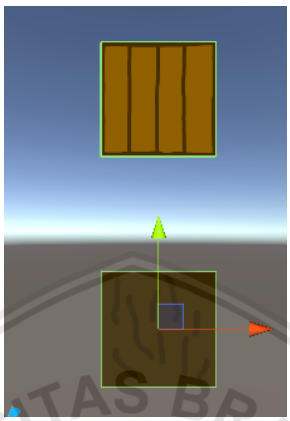
5.1.1 Implementasi *Collider Character Player*

Gambar 5.1 adalah implementasi *collider* pada karakter pemain yang nantinya objek tersebut dapat menerima *raycast* dari modul *dynamic lighting* yang dibuat. Sesuai dengan perancangan bentuk *collider* adalah persegi pada tubuh karakter pemain.

Gambar 5.1 Implementasi *Collider Character Player*

5.1.2 Implementasi *Collider Obstacle*

Gambar 5.2 adalah implementasi *collider* pada *platform* dan *box* yang nantinya ia dapat menerima *raycast* dari modul *dynamic lighting* yang dibuat. Sesuai dengan perancangan bentuk *collider* adalah persegi yang bentuknya sesuai dengan wujudnya.



Gambar 5.2 Implementasi *Collider Obstacle*

5.2 Implementasi Algoritme *Dynamic Lighting*

Pada Subbab ini akan dijelaskan mengenai implementasi algoritme *dynamic lighting* yang telah dirancang pada bab 4. Rancangan algoritme diimplementasikan dengan Bahasa *C#* dan menggunakan *library* yang ada pada *engine Unity*. Tabel 5.1 adalah implementasi kode *dynamic lighting* secara umum yang telah dirancang pada bab 4.

Tabel 5.1 Kode Implementasi Algoritme *Dynamic Lighting*

Implementasi Algoritme Dynamic Lighting

```
void DrawRaycast(){
   int rayCount = meshResolution;
   float rayAngleSize = lightAngle / (rayCount);
   List<Vector3> rayPoints = new List<Vector3>();


   for (int i = 0; i <= rayCount; i++){
      float angle = transform.eulerAngles.z - lightAngle / 2 + rayAngleSize * i;
       Debug.DrawLine(transform.position, transform.position + DirFromAngle(angle, true) * lightRadius, Color.red);
       RayCastInfo newRayCastInfo = RayCasting(angle);
       rayPoints.Add(newRayCastInfo.point);
       }
```

```
        int vertexCount = rayPoints.Count+1;
        Vector3[] vertices = new Vector3[vertexCount+1];
        int[] triangles = new int[(vertexCount - 2) * 3];
        vertices[0] = Vector3.zero;
        for (int i = 0; i < vertexCount-1; i++){
        vertices[i+1]=transform.InverseTransformPoint(rayPoints[i])
            if (i < vertexCount - 2){
                triangles[i * 3] = 0;
                triangles[i * 3 + 1] = i + 1;
                triangles[i * 3 + 2] = i + 2;
            }
        }
        viewMesh.Clear();
        viewMesh.vertices = vertices;
        viewMesh.triangles = triangles;
        viewMesh.RecalculateNormals();
    }
```

Dari kode yang telah diimplementasikan pada Tabel 5.1 dapat dihasilkan suatu modul *dynamic lighting* pada Gambar 5.3 yang pada gambar tersebut dapat dilihat *ray*/cahaya yang dipancarkan dari titik sumber yang direpresentasikan dengan garis *debug* berwarna merah. Dari algoritme yang diimplementasikan polygon dapat terbentuk dari informasi *raycast* yang berasal dari sumber cahaya. Apabila cahaya mengenai karakter atau obstacle maka raycast akan menyimpan titik/koordinat dimana terjadi *collision* dengan dengan *collider.*

Gambar 5.3 Implementasi *Dynamic Lighting* Pada *Game*

5.3 Implementasi Algoritme *Raycasting*

Pada subbab ini akan dijelaskan mengenai implementasi algoritme *raycasting* yang telah dirancang pada bab 4. Potongan kode yang terdapat pada Tabel 5.2 adalah fungsi yang dipanggil pada perulangan pada Tabel 5.1, yang mana perulangan tersebut melakukan komputasi *raycastin*g pada tiap perulangannya.

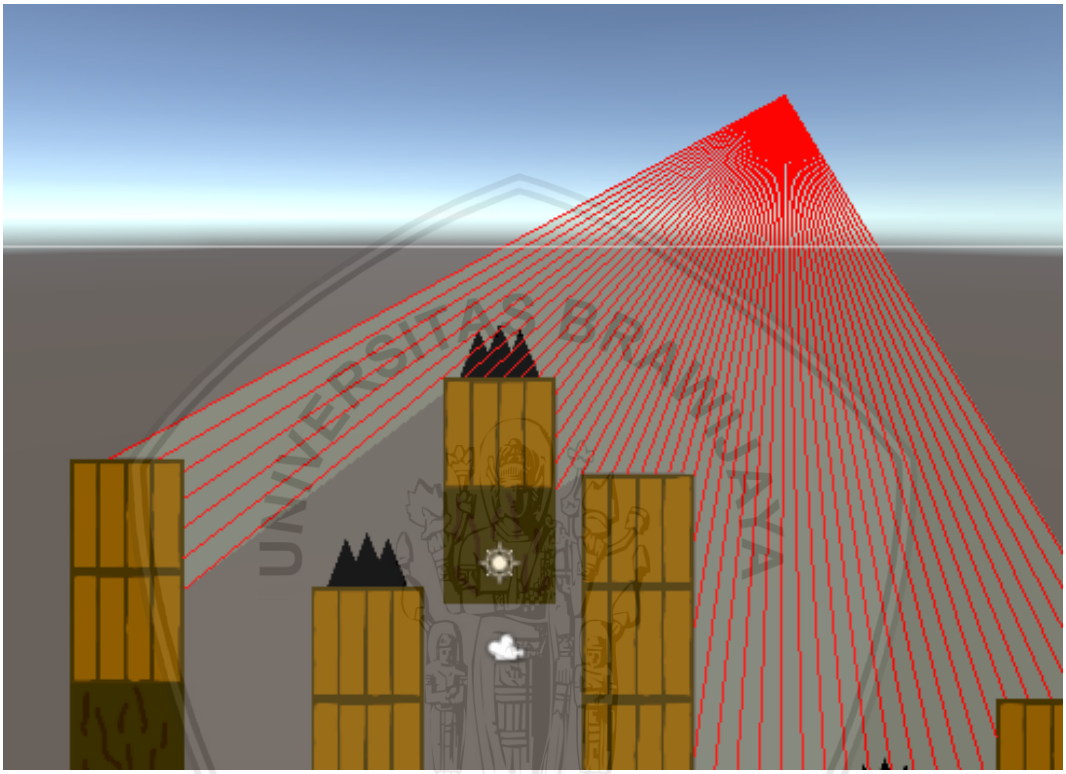
Tabel 5.2 Implementasi Algoritme *Raycasting*

Implementasi Algoritme Raycasting

`RayCastInfo RayCasting(float globalAngle){`
`Vector3 direction = DirFromAngle(globalAngle, true);`
`RaycastHit hit;`
`if(Physics.Raycast(transform.position,direction,out hit,lightRadius,obstacleMask)){`
`print("hit!");`
`return new RayCastInfo(true, hit.point, hit.distance, globalAngle);`
`}`
`else{`
`print("not Hit");`
`return new RayCastInfo(false, transform.position + direction * lightRadius, lightRadius, globalAngle);`
`}`
`}`

Potongan kode pada Tabel 5.2 berfungsi menyimpan informasi *raycast* yang dilakukan modul utama yang mana apabila *raycast* mengenai *collider* akan mengembalikan nilai jarak saat terjadi *collision*. Apabila *raycast* tidak mengenai *collider* / tidak terjadi *collision* maka fungsi ini akan mengembalikan jarak maksimum sesuai jarak maksimal yang ditentukan pada parameter modul saat dijalankan.

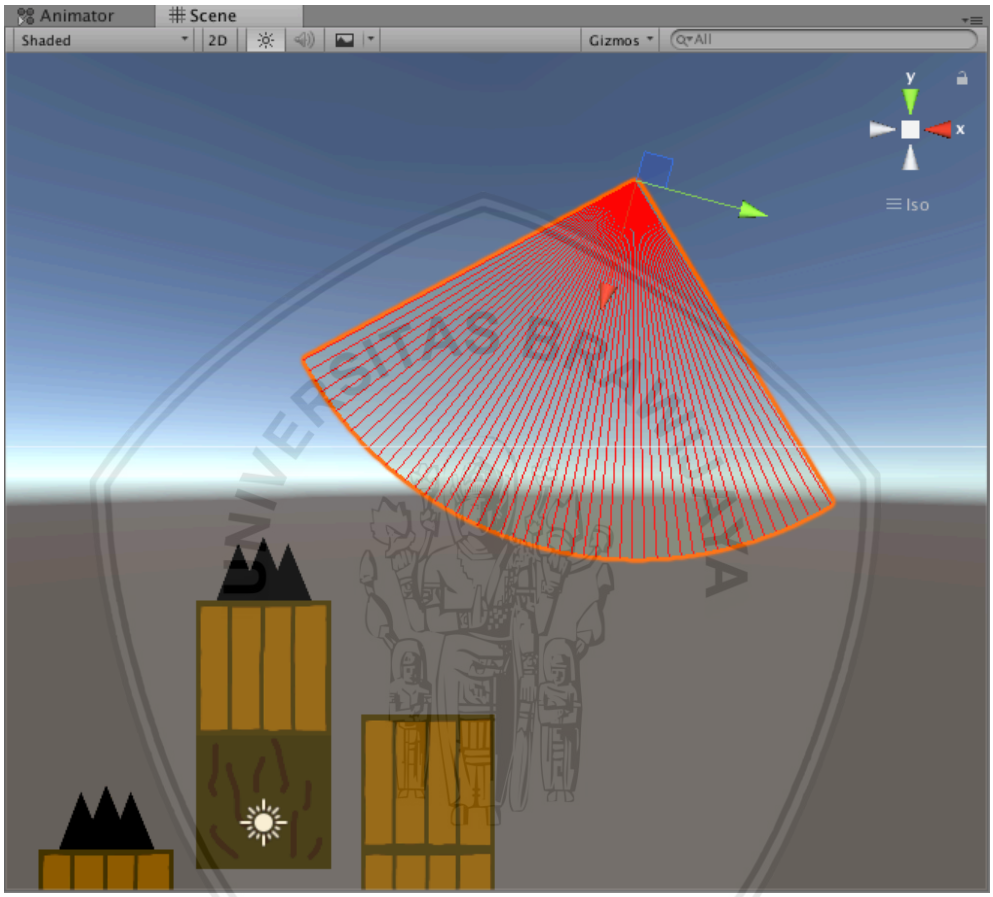
Pada Gambar 5.4 terdapat garis debug berwarna merah yang mana garis terebut merepresentasikan *ray*/cahaya yang ditembakan dari sumbernya sedangkan mesh filter berwarna kekuningan adalah polygon yang terbentuk dari titik *raycast* yang dilakukan fungsi *raycasting.* Pada saat ray melewati suatu objek yang telah diberi *collider* maka titik *raycast* akan berhenti pada titik tersebut sehingga seolah seperti cahaya yang membentuk bayangan.



Gambar 5.4 Visualisasi Algoritme *Raycasting*

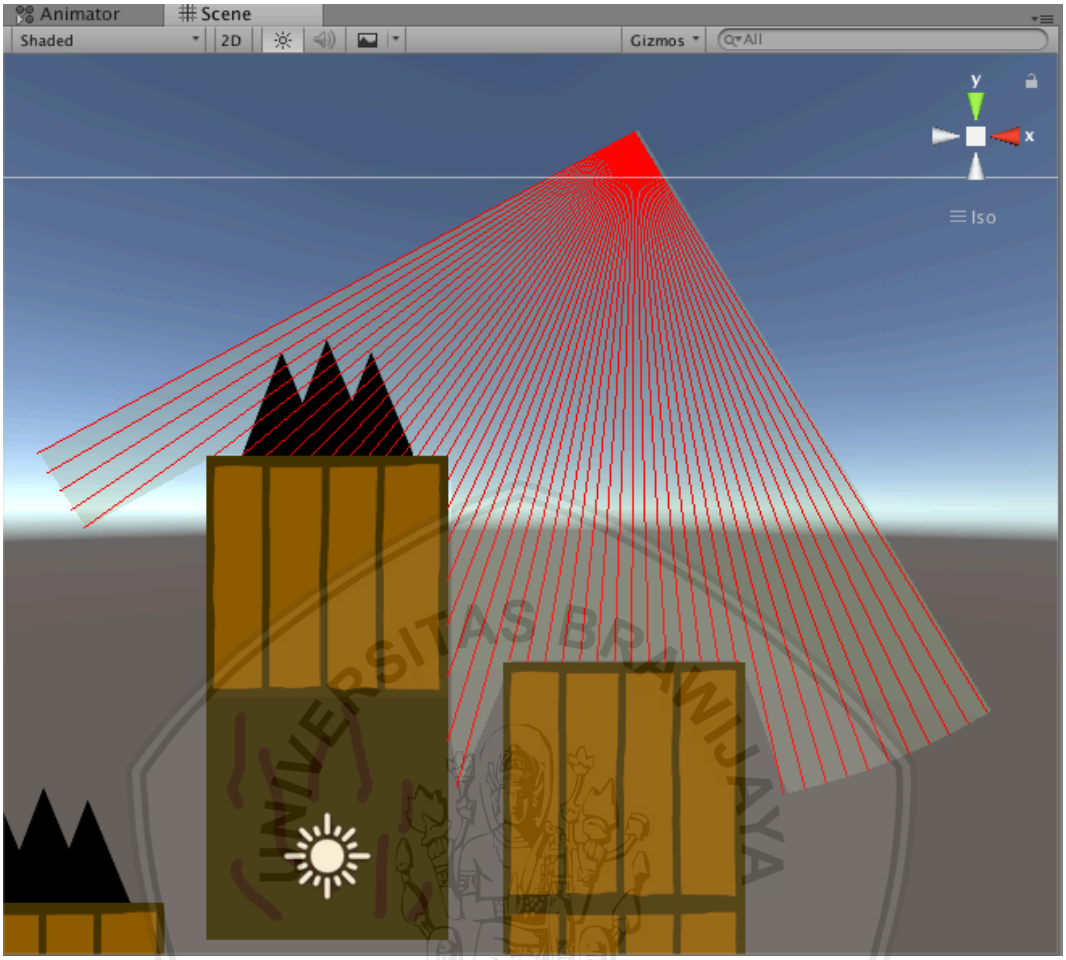
5.4 Implementasi *Polygon Dynamic Lighting*

Pada subbab ini akan dijelaskan hasil implementasi dari modul sesuai dengan perancangan pada subbab 4.4 yang merancang bagaimana bentuk polygon dari hasil *raycast* apabila mengenai raycast mengenai *collider* atau tidak mengenai *collider*.



Gambar 5.5 Implementasi *Polygon* Apabila *Raycast* Tidak Mengenai *Collider*

Pada Gambar 5.5 dapat dilihat hasil implementasi sesuai rancangan pada subbab 4.4.1 dimana raycast tidak mengenai collider sehingga titik raycast berhenti sesuai pada jarak maksimum sesuai dengan jarak radius yang ditentukan sehingga bentuk polygon yang dihasilkan adalah dari titik raycast yang berhenti pada jarak maksimum.



Gambar 5.6 Implementasi *Polygon* Apabila *Raycast* Mengenai *Collider*

Pada Gambar 5.6 dapat dilihat hasil implementasi sesuai rancangan pada subbab 4.4.2 dimana raycast mengenai collider yang ada pada *obstacle* sehingga titik raycast berhenti pada collider apabila terjadi *collision* dan apabila *raycast* tidak mengenai *collider* maka jarak *raycast* akan berhenti sesuai dengan jarak radius yang ditentukan sehingga titik berhentinya *raycast* inilah yang menjadi titik untuk membentuk *polygon dynamic lighting*.

BAB 6 PENGUJIAN DAN ANALISIS

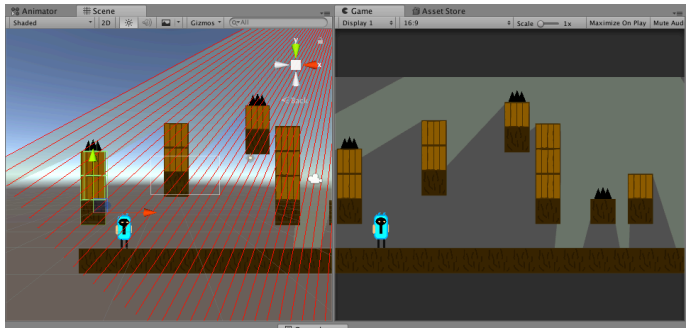
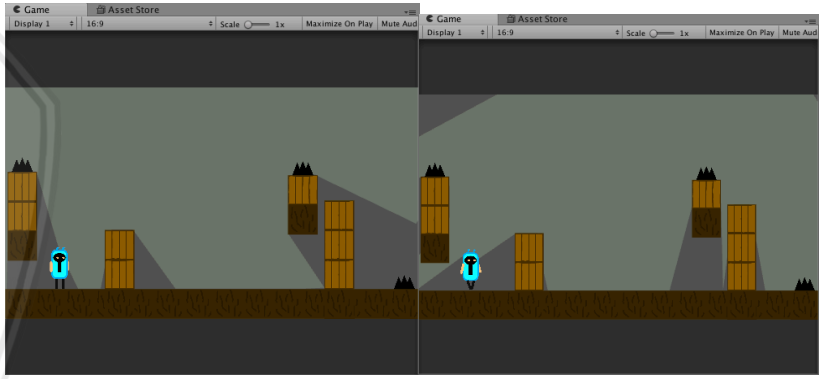
Pada bab ini akan dibahas mengenai pengujian modul yang telah diimplementasikan pada game dibuat. Pengujian yang dilakukan adalah pengujian validitas untuk mengetahui apakah fungsional dari modul yang dibuat telah valid, Kemudian dilakukan pengujian kinerja berdasarkan jumlah modul yang diimplementasikan pada suatu scene dengan spesifikasi yang sama dan juga dilakukan pengujian kinerja suatu modul dengan variabel jumlah resolusi *raycast* yang berbeda.

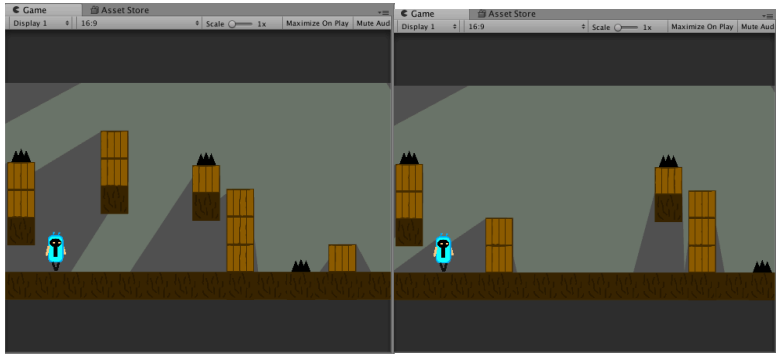
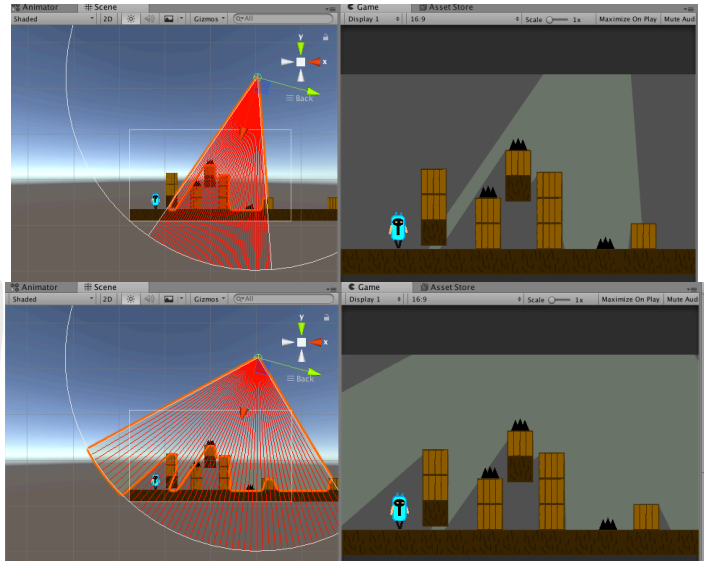
6.1 Pengujian Fungsional Modul

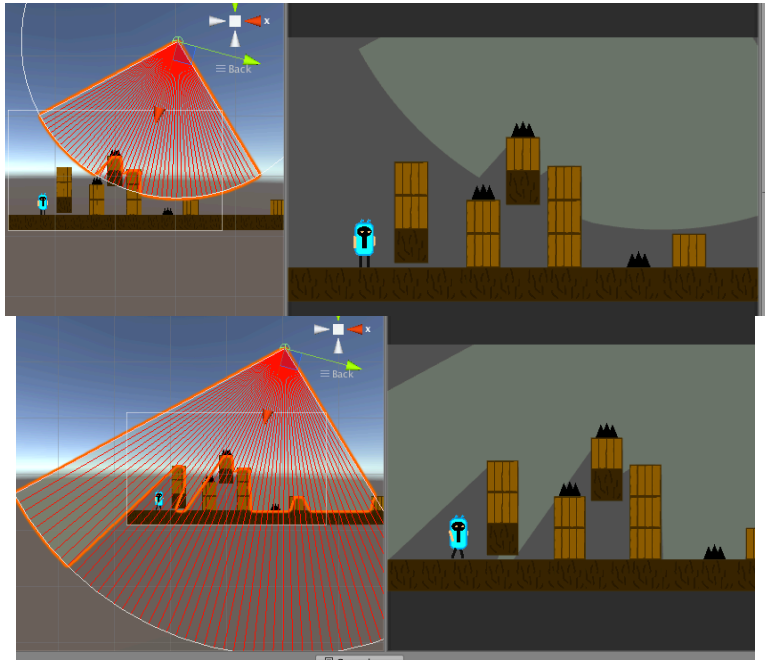
Pengujian fungsional dilakukan dengan tujuan apakah modul telah berfungsi sesuai ekspetasi. Pengujian akan dilakukan dengan metode black-box dengan parameter sebagai berikut.

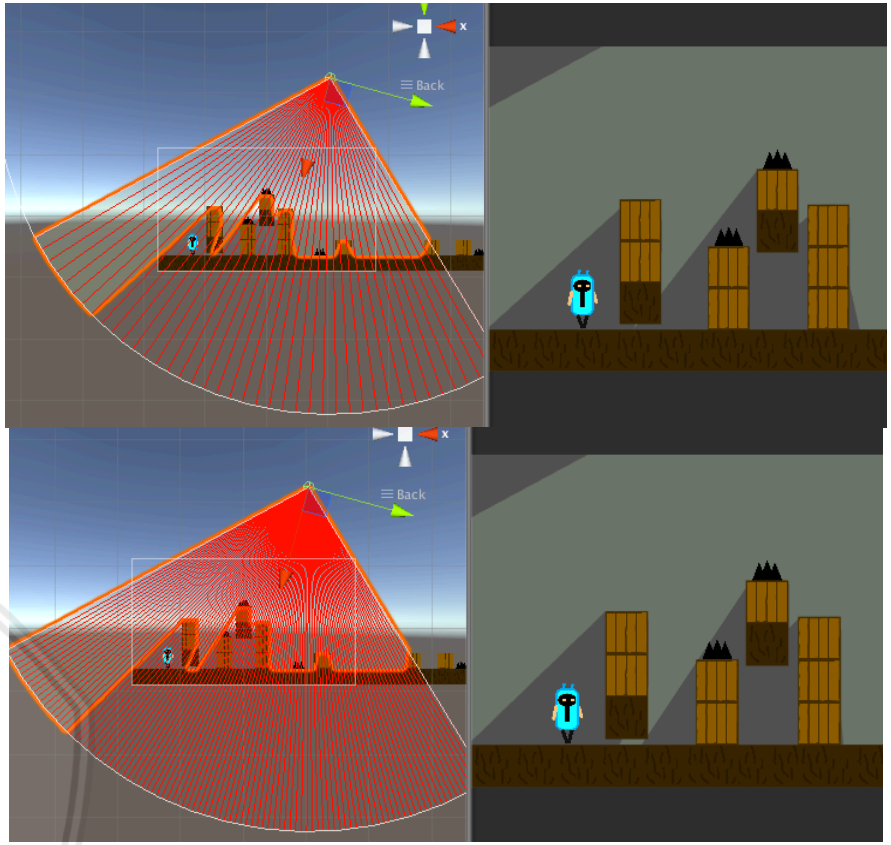
1. Modul *dynamic lighting* dapat membentuk bayangan sesuai *obstacle* yang menghalangi
2. Bentuk *polygon* cahaya dapat berubah apabila objek *dynamic lighting* berubah posisi
3. Bentuk *polygon* cahaya dapat berubah apabila *obstacle* berubah posisi
4. Bentuk *polygon* cahaya membentuk sudut sesuai nilai yang dimasukan
5. Bentuk *polygon* cahaya membentuk radius sesuai nilai yang dimasukan
6. Banyaknya cahaya/*raycast* sesuai nilai yang dimasukan

Tabel 6.1 Tabel Pengujian Fungsional

Tabel Pengujian Fungsional				
No	**Parameter**	**Hasil yang Diharapkan**	**Hasil**	**Status**
1	Modul *dynamic lighting* dapat membentuk bayangan sesuai *obstacle* yang menghalangi	Modul *dynamic lighting* membentuk polygon sesuai *collider* objek	Modul *dynamic lighting* membentuk mesh polygon sesuai berdasarkan objek yang menghalangi	Valid 
2	Bentuk polygon cahaya dapat berubah apabila objek *dynamic lighting* berubah posisi	Bentuk polygon cahaya dapat berubah apabila objek *dynamic lighting* berubah posisi	Bentuk mesh polygon cahaya berubah apabila objek *dynamic lighting* berubah posisi	Valid 

3	Bentuk polygon cahaya dapat berubah apabila *obstacle* berubah posisi	Bentuk polygon cahaya dapat berubah apabila *obstacle* berubah posisi	Bentuk polygon cahaya dapat berubah apabila *obstacle* berubah posisi	Valid 
4	Bentuk polygon cahaya membentuk sudut sesuai nilai yang dimasukan	Bentuk polygon cahaya membentuk sesuai nilai yang dimasukan	Bentuk polygon cahaya membentuk sudut sesuai nilai yang dimasukan	Valid 

5	Bentuk polygon cahaya membentuk radius sesuai nilai yang dimasukan	Bentuk polygon cahaya membentuk radius sesuai nilai yang dimasukan	Bentuk polygon cahaya membentuk radius sesuai nilai yang dimasukan	Valid 

6	Banyaknya cahaya/*raycast* sesuai nilai yang dimasukan	Banyaknya cahaya/*raycast* sesuai nilai yang dimasukan	Banyaknya jumlah cahaya/*raycast* yang dipancarkan sesuai jumlah nilai yang dimasukan	Valid 

Pada Tabel 6.1 telah dijabarkan mengenai pengujian fungsional yang dilakukan pada modul yang telah diimplementasikan. Dari 6 parameter pengujian yang diuji didapatkan hasil 100% valid.

Pada Tabel 6.1 Nomor 1 adalah hasil pengujian parameter uji fungsional yang pertama dan didapatkan hasil valid karena bentuk *dynamic lighting* sesuai dengan *collider* yang diimplementasikan pada objek yang dirancang pada bab 4. Hal ini disebabkan karena bentuk polygon *dynamic lighting* sesuai dengan collider yang ada pada saat pengujian.

Pada Tabel 6.1 Nomor 2 adalah hasil pengujian parameter uji fungsional yang ke-dua dan didapatkan hasil valid karena bentuk polygon *dynamic lighting* dapat berubah sesuai dengan posisi objek yang mana pada kolom status dapat dilihat posisi objek yang sebelumnya berada di sebelah kiri kemudian dipindahkan ke sebelah kanan menghasilkan bentuk polygon yang berbeda.

Pada Tabel 6.1 Nomor 3 adalah hasil pengujian parameter uji fungsional yang ke-tiga dan didapatkan hasil valid karena bentuk *polygon dynamic lighting* dapat berubah apabila posisi *obstacle* berubah/berpindah yang mana pada kolom status dapat dilihat pada gambar kiri sebelumnya *obstacle* berada dibawah dan pada gambar kanan *obstacle* tersebut berpindah sedikit keatas dan didapati bentuk *polygon dynamic lighting* yang berubah.

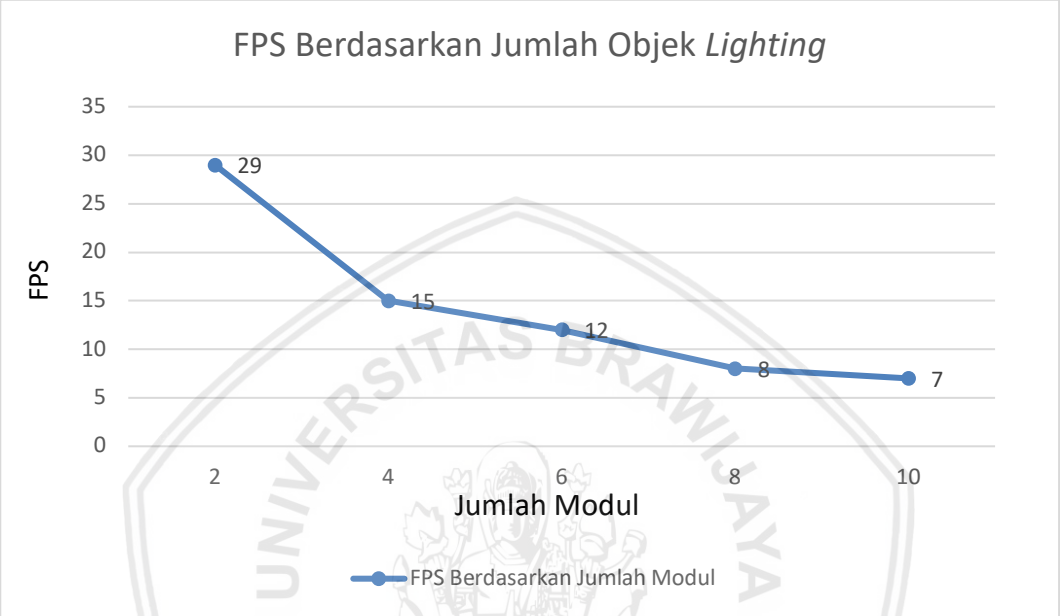
Pada Tabel 6.1 Nomor 4 adalah hasil pengujian parameter uji fungsional yang ke-empat dan didapatkan hasil valid karena bentuk *dynamic lighting* dapat berubah sesuai dengan sudut yang ditentukan pada modul. Pada kolom status dapat dilihat gambar atas adalah modul dengan nilai sudut yang lebih kecil daripada nilai sudut yang diberikan pada gambar bagian bawah, dari kedua gambar tersebut didapati bentuk *polygon* yang berubah sesuai sudut yang ditentukan.

Pada Tabel 6.1 Nomor 5 adalah hasil pengujian parameter uji fungsional yang ke-lima dan didapatkan hasil valid karena bentuk *dynamic lighting* dapat berubah sesuai dengan radius yang ditentukan pada modul yang mana pada kolom status dapat dilihat pada gambar bagian atas radius yang diberikan adalah lebih kecil daripada nilai radius yang diberikan pada gambar bagian bawah, dari kedua gambar tersebut didapati bentuk *polygon* yang berubah sesuai dengan nilai radius yang diberikan

Pada Tabel 6.1 Nomor 6 adalah hasil pengujian parameter uji fungsional yang ke-6 dan didapatkan hasil valid karena garis *debug* yang merepresentasikan *raycast* pada modul berjumlah sesuai dengan nilai resolusi cahaya yang ditentukan pada parameter pada modul, pada kolom status dapat dilihat pada gambar bagian atas garis raycast berjumlah lebih sedikit dari pada garis *raycast* pada gambar bagian bawah, dari gambar tersebut didapati bentuk *polygon* juga berubah sesuai dengan jumlah raycast yang diberikan pada parameter modul *dynamic lighting*.

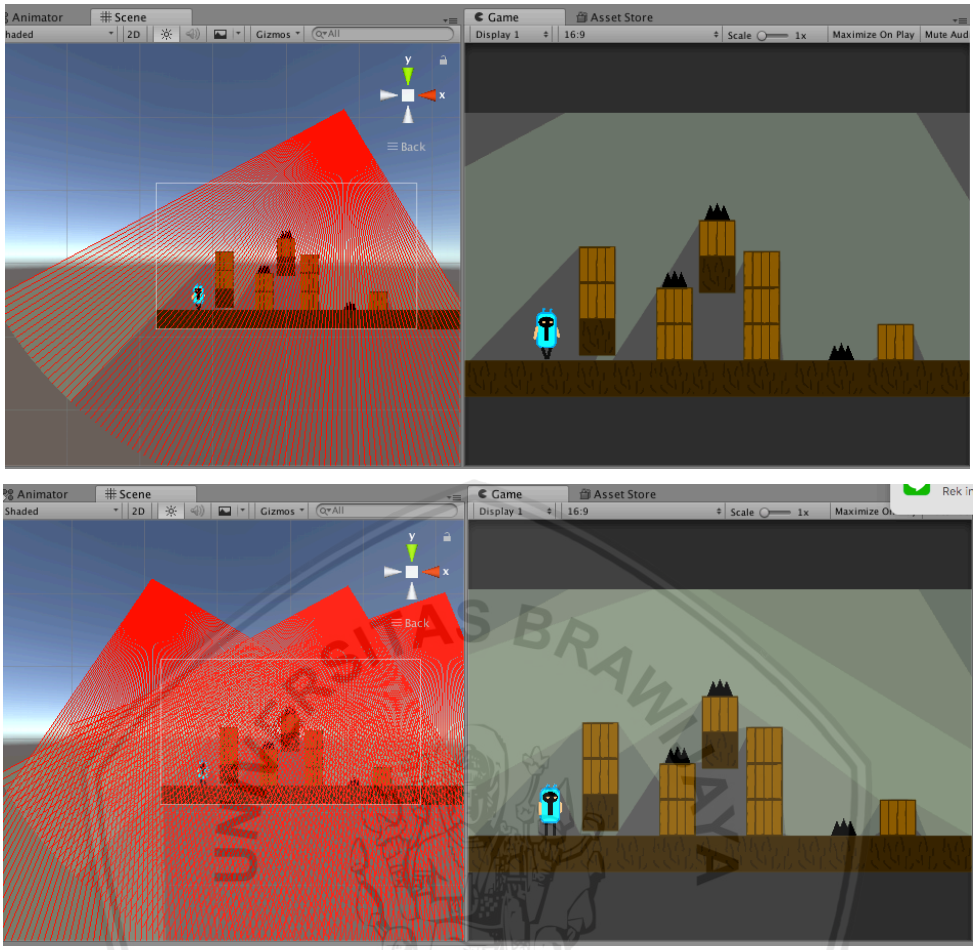
6.2 Pengujian Kinerja Modul

Pada subbab ini akan dilakukan pengujian kinerja dengan dua kasus uji yang berbeda yaitu uji *fps* berdasarkan jumlah modul yang diimplementasikan dan yang kedua adalah pengujian *fps* berdasarkan resolusi cahaya pada satu modul yang diimplementasikan.



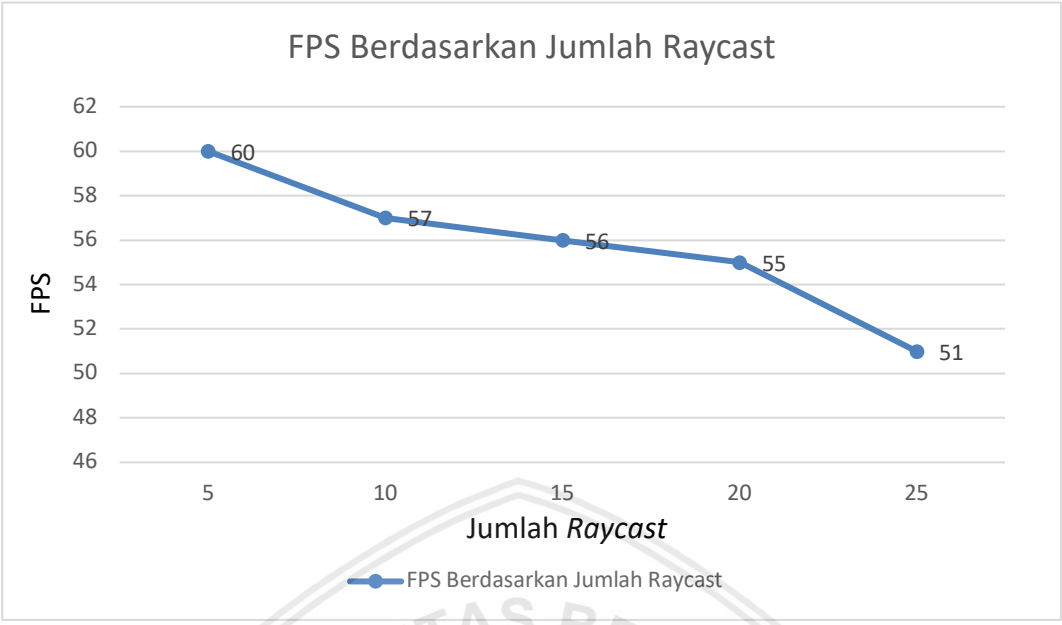
Gambar 6.1 Grafik FPS Berdasarkan Jumlah Modul

Gambar 6.1 adalah hasil uji kinerja modul berdasarkan jumlah modul *dynamic lighting* yang di-*instance* pada game. Parameter modul yang diimplementasikan memiliki radius, resolusi dan sudut yang sama pada tiap modul. Pengujian dilakukan dengan melakukan *clone* pada modul *dynamic lighting* dengan parameter yang sama. Dari pengujian yang dilakukan didapatkan hasil yaitu penurunan *FPS* ketika jumlah modul bertambah. Dikarenakan batas aman FPS untuk dapat dinikmati user adalah 24 FPS (Salmon, Armstrong dan Jolly, 2011) maka berdasarkan dari hasil pengujian disarankan untuk tidak mengimplementasikan lebih dari 2 objek *dynamic lighting* karena dapat menurunkan FPS dibawah 24 dan mempengaruhi imersifitas user dalam menikmati permainan yang diimplementasikan modul ini.



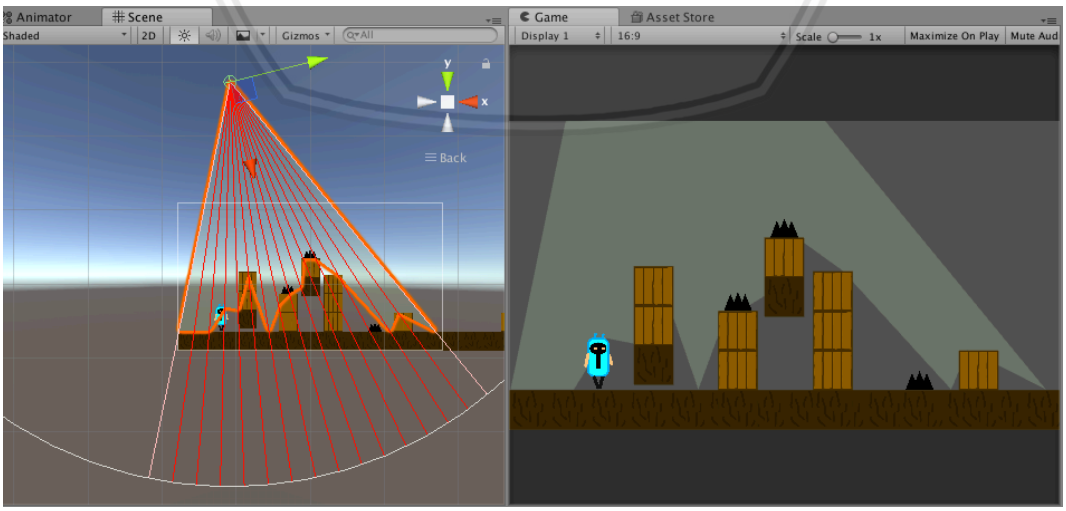
Gambar 6.2 *Screenshoot* Pengujian *FPS* Berdasarkan Jumlah Modul

Pada Gambar 6.2 dapat dilihat yaitu pengujian dengan menambahkan jumlah modul yang diimplementasikan pada gambar bagian atas terlihat 2 modul yang diimplementasikan secara bersamaan dan pada bagian bawah dapat terlihat 3 modul yang diimplementasikan secara bersamaan dengan konfigurasi parameter yang sesuai dengan perancangan yang dilakukan pada subbab 4.5.1.



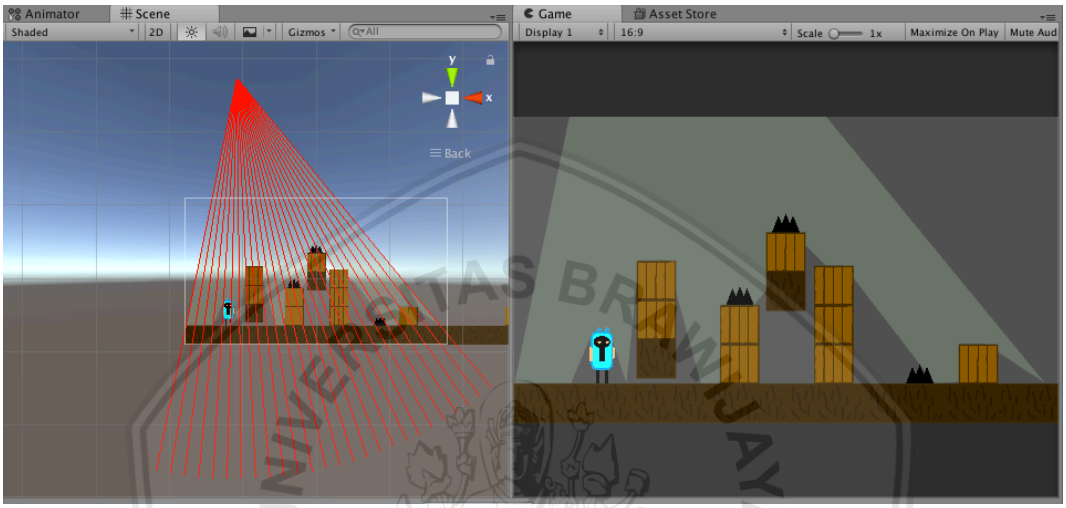
Gambar 6.3 Grafik FPS Berdasarkan Jumlah *Raycast*

Gambar 6.3 adalah hasil uji kinerja modul berdasarkan resolusi *raycast* pada modul *dynamic lighting* yang di instance pada game. Hanya 1 modul yang diujikan dengan merubah *value* dari resolusi pada parameter modul. Pengujian dilakukan dengan menambahkan value berdasarkan perancangan pengujian sebelumnya. Didapatkan hasil yaitu penurunan FPS ketika jumlah resolusi *raycast* bertambah namun penurunan FPS yang terjadi masih dapat ditoleransi karena masih diatas batas 24 FPS yang mana apabila jumlah *raycast* 25 masih mendapatkan FPS diatas 24 yang tidak signifikan mempengaruhi *playability* game.



Gambar 6.4 Pengujian Resolusi *Raycast* Dengan Nilai 15

Pada Gambar 6.4 dan Gambar 6.5 adalah *screenshoot* pengujian dengan mengubah nilai resolusi *raycast* pada parameter modul dengan konfigurasi parameter yang sesuai dengan perancangan yang dilakukan pada subbab 4.5.2. Perbedaan yang terlihat adalah apabila resolusi modul semakin kecil maka bentuk *polygon dynamic lighting* menjadi tidak akurat dan FPS akan meningkat. Apabila resolusi semakin besar maka *bentuk polygon dynamic lighting* semakin akurat dan FPS akan menurun.



Gambar 6.5 Pengujian Resolusi *Raycast* Dengan Nilai 25

BAB 7 KESIMPULAN

7.1 Kesimpulan

Penelitian tentang *dynamic lighting* pada *2d endless runner game* menggunakan *visibility polygon computation* telah berhasil dilakukan sesuai dengan metode yang telah ditentukan sebelumnya. Hasil dari penelitian ini dapat disimpulkan sebagai berikut.

1. Implementasi *dynamic lighting* menggunakan *visibility polygon* dapat dilakukan dengan menggunakan *raycasting*, yang mana *visibility polygon* dapat terbentuk menggunakan *raycasting* dan hasil titik *raycast* pada *raycasting* tersebut dapat digunakan untuk membentuk *polygon* yang menjadi *dynamic lighting* pada game berlingkungan 2D.
2. Dari pengujian fungsional didapatkan hasil 100% valid yang mana modul yang dibuat dapat diimplementasikan dan berfungsi pada game 2D endless runner yang dijadikan bahan penelitian skripsi ini. Dan dari hasil pengujian *FPS* yang telah dilakukan dapat disimpulkan bahwa semakin banyak modul dan *raycast* yang diimplementasikan/dikomputasi karena meng-instance object pada game didapati FPS yang menurun, sehingga dapat disimpulkan bahwa semakin banyak *raycast* maka akan mempengaruhi *FPS* pada game tersebut. Namun pada kasus pengujian kinerja pertama dapat disimpulkan apabila modul yang diimplementasikan berjumlah 2 objek dengan parameter yang sama didapatkan FPS yang masih diatas batas 24 yang mana hasil tersebut adalah masih dianggap *playable,* sehingga object *lighting* yang diimplementasikan dapat dibatasi menjadi 2 objek saja agar tidak menurunkan FPS dibawah 24, sedangkan pada pengujian kinerja kedua implementasi satu modul dengan jumlah raycast sampai dengan 25 yang mana pada tidak menurunkan FPS sampai pada batas minimum 24, sehingga apabila diimplementasikan satu modul dengan jumlah raycast tersebut masih dapat ditoleransi dalam hal kinerja FPS.

7.2 Saran

Berdasarkan pada permasalahan yang diangkat oleh penulis yaitu mengenai penerapan *dynamic lighting* pada *2D endless runner game* menggunakan *visibility polygon computation*, maka dari itu penulis memberikan saran sebagai berikut :

1. Penggunaan *raycasting* pada *visibility polygon* untuk *dynamic lighting* dapat lebih efisien apabila *raycast* ditembakan dikurangi menjadi hanya ditembakan pada sudut *collider* untuk menghemat komputasi *raycast* sehingga tidak terjadi penurunan kinerja yang signifikan karena semakin banyaknya *raycast* yang dipancarkan akan menambah waktu komputasi yang menyebabkan turunnya kinerja/*FPS* dari game dibuat.

2. Penggunaan *visibility polygon dengan raycasting* dapat juga digunakan untuk modul *field-of-view/detection* suatu *npc* pada *video game,* dengan memanfaatkan kemampuan deteksi suatu *raycast* saat mengenai *collider* objek pada *game*.

DAFTAR PUSTAKA

Bhosale, T., Kulkarni, S. dan Patankar, S.N., 2018. 2D PLATFORMER GAME IN UNITY ENGINE. *International Research Journal of Engineering and Technology*. [daring] Tersedia pada: <www.irjet.net> [Diakses 28 Agu 2018].

Bungiu, F., 2014. Efficient Computation of Visibility Polygons. hal.3–6.

Cao, D., Paraschakis, D., Mihailescu, R.-C. dan Eriksson, J., 2014. *Game Design Patterns in Endless Mobile Minigames*. [daring] Tersedia pada: <https://muep.mau.se/bitstream/handle/2043/21375/DavidCao_GameDesignPatternsInEmm_2016.pdf?sequence=2> [Diakses 28 Agu 2018].

El-Nasr, M.S., 2005. Intelligent lighting for game environments. *Journal of Game Development*, 1(2), hal.17–50.

El-Nasr, M.S., Niedenthal, S., Knez, I., Almeida, P. dan Zupko, J., 2006. Dynamic lighting for tension in games. *Game Studies*, 7(1).

Federico Viticci, 2015. *Alto's Adventure Review: A Beautiful Descent – MacStories*. [daring] Tersedia pada: <https://www.macstories.net/reviews/altos-adventure-review-a-beautiful-descent/> [Diakses 31 Agu 2018].

Ghosh, S.K., 2007. *Visibility algorithms in the plane*. *Visibility Algorithms in the Plane*, .

Keo, M., 2017. Graphical Style in Video Games.

Konferencia, Š.V., 2013. ADVANCED LIGHTING IN 2D GRAPHICS.

Osipovic, K., 2017. Volumetric lighting implementations in games.

Rick Taylor, 2014. *5 Reasons Video Games Lead American Entertainment | HuffPost*. [daring] Tersedia pada: <https://www.huffingtonpost.com/rich-taylor/5-reasons-video-games-lea_b_5309230.html> [Diakses 28 Agu 2018].

Roth, S.D., 1982. Ray casting for modeling solids. *Computer Graphics and Image Processing*, [daring] 18(2), hal.109–144. Tersedia pada: <http://linkinghub.elsevier.com/retrieve/pii/0146664X82901691> [Diakses 14 Nov 2018].

Salmon, R.A., Armstrong, M.G. dan Jolly, S.J.E., 2011. Higher Frame Rates for More Immersive Video and Telvision. *BBC Research White Paper 209*, (October).

Sam Byford, 2014. *Pixel art games aren't retro, they're the future | The Verge*. [daring] Tersedia pada: <https://www.theverge.com/2014/7/3/5865849/pixel-art-is-here-to-stay> [Diakses 28 Agu 2018].

Schultz, C.P., Bryant, R. dan Langdell, T., 2005. *Game Testing All in One by Charles P. Schultz, Robert Bryant and Tim Langdell Course Technology © 2005 (520*.

William Usher, 2014. *75% Of Gamers Say That Graphics Do Matter When Purchasing A Game*. [daring] Tersedia pada: <https://www.cinemablend.com/games/75-Gamers-Say-Graphics-Do-Matter-Purchasing-Game-64659.html> [Diakses 28 Agu 2018].