MJI

Media Jurnal Informatika
Vol. X, no.X, [nama_bulan] 20XX
http://jurnal.unsur.ac.id/mjinformatika

p-issn : 2088-2114
e-issn : 2477-2542

Implementasi Algoritma *Jump Point Search* Pada Game *Prince Of Walangsungsang*

Rio Andriyat Krisdiawan [1], Heru Budianto [2], Fitra Imaniar Romadhon [3]

Teknik Informatika Fakultas Ilmu Komputer Universitas Kuningan[13], Sistem Informasi Fakultas Ilmu Komputer Universitas Kuningan[2]
rioandriyat@uniku.ac.id, heru.budianto@uniku.ac.id, imaniar.fitraa@gmail.com

Abstract

The development of the gaming industry has provided an opportunity to combine entertainment with education. In this context, this study aims to implement the Jump Point Search (JPS) algorithm in the game "Prince of Walangsungsang" to provide a more interesting playing experience and introduce players to the history of Prince Walangsungsang journey. This study focuses on the action role-playing game (RPG) genre that combines game elements with historical stories. In making games, the development method used is the Game Development Life Cycle (GDLC) to ensure structured and efficient development steps. The JPS algorithm is implemented as an artificial intelligence (AI) agent on non-player characters (NPC) in games. By implementing the JPS algorithm, in-game NPCs can optimize their movements in a complex game environment, speed up finding the shortest path, and avoid unnecessary routes, as well as avoid dynamic obstacles. Games "Prince of Walangsungsang" provide a more realistic and challenging gaming experience for players, as well as introduce historical aspects through interactions with NPCs. The results of the study show that the implementation of the JPS algorithm in the game "Prince of Walangsungsang" improves the efficiency and intelligence of NPCs, so that players can experience a more dynamic and interesting playing experience. In addition, game-based introduction to the history of Prince Walangsungsang also gives players the opportunity to learn about and appreciate the cultural heritage associated with this historical figure.
Keywords: Action RPG Game, Prince Walangsungsang History, Jump Point Search Algorithm

Abstrak

Perkembangan industri game telah memberikan peluang untuk menggabungkan hiburan dengan pendidikan. Dalam konteks ini, penelitian ini bertujuan untuk mengimplementasikan algoritma Jump Point Search (JPS) pada game "Prince of Walangsungsang" untuk memberikan pengalaman bermain yang lebih menarik dan memperkenalkan sejarah perjalanan Pangeran Walangsungsang kepada pemain. Studi ini berfokus pada genre game aksi peran (RPG) yang menggabungkan elemen permainan dengan cerita sejarah. Dalam pembuatan game, metode pengembangan yang digunakan adalah Game Development Life Cycle (GDLC) untuk memastikan langkah-langkah pengembangan yang terstruktur dan efisien. Algoritma JPS diimplementasikan sebagai agen kecerdasan buatan (AI) pada karakter non-pemain (NPC) dalam game. Dengan menerapkan algoritma JPS, NPC dalam game dapat mengoptimalkan pergerakan mereka dalam lingkungan permainan yang kompleks, mempercepat pencarian jalan terpendek, dan menghindari rute yang tidak perlu, serta menghindari obstacle yang bersifat dinamis. Games "Prince of Walangsungsang" memberikan pengalaman bermain yang lebih realistis dan menantang bagi pemain, serta memperkenalkan aspek sejarah melalui interaksi dengan NPC. Hasil penelitian menunjukkan bahwa implementasi algoritma JPS pada game "Prince of Walangsungsang" meningkatkan efisiensi dan kecerdasan NPC, sehingga pemain dapat merasakan pengalaman bermain yang lebih dinamis dan menarik. Selain itu, media pengenalan sejarah Pangeran Walangsungsang berbasis game juga memberikan pemain kesempatan untuk mempelajari dan menghargai warisan budaya yang terkait dengan tokoh sejarah ini.

Kata kunci: Game Action RPG, Sejarah Pangeran Walangsungsang, Algoritma Jump Point Search

I. PENDAHULUAN

Kerajaan Cirebon merupakan sebuah kerajaan bercorak Islam ternama yang berasal dari Jawa Barat. Kesultanan Cirebon berdiri pada abad ke-15 dan 16 Masehi. Kesultanan Cirebon juga merupakan pangkalan penting yang menghubungkan jalur perdagangan antar pulau. Kesultanan Cirebon berlokasi di pantai utara pulau Jawa yang menjadi perbatasan antara wilayah Jawa Tengah dan Jawa Barat, ini membuat Kesultanan Cirebon menjadi pelabuhan sekaligus “jembatan” antara 2 kebudayaan, yaitu budaya Jawa dan Sunda.

Pangeran Walangsungsang atau dikenal juga Pangeran Cakrabuana mungkin tidak pernah menyangka bahwa wilayah yang dulu dibuka dan ditinggali akan menjadi cikal bakal dari wilayah Cirebon seperti saat ini. Tanah yang dibabat/dibangun dan ditempati pada tanggal

1 Syuro 790 Hijriah atau 650 tahun lalu, atau pada 1445 Masehi [1].

Pesatnya perkembangan zaman dan kemajuan teknologi dan informasi mendukung arus globalisasi yang semakin mencuat. Kebudayaan sejatinya adalah warisan yang wajib harus dilestarikan. Berdasarkan hasil observasi dan wawancara yang dilakukan ke keraton kasepuhan cirebon, dan didukung dengan penyebaran angket quesioner ke masyarakat cirebon dan pengunjung keraton kasepuhan cirebon, banyak masyarakat yang belum mengetahui sejarah cirebon, terutama pendiri Kerajaan Cirebon, terutama sejarah dari perjalanan pangeran walangsungsang. Berdasarkan data hasil kuesioner dari 50 orang yang tinggal diluar wilayah Cirebon dan 25 orang yang tinggal di wilayah Cirebon, didapat 80% belum mengetahui tentang sejarah Pangeran Walangsungsang, dan didapat 80% yang belum mengetahui tentang sejarah Pangeran Walangsungsang.

Games adalah sebuah sarana hiburan yang memiliki aturan untuk tujuan menjadi pemenang. Permainan (games) adalah cara bermain dengan mengikuti aturan-aturan tertentu yang dapat dilakukan secara individu maupun berkelompok guna mencapai tujuan tertentu[1]. Pendapat lain tentang Game merupakan suatu program yang dirancang sedemikian rupa dimana pemainnya berinteraksi dengan konflik buatan yang memiliki aturan dan memiliki hasil keluaran[3].

Di era milenial ini, kita melihat adanya berbagai jenis game, mulai dari game aksi, petualangan, peran (RPG), olahraga, hingga game simulasi dan strategi. Perkembangan teknologi seperti grafika 3D, realitas virtual (VR), dan realitas augmentasi (AR) telah mengubah cara kita berinteraksi dengan game, seperti sarana pembelajaran menggunakan game edukasi, bisnis menggunakan game online, bahkan untuk mengikuti ajang perlombaan E-Sport. Jenis atau genre game yang dimainkan juga bermacam-macam, seperti RPG (Role-Play Game), RTS (Real-Time Strategy), atau FPS (First-Person Shooter) yang saat ini populer dimainkan orang-orang. Selain itu, game juga telah menjadi platform untuk cerita yang mendalam dan kompleks. Kualitas narasi dalam game modern semakin meningkat, dengan pengembang yang menghadirkan alur cerita yang memikat, karakter yang mendalam, dan pilihan moral yang berdampak.

Dalam membangun game Action *RPG*, terdapat beberapa hal yang perlu diperhatikan, agar permainan serasa sangat menyenangkan dan alur permainan yang disajikan menjadi daya tarik pemain, salah satunya adalah dalam mengatur pergerakan *Non Player Character* (*NPC). Non player character (Non Players Character) is any character in the game that is not controlled by the player, such as monsters, villagers, and animals in the forest. NPCs represent characters in the story or game and have the ability to improvise their actions.*[4] Pergerakan NPC yang dinamis dan natural bertujuan menambah keseruan dan tantangan. Pergerakan NPC dalam permainan membutuhkan sebuah AI (Artificial Intelligence) atau kecerdasan buatan, dengan prilaku yang dapat menghampiri pemain dengan cepat dan efisien serta dibutuhkan sebuah algoritma untuk pencarian rute tercepat dalam menghampiri player.

Algoritma *JPS* adalah salah satu algoritma *Pathfinding* yang dapat dipakai dalam mencari dan mengenali jalur dari satu titik ke titik lain. Algoritma *Jump Point Search (JPS)* merupakan pengembangan dari algoritma A* yang dikembangkan oleh “Daniel Harabor dan Alban Grastien pada tahun 2011[5] Algoritma JPS algoritma yang kompetitif dan lebih cepat dari HPA*, sederhana tetapi efektif mempertahankan optimalitas tetapi tidak memerlukan memori tambahan, dan sangat cepat tetapi tidak memerlukan preprocessing[6]. Dengan penerapan algoritma *JPS* akan membantu *NPC* dalam mengoptimalkan pergerakan mereka dalam lingkungan game yang kompleks, dengan mencari rute terpendek dan menghindari rute yang tidak perlu.

II. METODE PENELITIAN

II.1 Meiode Pengumpulan Data

a. Observasi

Guna memperoleh data untuk kebutuhan proposal dan aplikasi yang akan dibangun, maka penulis melakukan observasi ke Keraton Kasepuhan Cirebon. Observasi dilakukan dengan melakukan kunjungan ke museum Keraton Kasepuhan Cirebon.

b. Wawancara

Dalam memperoleh data untuk kebutuhan proposal dan aplikasi yang akan dibangun, maka penulis melakukan wawancara ke Keraton Kasepuhan Cirebon. Wawancara ini berlangsung dengan narasumber bernama Bapak Rudi selaku sesepuh yang mengetahui sejarah Cirebon sekaligus pemandu Museum Keraton Kasepuhan Cirebon.

c. Studi Pustaka

Pada Sebagai penunjang dalam penelitian, penulis melakukan pengumpulan data berupa studi pustaka dari berbagai sumber seperti google play buku, jurnal, internet dan buku yang yang berjudul “Pangeran Cakrabuana : Sang Perintis Kerajaan Cirebon” yang diterbitkan oleh Kiblat Buku Utama.

II.2 Metode Penyelesaian Masalah

Algoritma *JPS* merupakan pengembangan dari algoritma *A** yang dikembangkan oleh “Daniel Harabor dan Alban Grastien pada tahun 2011”[6]. Algoritma ini dikatakan adalah algoritma yang kompetitif dan lebih cepat dari *HPA**, sederhana tetapi efektif, mempertahankan optimalitas tetapi tidak memerlukan memori tambahan, dan sangat cepat tetapi tidak memerlukan preprocessing. Penerapan algoritma JPS dalam *game* yang menggambarkan cerita sejarah Pangeran Walangsungsang diterapkan untuk mengenali keadaan sekitar dari *NPC* sehingga dapat mencari jalur terbaik dalam mengejar pemain. Algoritma *JPS* dilakukan untuk mempercepat proses pencarian jalur dan mengurangi penggunaan memori untuk

open dan closed list dalam menyimpan *node*s pada algoritma *A**. Dalam *JPS* penyimpanan *node*s akan dipangkas menjadi hanya beberapa yang disebut sebagai *jump point*.

Pseudocode Identifikasi *successor*

Require: *x*: current *node*, *s*: start, *g*: goal

```
successor(x) ← Ø
neighbours(x) ← prune( x, neighbours(x))
for all n ∈ neighbours(x) do
    n ← jump(x, direction(x,n), s, g)
    add n to successor(x)
return successor(x)
```

Pseudocode Fungsi *Jump Search*

Require: *x*: initial *node*, *d*: direction, *s*: start, *g*: goal

```
n ← step(x,d)
if n is an obstacle or is outside the grid then
return null
if n = g then
    return n
if ∃ n' ∈ neighbours(n) s.t. n' is force then
    return n
if d is diagonal then
    for all i ∈ {1,2} do
        if jump(n, d, s, g) is not null
    then
            return n
return jump(n,d,s,g)
```

Sumber : (Delima, Indrajaya, Takaredase, E.K.R., & C. , 2016)[7]

Notasi dan terminologi yang dipakai dalam algoritma JPS adalah sebagai berikut :

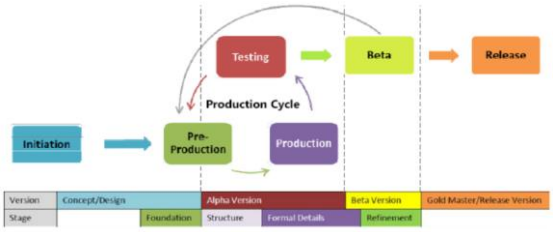
1. ***s*** adalah start atau *node* awal
2. ***g*** merupakan goal atau *node* tujuan
3. ***neighbours(x)*** adalah *node* sekitar *node* ***x***
4. ***y*** dapat dicapai dari ***x*** dengan melewati beberapa *node* pada arah yang ditentukan, ***y*** dapat dinyatakan sebagai berikut :

$$y = x + kd \text{ ……..(1)}$$

x = *node* yang aktif
k = nilai dari unit ***x*** melangkah ke ***y***
d = arah yang dituju
y = *node* calon suksesor ***x***

II.3 Metode Pengembangan Game

Metode pengembangan sistem yang digunakan dalam penelitian ini adalah metode *GDLC (Game Development Life Cycle)*. *GDLC* adalah suatu proses pengembangan sebuah *game* yang menerapkan pendekatan iteratif yang terdiri dari 6 fase pengembangan, dimulai dari fase *inisialisasi* atau pembuatan konsep, *pre production*, *production*, *testing*, *beta* dan *release*. [8]



Gambar 1. Fase dan Proses GDLC[8]

Pada gambar 1 dari 6 fase tersebut dapat dikelompokkan menjadi 3 proses utama yaitu:

1. Proses Inisialisasi yang terdiri dari konsep dan design.
2. Proses produksi terdiri dari Pra Produksi, Produksi, dan Pengujian (*Alpha* dan *Beta*)
3. *Release*

Berikut 6 tahapan dan *GDLC (Game Development Life Cycle)* penjelasannya:

1) Inisiasi

Pada tahapan ini penulis melakukan pembuatan konsep dasar *game*, pertama *game* yang akan dibuat adalah *game* yang menggambarkan cerita Pangeran Walangsungsang yang ber*genre Action RPG*, serta target *user* yang dituju adalah masyarakat luas. Pada tahap ini penulis sudah menghasilkan konsep *game* dan deskripsi permainan.

2) Pra-produksi

Pada tahapan ini penulis masuk ke penciptaan dan revisi desain permainan, dan pembuatan prototipe permainan. Permainan berfokus pada genre *Action RPG* dengan alur cerita yang menggambarkan cerita Pangeran Walangsungsang. Tantangan dan faktor kesenangan yang diberikan yaitu menghindari musuh, melawan musuh dan mencari *Item Quest* dengan tahapan tingkat kesulitan mudah, sedang dan sulit. Aspek teknis yang menggambarkan tempat Kerajaan Pajajaran. Tahapan ini selesai ketika hasil revisi atau perubahan desain *game* telah disetujui dan didokumentasikan di *GDD*.

3) Produksi

Pada tahap ini penulis mulai melakukan penciptaan *aset*, penulisan kode program (*coding)* dengan menggunakan bahasa pemrograman *C#* dan pembuatan *game* menggunakan *Game Engine Unity 3D*.

4) Pengujian

Pada tahapan ini penulis melakukan pengujian kegunaan permainan dan pemutaran, Untuk menilai

fungsionalitas fitur dan kesulitan permainan penulis hanya menjalankan pengujian melalui *Game Engine Unity*, sedangkan metode untuk menguji kriteria kualitas fungsional pengujian dilakukan melalui *game* yang sudah di *build* dan di mainkan di perangkat *smartphone*. Hasil dari tahapan ini adalah laporan *bug*, permintaan perubahan, dan keputusan pengembangan, jika hasil nya tidak ada akan di lanjut ke fase *Beta*.

5) Beta

Pada tahap *beta* pengujian dilakukan oleh pihak ketiga atau eksternal, penulis melakukan 2 jenis metode pemilihan testing yaitu beta tertutup (*Close Beta*) yang hanya memungkinkan individu yang diundang untuk menjadi peserta dan beta terbuka (*Open Beta*) yang memungkinkan siapa saja yang menjadi peserta. Hasil dari beta ini adalah laporan bug dan masukan dari para penguji atau pengguna.

6) Rilis

Tahapan adalah proses dimana *game* yang sudah selesai dibuat dan lulus beta testing yang melibatkan peluncuran produk, dokumentasi proyek, berbagi pengetahuan, post mortems, penulis melakukan publikasi aplikasi menggunakan website.

III. HASIL PENELITIAN

1. Langkah Initiation (Inisiasi)

Analisis Kebutuhan Pengguna (User)

Adapun kebutuhan pengguna akan game yang dibangun antara lain :

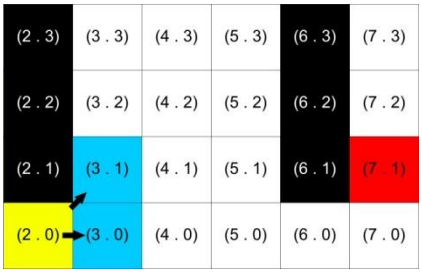
a. Memiliki pengalaman dan pengetahuan tentang cerita sejarah.
b. Memiliki unsur hiburan dan daya tarik untuk di mainkan.
c. Memiliki kemudahan dalam instalasi dan kegunaan.
d. Dapat dimainkan di smartphone dengan OS android.
e. Memiliki tantangan.

Analisis Kebutuhan AI

Disetiap musuh pada permainan terdapat sebuah sistem AI (Artificial Intelligence) untuk pergerakan dan kecerdasan musuh. Di dalam game yang menggambarkan cerita sejarah Pangeran Walangsungsang setiap musuh memiliki AI (Artificial Intelligence) dimana musuh akan bisa berjalan dan menghampiri pemain, dan bukan hanya itu jika pemain bersembunyi masuk ke rumah atau diluar jangkauan musuh, maka musuh akan kembali ke tempat semula dengan sistem AI (Artificial Intelligence) yang menggunakan algoritma Jump Point Search.

Analisis Penerapan Algoritma Jump Point Seaarch

Berikut adalah cara kerja algoritma Jump Point Search dengan perhitungan manual yang diterapkan untuk pergerakan NPC.



Gambar 2. Mencari Node tetangga (2,0)

Pada gambar 2 *node* yang berwarna hijau adalah *start* atau *node* awal, *node* yang berwarna merah adalah *goal* atau *node* tujuan, *node* yang berwarna hitam adalah *node* yang tidak bisa dilewati, sementara *node* yang berwarna putih adalah *node* x atau *node* aktif. Angka yang berada di setiap *node* adalah nilai koordinat x dan y.

***s* = (2 , 0)**
***g* = (7 , 1)**
***x* = *node* aktif**
g(y) = Biaya dari *node* ***x***
h(y) = Perkiraan biaya minimum dari *node* ***y*** ke *node* ***g***
f(y) = Nilai dari *node* ***y***

Rumus yang ada pada algoritma *Jump Point Search :*

2. ***jump((x, direction(x,n), s, g)…….(2)***

3. $$g(y)=\sqrt{(direction(x,n).x - x.x)^2 + (direction(x,n).y - x.y)^2} \quad ...(3)$$

4. $$h(y) = \sqrt{(direction(x,n).x - g.x)^2 + (direction(x,n).y - g.y)^2} \quad ...(4)$$

5. **f(y) = g(y) + h(y)…….(5)**

p(x) = (2,0)
neighbours(x) = (3,1), (3,0)

Melakukan *looping* sesuai banyaknya *node* tetangga yang di dapat.
(kondisi 1)
neighbours(x) = (3,0)
n = jump((2,0), (3,0), (2,0), (7,1))

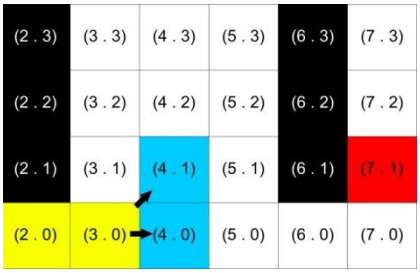
1. Apakah nilai *node* ***n*** adalah nilai ***node tidak aktif*** atau di luar map = ***Tidak***.
2. Apakah nilai *node* ***n*** adalah nilai ***node g*** = ***Tidak.***

Karena kedua kondisi tidak memenuhi syarat maka lanjut ke pencarian :

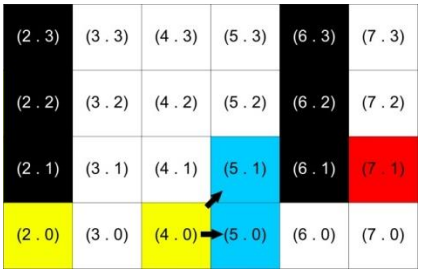
tDx = 3 – 2 = 1
tDy = 0 – 0 = 0

karena nilai ***tDx*** tidak sama dengan ***0*** maka **:**
jika horizontal atau vertikal :
direction(x,n).x + tDx , direction(x,n).y + 1 = (4,1) adalah *node* aktif dan ***direction(x,n).x, direction(x,n).y + 1 = (3,1)*** adalah *node* tidak aktif.
karena *node* ***(3,1)*** adalah *node* aktif maka tidak terpenuhi .

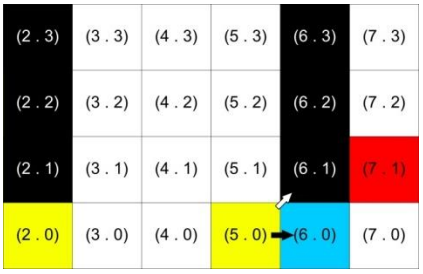
Lakukan *jump point* vertikal :
jika vertikal :
direction(x,n).x + tDx , direction(x,n).y = (4,0) adalah *node* aktif.
Karena kondisi terpenuhi maka :
x.x = direction(x,n).x
y.y = direction(x,n).y
direction(x,n).x + tDx = 4
direction(x,n).y + tDy = 0
n = jump((3,0), (4,0), (2,0), (7,1))



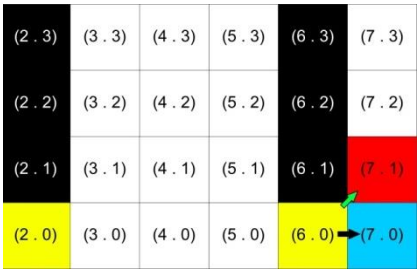
Gambar 3. Mencari Node tetangga (3,0)



Gambar 4. Mencari Node tetangga (4,0)

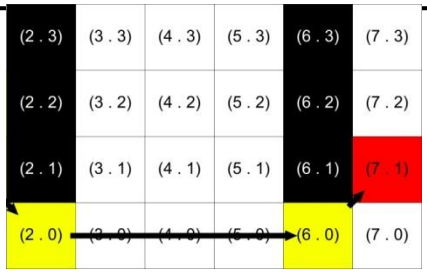


Gambar 5. Mencari Node tetangga (5,0)



Gambar 6. Mencari Node tetangga (6,0)

Proses ini akan terus berulang hingga menemukan sebuah node seperti gambar 6, node (6,0) yang menjadi calon successor yang bisa di jangkau oleh node (2,0)



Gambar 7. Jalur yang ditempuh

Pada gambar 7 bisa dilihat bahwa algoritma *Jump Point Search* melakukan 2 kali lompatan dari *node* (2,0), (6,0) terakhir (7,1) atau *node* tujuan.

2. Langkah Pra-Production

Story Board Game

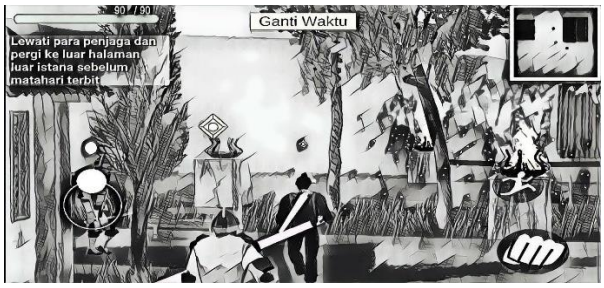
Story Board Game menggambarkan rangkaian sketsa gambar yang tersusun secara berurutan menggambarkan perubahan penting dari adegan dan aksi yang digunakan untuk menggambarkan alur cerita permainan yang akan dibuat[9]. Berikut adalah rancangan storyboard game pangeran walangsungsang yang digambarkan dibawah ini.

Table 1. StoryBoard Scene 1

Scene 1	Kerajaan Pajajaran	1/5
		
Lokasi	Kerajaan Pajajaran	
Aksi	Saat permainan dimulai, pemain akan di perkenalkan dengan suasana kerajaan Pajajaran. Pemain juga akan diberi bantuan bagaimana cara menggerakan karakter dan memperkenalkan menu-menu interface. Tab quest memberikan deskripsi tujuan permainan untuk diselesaikan, pemain juga akan diberi petunjuk arah kemana harus bergerak.	

Table 2. StoryBoard Scene 3

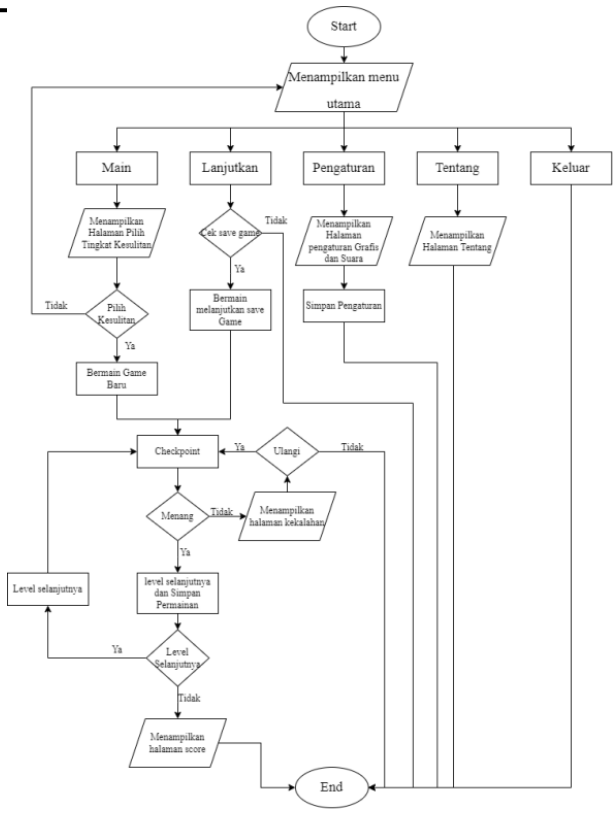
Scene 3	Malam di Kerajaan Pajajaran	3/5



Lokasi	Kerajaan Pajajaran
Aksi	Pada saat malam hari jika pemain menggerakan karakternya ke luar rumah, maka penjaga di area tersebut dan menghadap ke pemain akan langsung mengejar dan menangkap pemain, kecepatan penjaga yang tadinya 1 akan berubah menjadi 1.8 agar memungkinkan penjaga mengejar pemain dan dalam jarak 0.8 maka penjaga akan melakukan animasi memukul dan jika mengenai pemain maka darah pemain akan berkurang dan pemain akan melakukan animasi terpukul, jika pemain tertangkap dengan syarat darah pemain mencapai angka nol, maka misi akan dianggap gagal dan pemain bisa mengulang dari awal *stage* atau dari *checkpoint*, namun jika pemain masuk ke rumah maka penjaga akan kembali ke tempatnya dan melakukan patroli.

Design Game Layout Chart yang diusulkan.

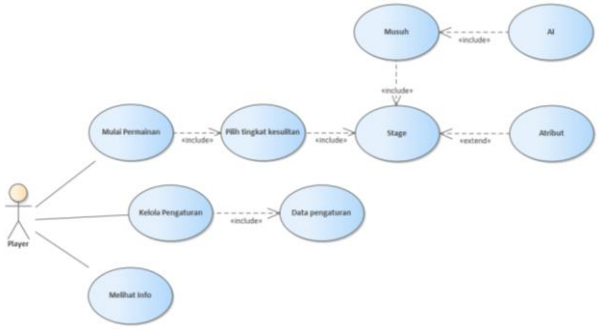
Game layout chart merupakan sebuah dokumen yang menjelaskan struktur sebuah game yang didalamnya termasuk konten dari game tersebut [9]. Berikut game layout chart pangeran walangsungsang.



Gambar 8. Game Layout Chart

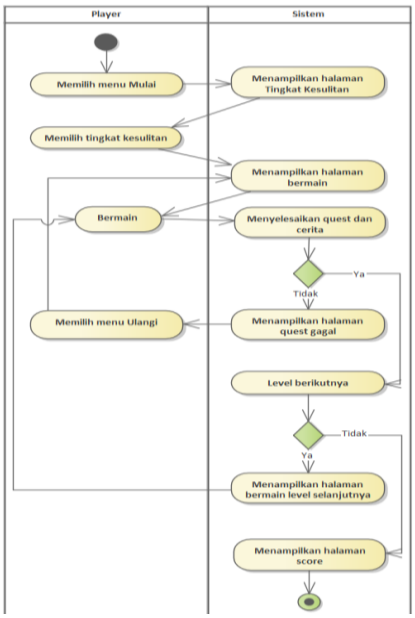
Perancangan Usecase

Use case diagram ini akan menggambarkan fungsi yang dapat dilakukan oleh sistem di dalam *game* Raden Walangsungsang antara *player* (*actor*) dengan sistem. Berikut pada gambar 3.17 *use case* diagram sistem *game* Raden Walangsungsang.

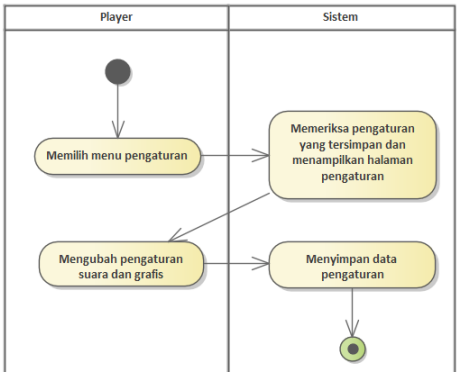


Gambar 9. Use Case Diagram

Perancangan *Activity* Diagram

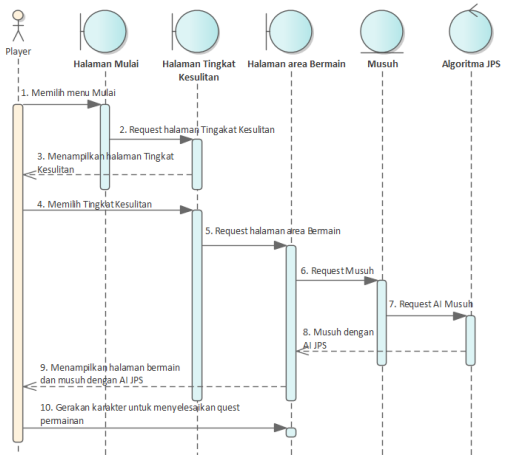


Gambar 10. Activity Diagram Mulai Permainan

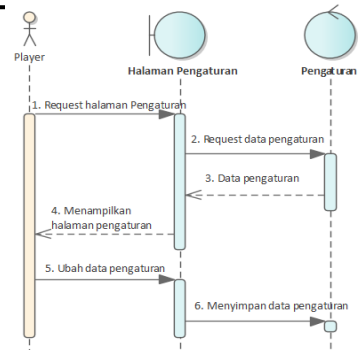


Gambar 11. Activity Diagram Kelola Pengaturan

Perancangan Sequence Diagram



Gambar 12. Squence Diagram Mulai Permainan



Gambar 13. Squence Diagram Kelola Pengaturan

Perancangan Class Diagram

Gambar 14. Class Diagram

3. Langkah Production

Implementasi Desain Interface

Gambar 15. Halaman Menu Utama

Gambar 16. Halaman Pengaturan

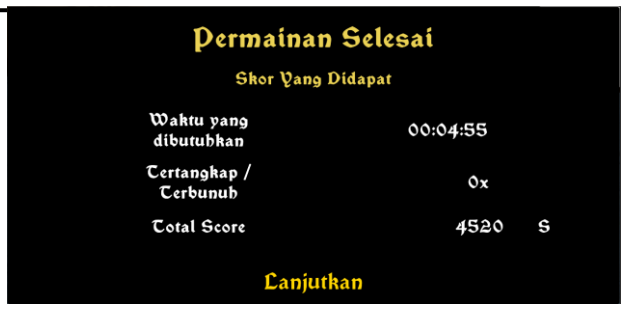
Gambar 17. Halaman Tentang

Gambar 18. Halaman Gameplay 1

Gambar 19. Halaman Dialog

Gambar 20. Halaman Pause Menu

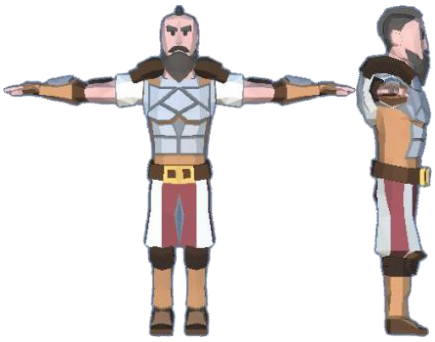
Gambar 21. Mendapatkan Item



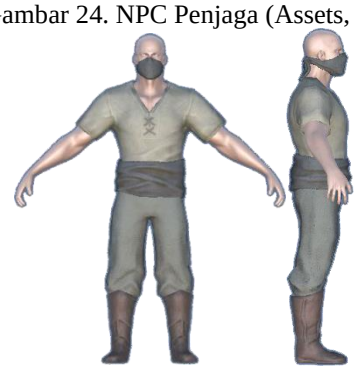
Gambar 22. Halaman Game Win

Pembuatan Game Object

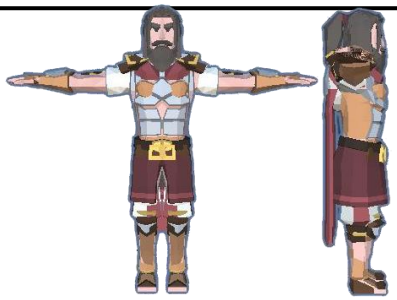
Gambar 23. Karakter Pemain



Gambar 24. NPC Penjaga (Assets, 2022)



Gambar 25. NPC Bandit (Morillas, 2017)



Gambar 26. NPC Raja Pajajaran (Assets, 2022)

IV. PEMBAHASAN

Pada pembahasan ini tidak lepas dari tahapan GDLC yaitu tahapan Alfa Testing, Beta testing dan Realease.

4. Langkah Alpha Testing

Pengujian Black Box

Tahap pengujian atau testing merupakan salah satu tahap yang harus ada dalam sebuah siklus pengembangan perangkat lunak, hasil pengujian dibantu oleh *beta test* yang memainkan permainan dan mendownload permainan tersebut melalui *website* penulis. Berikut ini adalah tabel pengujian *Black Box* :

Table 3. BlackBox Bermain

Pengujian *Black Box* dari Halaman Bermain				
No	Tombol	Aksi Pemain	Reaksi Sistem	Harapan
1	*Joystick*	Pemain menyentuh dan menggerakan *Joystick*	Karakter bergerak sesuai arah dari *Joystick*	Sesuai harapan
2	Serang	Pemain menekan tombol serang	Karakter melakukan gerakan serangan maksimal tiga kali *Combo*	Sesuai harapan
3	Tangkis	Pemain menekan tangkis	Karakter menangkis segala serangan musuh	Sesuai harapan
4	Loncat	Pemain menekan tombol loncat	Karakter loncat	Sesuai harapan
5	*Pause* Menu	Pemain menekan tombol *Pause* Menu	Sistem menampilkan halaman *Pause* Menu	Sesuai harapan

Pengujian White Box

Pengujian *White Box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, menggunakan struktur kontrol dari desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian.

Table 4. WhiteBox *Algoritma JPS*

1	public static List<GridPos> FindPath(JumpPointParam iParam)
	{
	IntervalHeap<*Node*> tOpenList = iParam.openList;
	Node tStart*Node* = iParam.Start*Node*;
	Node tEnd*Node* = iParam.End*Node*;
	Node t*Node*;
	bool revertEnd*Node*Walkable = false;
	tStart*Node*.startToCur*Node*Len = 0;
	tStart*Node*.heuristicStartToEndLen = 0;
	tOpenList.Add(tStart*Node*);
	tStart*Node*.isOpened = true;

2	while (tOpenList.Count > 0) {

3	t*Node* = tOpenList.DeleteMin();
	t*Node*.isClosed = true;
	identifySuccessors(iParam, t*Node*);
	}

4	return new List<GridPos>(); }

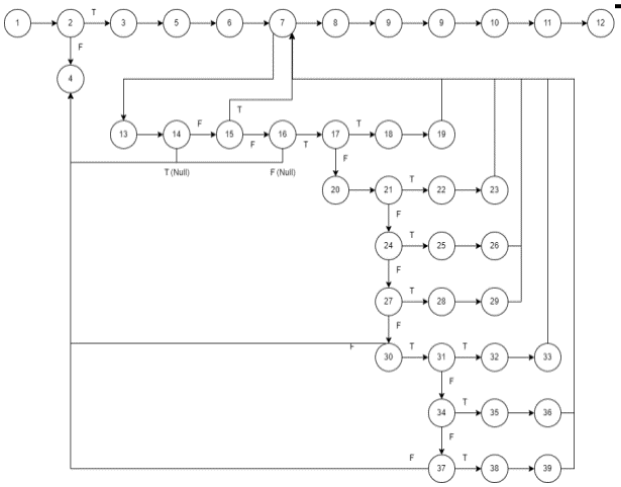
5	private static void identifySuccessors(JumpPointParam iParam, *Node* i*Node*)
	{
	HeuristicDelegate tHeuristic = iParam.HeuristicFunc;
	IntervalHeap<*Node*> tOpenList = iParam.openList;
	int tEndX = iParam.End*Node*.x;
	int tEndY = iParam.End*Node*.y;
	int tEndZ = iParam.End*Node*.z;
	GridPos tNeighbor;
	GridPos tJumpPoint;
	Node tJump*Node*;

6	List<GridPos> tNeighbors = findNeighbors(iParam, i*Node*);
	for (int i = 0; i < tNeighbors.Count; i++)
	{

7	tNeighbor = tNeighbors[i]; tJumpPoint = jump(iParam, tNeighbor.x, tNeighbor.y, tNeighbor.z, i*Node*.x, i*Node*.y, i*Node*.z);
8	if (tJumpPoint != null) {
9	if (!tJump*Node*.isOpened || tStartToJump*Node*Len < tJump*Node*.startToCur*Node*Len) {
10	tJump*Node*.startToCur*Node*Len = tStartToJump*Node*Len;

```
                tJump Node.heuristicCur NodeToEndLen = (tJump Node.heuristicCur NodeToEndLen == null ? tHeuristic(Math.Abs(tJumpPoint.x - tEndX), Math.Abs(tJumpPoint.y - tEndY), Math.Abs(tJumpPoint.z - tEndZ)) : tJump Node.heuristicCur NodeToEndLen);

                tJump Node.heuristicStartToEndLen = tJump Node.startToCur NodeLen + tJump Node.heuristicCur NodeToEndLen.Value;
                    tJump Node.parent = i Node;
11              if (!tJump Node.isOpened)
                {
12                  tOpenList.Add(tJump Node);
                }
13              private static GridPos jump(JumpPointParam iParam, int iX, int iY, int iZ, int iPx, int iPy, int iPz)
                {
14              if (!iParam.SearchGrid.IsWalkableAt(iX, iY, iZ))
                {
                    return null;
                }
15              else if (iParam.SearchGrid.Get NodeAt(iX, iY, iZ).Equals(iParam.End Node))
                {
                    return new GridPos(iX, iY, iZ);
                }
16              if (iParam.DiagonalMovement == DiagonalMovement.Always || iParam.DiagonalMovement == DiagonalMovement.IfAtLeastOneWalkable)
                {
17              if(tDx != 0 && tDy != 0 && tDz != 0)
                {
18              if ((iParam.SearchGrid.IsWalkableAt(iX - tDx, iY + tDy, iZ) && !iParam.SearchGrid.IsWalkableAt(iX - tDx, iY, iZ) && !iParam.SearchGrid.IsWalkableAt(iX - tDx, iY, iZ - 1)) {
19                  return new GridPos(iX, iY, iZ);
                }
                }
20              else
                {
21              if (tDx != 0 && tDy != 0)
                {
22              if ((iParam.SearchGrid.IsWalkableAt(iX - tDx, iY + tDy, iZ) && !iParam.SearchGrid.IsWalkableAt(iX - tDx, iY, iZ)) {
23                  return new GridPos(iX, iY, iZ);
                }
                }
24              else if(tDx != 0 && tDz != 0)
                {
25              if ((iParam.SearchGrid.IsWalkableAt(iX - tDx, iY, iZ + tDz) && !iParam.SearchGrid.IsWalkableAt(iX - tDx, iY, iZ)) {
26                  return new GridPos(iX, iY, iZ);
                }
                }
27              else if (tDy !=0 && tDz != 0)
                {
28              if ((iParam.SearchGrid.IsWalkableAt(iX, iY + tDy, iZ - tDz) && !iParam.SearchGrid.IsWalkableAt(iX, iY, iZ - tDz)){
29                  return new GridPos(iX, iY, iZ);
                }
                }
30              else
                {
31                  if (tDx != 0)
                    {
32              if ((iParam.SearchGrid.IsWalkableAt(iX + tDx, iY + 1, iZ) && !iParam.SearchGrid.IsWalkableAt(iX, iY + 1, iZ {
33                      return new GridPos(iX, iY, iZ);
                    }
                }
34                  else if (tDy != 0)
                    {
35              if ((iParam.SearchGrid.IsWalkableAt(iX + 1, iY + tDy, iZ) && !iParam.SearchGrid.IsWalkableAt(iX + 1, iY, iZ)) {
36                      return new GridPos(iX, iY, iZ);
                    }
                }
37                  else // tDz != 0
                    {
38              if ((iParam.SearchGrid.IsWalkableAt(iX, iY + 1, iZ + tDz) && !iParam.SearchGrid.IsWalkableAt(iX, iY + 1, iZ)) {
39                      return new GridPos(iX, iY, iZ);
                    }
```



Gambar 27. Flowgraph Jump Point Search

Cyclomatic complexity dari *flowgraph* diatas dapat dihitung dengan menggunakan rumus :

V(G) = E – N + 2

Diketahui :

E (jumlah edge pada flowgraph) = 50

N (Jumlah *node* pada flowgraph) = 39

V (G) = 50 – 39 + 2 = 13

Dari hasil perhitungan Cyclomatic complexity terdapat 13 path (jalur), yaitu :

1. ***Path* 1** = 1-2-3-5-6-7-8-9-10-11-12
2. ***Path* 2** = 1-2-4
3. ***Path* 3** = 1-2-3-5-6-7-13-14-4
4. ***Path* 4** = 1-2-3-5-6-7-13-14-15-7-8-9-10-11-12
5. ***Path* 5** = 1-2-3-5-6-7-13-14-15-16-4
6. ***Path* 6** = 1-2-3-5-6-7-13-14-15-16-17-18-19-7-8-9-10-11-12
7. ***Path* 7** = 1-2-3-5-6-7-13-14-15-16-17-20-21-22-23-7-8-9-10-11-12
8. ***Path* 8** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-25-26-7-8-9-10-11-12
9. ***Path* 9** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-27-28-29-7-8-9-10-11-12
10. ***Path* 10** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-27-30-4
11. ***Path* 11** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-27-30-31-32-33-7-8-9-10-11-12
12. ***Path* 12** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-27-30-31-34-35-36-7-8-9-10-11-12
13. ***Path* 13** = 1-2-3-5-6-7-13-14-15-16-17-20-21-24-27-30-31-34-37-38-39-7-8-9-10-11-12

5. Langkah Beta Testing

Untuk mengetahui tanggapan responden (user) terhadap aplikasi *Game* Walangsungsang yang dibuat, maka dilakukan pengujian dengan memberikan kepada 15 responden. Jawaban dari pertanyaan, terdiri dari tingkatan yang dipilih sebagai berikut :

Table 5. Pilihan Jawaban UAT

A	Sangat Bagus atau Sangat Puas
B	Bagus atau Puas
C	Cukup
D	Tidak Bagus atau Tidak Puas

Table 6. Bobot Nilai Jawaban

Jawaban	**Bobot**
Sangat Bagus	4
Bagus	3
Cukup	2
Tidak Bagus	1

Table 7. Pernyataan atau Kuesioner

No	**Pertanyaan**	**A**	**B**	**C**	**D**
1.	Apakah *user interface* menu aplikasi *Game Walangsungsang* menarik ?	8	7	0	0
2.	Apakah menu yang ada pada aplikasi *Game Walangsungsang* dapat dipahami?	15	0	0	0
3.	Apakah aplikasi *Game Walangsungsang* menarik atau seru untuk dimainkan ?	9	6	0	0
4.	Apakah dengan adanya aplikasi *Game Walangsungsang* ini anda mendapat pengetahuan tentang sejarah Pangeran Walangsungsang ?	7	6	2	0
5.	Bagaimana AI musuh pada aplikasi Game Walangsungsang ?	9	6	0	0
6	Apakah musik dan efek suara *Game Walangsungsang* ini enak untuk didengar ?	12	3	0	0
7	Apakah *graphic* dari aplikasi *Game Walangsungsang* ini bagus ?	12	3	0	0

Table 8. Data Kuesioner Yang Telah diproses

No	Pertanyaan	Ax4	Bx3	Cx2	Dx1	Jml	%
1	Apakah *user interface* menu aplikasi *Game Walangsungsang* menarik ?	32	21	0	0	53	88
2	Apakah menu yang ada pada aplikasi *Game Walangsungsang* dapat dipahami?	60	0	0	0	60	100
3	Apakah aplikasi *Game Walangsungsang* menarik atau seru untuk	36	18	0	0	54	90

No	Pertanyaan	Ax4	Bx3	Cx2	Dx1	Jml	%
	dimainkan?						
4	Apakah dengan adanya aplikasi *Game Walangsungsang* ini anda mendapat pengetahuan tentang sejarah Pangeran Walangsungsang ?	28	18	4	0	50	83
5	Bagaimana AI musuh pada aplikasi Game Walangsungsang ?	36	18	0	0	54	90
6	Apakah musik dan efek suara *Game Walangsungsang* ini enak untuk didengar ?	48	9	0	0	57	95
7	Apakah *graphic* dari aplikasi *Game Walangsungsang* ini bagus ?	48	9	0	0	57	95

1. Analisis Pertanyaan Pertama

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 53
   Nilai rata-ratanya adalah :
   53 / 15 = 3.53
   Presentasi nilainya adalah (3.53 / 4) x 100 = 88%.
2. Analisis Pertanyaan Kedua

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 60
   Nilai rata-ratanya adalah :
   60 / 15 = 4
   Presentasi nilainya adalah (4 / 4) x 100 = 100%.
3. Analisis Pertanyaan Ketiga

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 54
   Nilai rata-ratanya adalah :
   54 / 15 = 3.6
   Presentasi nilainya adalah (3.6 / 4) x 100 = 90%.
4. Analisis Pertanyaan Keempat

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 50
   Nilai rata-ratanya adalah :
   50 / 15 = 3.3
   Presentasi nilainya adalah (3.3 / 4) x 100 = 83%.
5. Analisis Pertanyaan Kelima

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 54
   Nilai rata-ratanya adalah :
   54 / 15 = 3.6
   Presentasi nilainya adalah (3.6 / 4) x 100 = 90%.
6. Analisis Pertanyaan Keenam

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 57
   Nilai rata-ratanya adalah :
   57 / 15 = 3.8
   Presentasi nilainya adalah (3.8 / 4) x 100 = 95%.
7. Analisis Pertanyaan Ketujuh

   Dari tabel diatas dapat dilihat bahwa jumlah nilai dari 15 responden untuk pertanyaan adalah 57
   Nilai rata-ratanya adalah :
   57 / 15 = 3.8
   Presentasi nilainya adalah (3.8 / 4) x 100 = 95%.

6. Realease

Pada tahapan ini, game yang telah dibuat didistribusikan ke keraton kasepuhan cirebon. Game yang telah dibuat dapat dijalankan di perangkat PC.

V. KESIMPULAN

Dari penelitian yang telah dilakukan bahwa permainan yang menggambarkan cerita sejarah Pangeran Walangsungsang dengan menggunakan algoritma *Jump Point Search* untuk *AI* (*Artificial Intelligence*) musuh pada permainan *Game Walangsungsang*, maka dapat disimpulkan bahwa :

1. Dengan adanya teknologi yang menggambarkan cerita sejarah pangeran walangsungsang yaitu permainan *Game Walangsungsang*, persentase pendapat masyarakat yang sudah memainkan permainan tersebut dan mendapat pengetahuan tentang sejarah Pangeran Walangsungsang adalah 46% Sangat Puas, 40% Puas, 13% Cukup dan 0% Tidak Puas.
2. Algoritma *Jump Point Search* yang digunakan sebagai *AI Pathfinding* musuh pada *Game Walangsungsang*, persentase pendapat masyarakat yang sudah memainkan permainan tersebut adalah 60% Sangat Bagus, 40% Bagus, 0% Cukup dan 0% Tidak bagus.
3. Algoritma *Jump Point Search* dapat di implementasikan secara optimal untuk *AI Pathfinding* musuh pada permainan *Game Walangsungsang* untuk mencari rute terdekat kepada pemain utama. Lalu kekurangannya pada saat algoritma bekerja, musuh tidak mempunyai *Physics3D* kepada sesama musuh, sehingga jika musuh banyak maka musuh akan saling bertumpukan.
4. Algoritma *Jump Point Search* tidak cocok untuk area (*Terrain*) permainan yang bergelombang, sehingga algoritma *Jump Point Search* yang diimplementasikan ini akan lebih optimal untuk permainan yang menggunakan area rata, tidak ada bukit atau pegunungan, atau pada game berbasis 2D.

UCAPAN TERIMA KASIH

Ucapan terimakasih kepada pihak keraton kasepuhan cirebon yang telah memberikan informasi dan data yang diperlukan dalam penelitian, serta masyarakat yang telah memberikan respon dalam membantu data penelitian dan Implementasi game yang dibuat..

VI. REFERENSI

[1] B. B. Kertawibowo, *Dinasti Raja Petapa I: Pangeran Cakrabuana: Sang Perintis Kerajaan Cirebon*. Bandung: Kiblat Buku Utama, 2007.

[2] *Morris, D., & Rollings, A. (2004). Game architecture and design: Learn the best practices for game design and programming. New Riders*.

[3] E. Z. Katie Salen Tekinbas, *Rules of Play: Game Design Fundamentals*. Massachusetts London, England: The MIT Press Cambridge., 2003.

[4] R. A. Krisdiawan, A. Permana, E. Darmawan, F. Nugraha, and A. Kriswandiyanto, "Implementation Dijkstra's Algorithm for Non-Players Characters in the Game Dark Lumber," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1933/1/012006.

[5] I. Ramadhan, "IMPLEMENTASI ALGORITMA JUMP POINT SEARCH PADA NPC MUSUH UNTUK MENGEJAR PEMAIN DALAM GAME LABIRIN."

[6] D. Harabor and A. Grastien, "The JPS Pathfinding System." [Online]. Available: www.aaai.org

[7] "There are several A* variant algorithms such as Iterative Deepening A* (IDA*) algorithm, Partial Expansion A* (PEA*), and Jump Point".

[8] R. A. Krisdiawan, "Implementasi Model Pengembangan Sistem Gdlc Dan Algoritma Linear Congruential Generator Pada Game Puzzle," 2018, [Online]. Available: https://journal.uniku.ac.id/index.php/ilkom/article/view/1634

[9] R. A. Krisdiawan, A. Fitriani, and H. Budianto, "Penerapan Algoritma Recursive Backtracking Sebagai Maze Generator Pada Game Labirin Aksara Sunda," *Media Jurnal Informatika*, vol. 14, no. 1, p. 31, Jun. 2022, doi: 10.35194/mji.v14i1.2326.