Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 5, Mei 2019, hIm. 4155-4163 http://j-ptiik.ub.ac.id

Penerapan *Dynamic Lighting* pada *2D Endless Runner Game* menggunakan *Visibility Polygon Computation*

Arda Satata Fitriajie[1], Eriq Muhammad Adams Jonemaro[2], Tri Afirianto[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]ardasatata@gmail.com, [2]eriq.adams@ub.ac.id, [3]tri.afirianto@ub.ac.id

Abstrak

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

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*

1. PENDAHULUAN

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*.

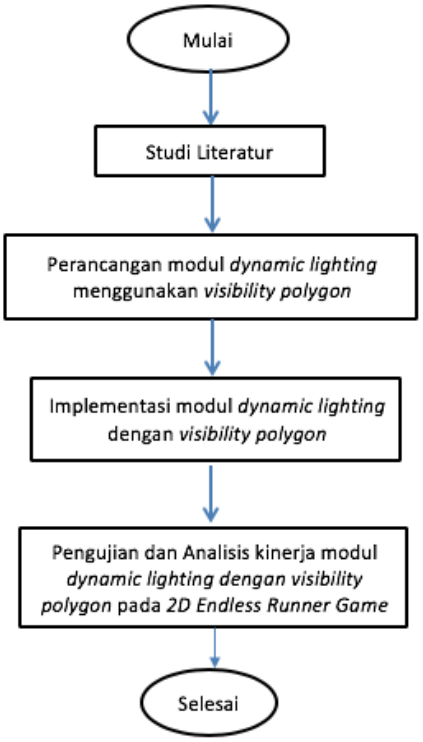
Visibility polygon adalah konsep dasar dari komputasi geometri yang menjelaskan bagaimana membentuk *polygon* yang menjadi daerah pada bidang 2D yang menjadi acuan apakah suatu titik pada bidang tersebut dapat terlihat dari titik yang diinginkan (Ghosh, 2007). Pada penelitian ini akan diimplementasikan *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 penelitian ini.

2. METODOLOGI

Pada bab ini akan menjelaskan langkah langah yang akan dilakukan dalam penyususnan penelitian. Penelitian ini bersifat implementatif dengan megimplementasikan *Dynamic Lighting* menggunakan *visibility polygon*. Metodologi penelitian yang digunakan dijelaskan pada Gambar 1.

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*



Gambar 1 Diagram Alir Metodologi

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 Tabel **1** dijelaskan parameter modul yang akan diimplementasikan menjadi modul.

Tabel 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

2.1 Perancangan

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

Karakter player akan menjadi salah satu objek yang dapat terkena raycasting dari modul yang dibuat. Gambar 2 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 2adalah persegi dengan garis berwarna hijau.

Gambar 2 Rancangan *Collider Character Player*

Objek *box* dan *platform* berbentuk persegi pada game akan diimplementasikan juga *collider* berbentuk persegi sesuai dengan bentuknya. Gambar 3 adalah rancangan *collider* yang akan diimplementasikan pada objek *platform* dan *wall* yang mana *collider* ditunjukan dengan persegi dengan garis berwarna hijau.

Gambar 3 Rancangan *Collider Obstacle*

Tabel 2 Pseudocode Modul Dynamic Lighting Secara Umum

No	Pseudocode Modul Dynamic Lighting
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
	Selesai

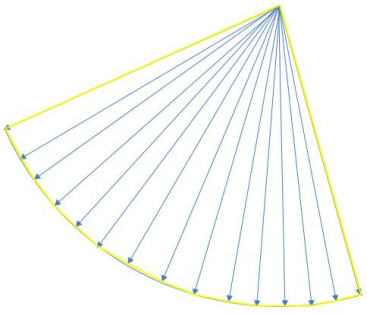
Penjelasan source code pada Tabel 2 Secara umum tahap-tahap yang dilakukan adalah pertama modul menerima input parameter yang terdiri dari radius, sudut, dan resolusi. Radius pada modul ini adalah seberapa besar jarak tembak setiap unit *raycast*-nya. Kemudian sudut adalah besarnya sudut tembak cahaya dengan

besaran derajat dari 0 sampai 360. Resolusi pada modul ini yaitu seberapa banyak cahaya/*raycast* yang ditembakan dari sumber cahaya.

Tabel 3 Pseudocode Algoritme Raycasting

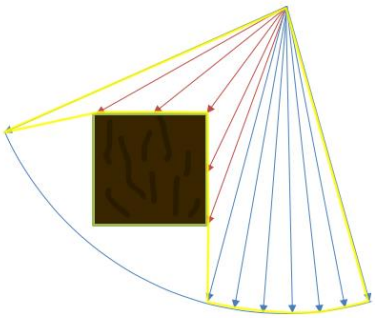
No	Pseudocode Algoritme Raycasting
1	Memasukan nilai arah cahaya (ray) dan Radius Cahaya
2	If raycast mengenai collider
3	Jarak Raycast = Titik Terkena Collider
4	Else
5	Jarak Raycast = Jarak Maksimum Radius Cahaya
6	Kembalikan nilai jarak raycast

Tabel 3 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.



Gambar 4 Hasil Polygon Dari Raycast Yang Tidak Mengenai Collider

Pada Gambar 4 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.



Gambar 5 Hasil Polygon Dari Raycast Yang Mengenai Collider

Pada Gambar 5 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.

Tabel 4 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Modul

Parameter	Nilai
Besar Sudut	90.0
Besar Radius	40.0
Jumlah Raycast	25

Tabel 4 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.

Tabel 5 Konfigurasi Parameter Pengujian FPS Berdasarkan Jumlah Raycast

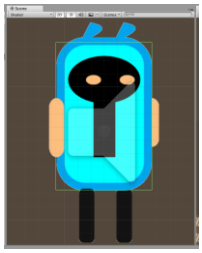
Parameter	Nilai
Besar Sudut	90.0
Besar Radius	40.0

Tabel 5 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 5 dan yang akan bervariasi adalah parameter jumlah raycast pada modul *dynamic lighting* yang akan dibuat.

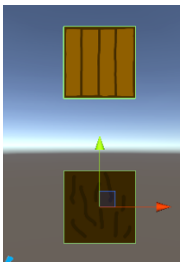
2.2 Implementasi

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



Gambar 6 Implementasi *Collider Character Player*

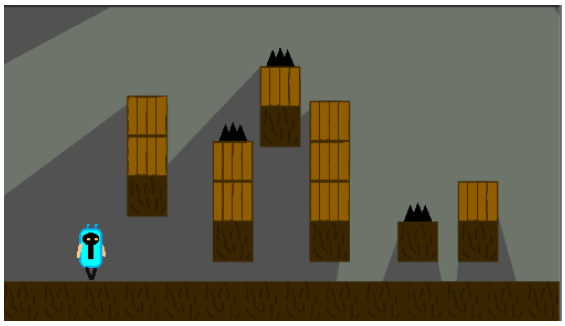
Gambar 6 adalah implementasi *collider* pada karakter pemain yang nantinya objek tersebut dapat menerima raycast dari modul dynamic lighting yang dibuat. Sesuai dengan perancangan bentuk *collider* adalah persegi pada tubuh karakter pemain.



Gambar 7 Implementasi *Collider Obstacle*

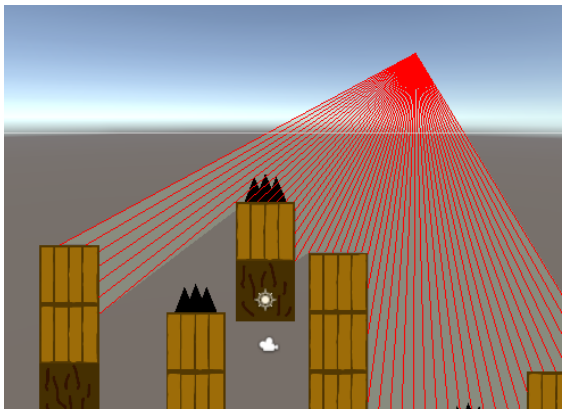
Gambar 7 adalah implementasi *collider* pada *platform* (bawah) dan *wall* (atas) yang nantinya ia dapat menerima raycast dari modul dynamic lighting yang dibuat. Sesuai dengan perancangan bentuk *collider* adalah persegi yang bentuknya sesuai dengan wujudnya.

Dari kode yang telah diimplementasikan, dapat dihasilkan suatu modul dynamic lighting pada Gambar 8 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 8 Implementasi Dynamic Lighting Pada Game

Pada Gambar 9 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.

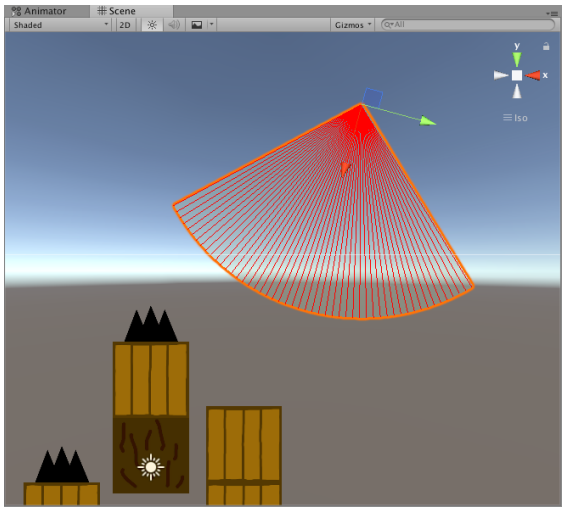


Gambar 9 Visualisasi Algoritme *Raycasting*

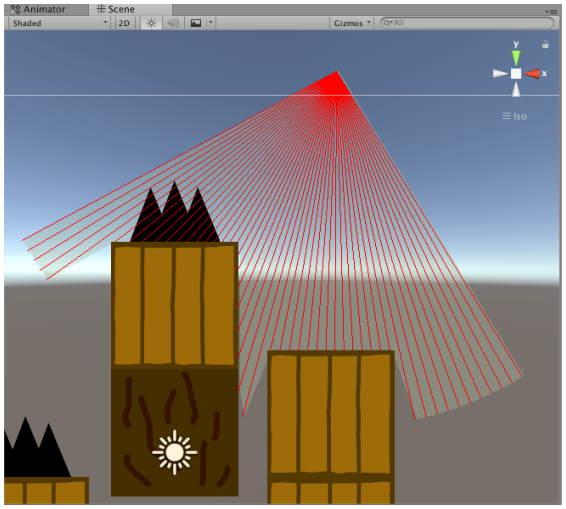
Pada Gambar 10 dapat dilihat hasil implementasi yang mana raycast tidak mengenai collider sehingga titik raycast berhenti sesuai pada jarak maksimum sesuai dengan jarak radius yang ditentukan sehingga bentuk polygon yang dihasilkan adalah dari titik raycast yang berhenti pada jarak maksimum.

Pada Gambar 11 dapat dilihat hasil implementasi yang mana 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*.



Gambar 10 Implementasi Polygon Apabila Raycast Tidak Mengenai Collider



Gambar 11 Implementasi Polygon Apabila Raycast Mengenai Collider

3. HASIL PENELITIAN

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

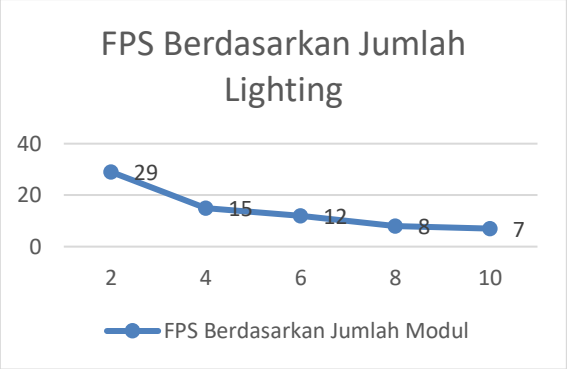
1. Modul *Dynamic Lighting* dapat membentuk bayangan sesuai *collider* objek
2. Bentuk *Polygon* Cahaya dapat berubah apabila modul berubah posisi
3. Bentuk *Polygon* Cahaya dapat berubah apabila objek berubah posisi
4. Bentuk *Polygon* Cahaya membentuk sudut sesuai nilai yang dimasukan
5. Bentuk *Polygon* Cahaya membentuk radius sesuai nilai yang dimasukan
6. Banyaknya cahaya/raycast sesuai nilai yang dimasukan

Tabel 6 Tabel Pengujian Fungsional

No	Parameter	Hasil	Status
1	Modul Dynamic Lighting dapat membentuk bayangan sesuai collider objek	Modul Dynamic Lighting membentuk bayangan sesuai collider objek	Valid
2	Bentuk polygon cahaya dapat berubah apabila modul berubah posisi	Bentuk polygon cahaya berubah apabila modul berubah posisi	Valid
3	Bentuk polygon cahaya dapat berubah apabila objek berubah posisi	Bentuk mesh polygon cahaya berubah apabila objek berubah posisi	Valid
4	Bentuk polygon cahaya membentuk sudut 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 radius yang dimasukan	Valid
6	Banyaknya cahaya/raycast sesuai nilai yang dimasukan	Banyaknya cahaya/raycast yang dipancarkan sesuai jumlah nilai yang dimasukan	Valid

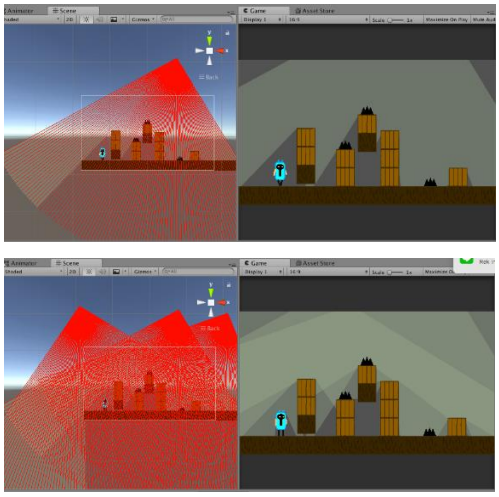
Pada Tabel 6 telah dijabarkan mengenai pengujian fungsional yang dilakukan pada modul yang telah diimplementasikan. Dari 6 parameter pengujian yang diuji didapatkan hasil 100% valid. Nomor 1 adalah hasil pengujian parameter uji fungsional yang pertama dan didapatkan hasil valid karena bentuk *dynamic lighting* sesuai dengan *collider* yang diimplementasikan pada objek yang dirancangan sebelumnya. Nomor 2 adalah hasil pengujian parameter uji fungsional yang ke-dua dan didapatkan hasil valid karena bentuk polygon *dynamic lighting* dapat berubah apabila posisi objek dynamic lighting berbah. Nomor 3 adalah

hasil pengujian parameter uji fungsional yang ke-tiga dan didapatkan hasil valid karena bentuk polygon *dynamic lighting* dapat berubah apabila posisi objek dynamic lighting berbah. 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 diinputkan pada modul. 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 diinputkan pada modul. 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 inputan resolusi cahaya pada modul.



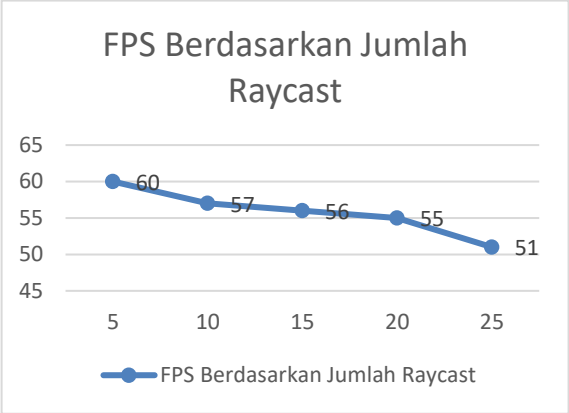
Gambar 12 Grafik FPS Berdasarkan Jumlah Modul

Gambar **12** 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, & 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.



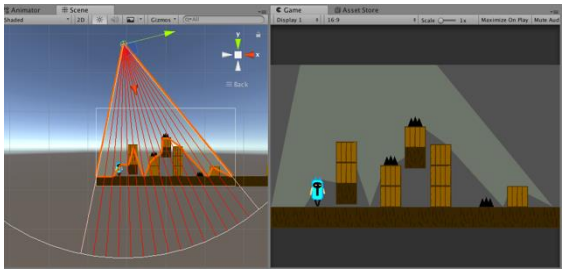
Gambar 13 *Screenshoot* Pengujian *FPS* Berdasarkan Jumlah Modul

Pada Gambar 13 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 4 modul yang diimplementasikan secara bersamaan.



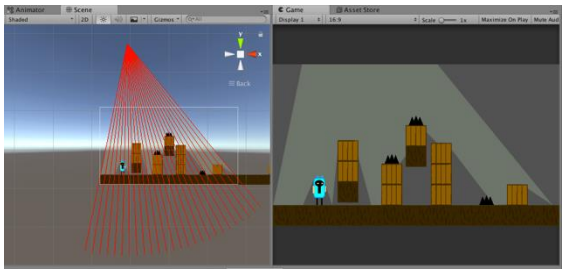
G**ambar 14** Grafik FPS Berdasarkan Jumlah Raycast

Gambar **14** 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 15 Pengujian Resolusi Raycast Dengan Nilai 15

Pada Gambar 15 dan Gambar 16 adalah *screenshoot* pengujian dengan mengubah nilai resolusi raycast pada parameter modul. Perbedaan yang terlihat adalah apabila resolusi modul semakin kecil maka bentuk *polygon dynamic lighting* menjadi tidak akurat. Apabila resolusi semakin besar maka *bentuk polygon dynamic lighting* semakin akurat.



Gambar 16 Pengujian Resolusi Raycast Dengan Nilai 25

4. KESIMPULAN DAN SARAN

4.1 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 teknik *raycasting*, yang mana *visibility polygon* dapat terbentuk menggunakan teknik *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.

4.2 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 teknik *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 *raycasting* dapat juga digunakan untuk modul *field-of-view* suatu *npc* pada *video game*, dengan memanfaatkan kemampuan deteksi suatu *raycast* saat mengenai *collider* objek pada *game*.

DAFTAR PUSTAKA

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

Ghosh, S. K. (2007). *Visibility algorithms in the plane*. *Visibility Algorithms in the Plane* (Vol. 9780521875). https://doi.org/10.1017/CBO9780511543340

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

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

Rick Taylor. (2014). 5 Reasons Video Games Lead American Entertainment | HuffPost. Diambil 28 Agustus 2018, dari https://www.huffingtonpost.com/rich-taylor/5-reasons-video-games-lea_b_5309230.html

Salmon, R. A., Armstrong, M. G., & Jolly, S. J. E. (2011). Higher Frame Rates for More Immersive Video and Telvision. *BBC Research White Paper 209*, (October). https://doi.org/10.1007/s10854-014-1999-7

Sam Byford. (2014). Pixel art games aren't retro, they're the future | The Verge. Diambil 28 Agustus 2018, dari https://www.theverge.com/2014/7/3/5865849/pixel-art-is-here-to-stay

William Usher. (2014). 75% Of Gamers Say That Graphics Do Matter When Purchasing A Game. Diambil 28 Agustus 2018, dari https://www.cinemablend.com/games/75-Gamers-Say-Graphics-Do-Matter-Purchasing-Game-64659.html