JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 12 No. 1, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v12i1.3830

RANCANG BANGUN *GAME MAZE* 3D "*LABYRINTH OF ISLAMIC KNOWLEDGE (LIKE)*" MENGGUNAKAN METODE GDLC DAN ALGORITMA *RECURSIVE* DFS

Vivin Octavia Cahyani[1*], Eka Mira Novita Subroto[2], Ahmad Faiz[3] , Fresy Nugroho[4] , Ahmad Fahmi Karami[5], Reza Putra Pradana[6]

[1,2,3,4,5,6]Teknik Informatika, Universitas Islam negeri Maulana Malik Ibrahim Malang Jalan Gajayana No. 50 Malang 65144 Telp: +62-341 551-354

Riwayat artikel:

Received: 11 Desember 2022
Accepted: 29 Desember 2023
Published: 1 Januari 2024

Keywords:
Game Maze;
Labiryrinth;
Depth First Search;
Game Developtment Life Cycle.

Corespondent Email:
210605110038@student.uin-malang.ac.id

Abstrak.*Game* ini ditujukan untuk anak-anak usia 7 tahun ke atas dengan fokus pada unsur-unsur agama Islam. Proses pengembangan *game* dimulai dari inisialisasi dengan tujuan menyusun labirin menggunakan algoritma *Recursive Depth First Search* (DFS) dengan penambahan langkah *shuffle* untuk menciptakan pola yang acak pada labirin. Tahap ini penting dalam membentuk struktur labirin yang berbeda setiap kali permainan dimulai. Selanjutnya, proses pembangunan *game* melalui tahap GDLC (*Game Development Life Cycle*) mencakup fase *pre-production, production*, dan *testing. Fase pre-production* mencakup gambaran umum *game*, inti *gameplay*, karakter-karakter utama, serta konsep desain lingkungan *game* yang berfokus pada nuansa Islami. *Fase production* menitik beratkan pada pembuatan lingkungan, labirin, karakter, dan elemen-elemen *visual* lainnya. Lingkungan dalam *game* direferensikan dari dunia nyata dengan gaya seni *realisme* untuk menciptakan lokasi ikonik seperti rumah tua dan sebagainya. Pengujian dilakukan menggunakan *Black Box Testing* yang menguji respons sistem terhadap berbagai input dari pengguna pada halaman-halaman utama *game* seperti *menu*, petunjuk, pengaturan, informasi tentang permainan, serta keluar dari permainan. Hasil pengujian sesuai dengan harapan dan menunjukkan *respons* yang *valid* sesuai dengan aksi yang dilakukan oleh pengguna.

Abstract. *Targeted at children aged 7 and above, the game focuses on Islamic concepts. The game development process begins with initialization, utilizing the Recursive Depth First Search (DFS) algorithm to generate maze structures, incorporating a shuffle step to create random patterns within the maze. This phase is crucial in creating a unique labyrinth structure each time the game starts. Subsequently, the game development follows the GDLC (Game Development Life Cycle), encompassing pre-production, production, and testing phases. The pre-production phase details the game's overview, core gameplay, main characters, and the conceptual design of the game environment with a focus on an Islamic ambiance. During production, the emphasis lies in creating the environment, labyrinth, characters, and other visual elements. The game's environment is inspired by real-world settings, adopting a realistic artistic style to establish iconic locations such as old houses. Black Box Testing is employed for testing, evaluating the system's response to various user inputs across different sections of the game, including menus, instructions, settings, game information, and exiting the game. The testing results align with expectations, demonstrating valid system responses corresponding to user actions.*

1. PENDAHULUAN

Penggunaan teknologi dalam pendidikan semakin merambah ke berbagai aspek pembelajaran. Salah satunya adalah melalui pengembangan permainan *edukatif* yang mampu menyajikan konten pendidikan dengan cara yang menarik dan *interaktif*. Dalam konteks ini, permainan komputer, khususnya *game* 3D, menjadi salah satu media yang menjanjikan dalam penyampaian materi pembelajaran. Dalam tulisan ini, akan dipaparkan tentang rancang bangun sebuah *game* 3D berjudul *"Labyrinth of Islamic Knowledge (LIKE)"* yang memiliki tujuan untuk menyajikan konsep pengetahuan Islam dalam format yang menghibur dan *edukatif*.

Game 3D "Labyrinth of Islamic Knowledge (LIKE)" diinisiasi sebagai upaya untuk memadukan kesenangan bermain *game* dengan pembelajaran konten keagamaan. Dengan memanfaatkan teknologi 3D, *game* ini dirancang untuk memberikan pengalaman eksplorasi labirin yang penuh dengan hikmah keislaman.

Game yang akan dikembangkan adalah *game maze* 3D, *single-player*, dengan map berbentuk *maze*. Pada *game* yang akan dikembangkan, *player* akan berusaha keluar dari *maze* dengan cara mengalahkan semua *monster*. Karena penempatann *enemy* ini akan secara otomatis masuk ke dalam ruangan *maze* nya maka saat mengenerate *maze* diberi *Artificial intelligence* menggunakan metode *Recursive Depth First Search Algoritm* (DFS). Algoritma ini memulai penelusuran dari satu simpul tertentu dan mengeksplorasi sejauh mungkin di setiap cabang sebelum melakukan *backtracking*.[1]

Games telah menjadi salah satu *aktivitas* yang mendominasi di berbagai kalangan. Seiring dengan kemajuan teknologi, *games* telah mengalami perkembangan yang signifikan, mulai dari *games* tradisional hingga games modern yang memanfaatkan teknologi tinggi. Dalam konteks ini, *games* tidak hanya dianggap sebagai bentuk hiburan semata, tetapi juga sebagai alat pembelajaran yang *efektif*. Penggunaan teknologi dalam pembelajaran, terutama melalui pengembangan *games edukatif* menjadi cara yang *inovatif* dalam menyampaikan materi pembelajaran.

Pada penelitian ini, peneliti akan berfokus pada rancang bangun sebuah *game* 3D berjudul *"Labyrinth of Islamic Knowledge (LIKE)"* yang memiliki tujuan untuk menyajikan konsep pengetahuan Islam dalam format yang menghibur dan *edukatif*. Dengan memanfaatkan teknologi 3D, *game* ini dirancang untuk memberikan pengalaman *eksplorasi* labirin dan pemahaman keislaman, terutama bagi anak-anak.

Langkah-langkah pengembangan *game* ini akan mengikuti *Game Development Life Cycle* (GDLC), sebuah *metode* yang menerapkan pendekatan *iteratif* dengan enam fase pengembangan.[2] Model GDLC diadopsi untuk memastikan bahwa *game* ini tidak hanya menarik, tetapi juga memenuhi standar kualitas yang diharapkan oleh pemain.

Melalui penelitian ini, kami berusaha menjawab pertanyaan mengenai bagaimana menerapkan model GDLC dalam pengembangan game platform yang menarik dan menyenangkan, serta merancang *game platform* 3D dengan menggunakan model perancangan sistem berbasis UML (*Unified Modeling Language*). Penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan permainan *edukatif* yang menggabungkan kesenangan bermain dengan nilai-nilai *edukatif*, khususnya dalam *konteks* pengetahuan Islam.

2. TINJAUAN PUSTAKA

2.1. *Unity*

Unity adalah *game engine* yang dibuat oleh *Unity Technologies Inc. Unity* digunakan untuk membuat *game* tiga dimensi, dua dimensi, *virtual reality*, bahkan *augmented reality*. *Unity* bisa untuk *games* PC dan *games Online*.[3] Untuk *games Online* diperlukan sebuah plugin, yaitu *Unity Web Player*, sama halnya dengan *Flash Player* pada *Browser* Kelebihan *software unity* dengan *software* pembuat *game* lainnya adalah mudah digunakan (*user friendly*) ditambah lagi *open source*. Dalam penerapannya, *unity* dapat

menggunakan berbagai bahasa pemograman, antara lain *JavaScript*, C#, dan Boo.[4]

2.2. *Maze Generator*

Maze adalah permainan labirin dengan jalan sempit yang berliku dimana pemain atau player harus menjelajahi labirin tersebut dan berjuang untuk menang sesuai dengan ketentuan yang ada[5]. *Maze Generator* merupakan salah satu teknik untuk membentuk *maze* yang berbeda setiap kali pemain memainkan *game* tersebut. Dengan begitu permainan akan lebih seru dan pemain merasa tertantang karena lingkungan labirin yang dinamis atau berubah-ubah.

2.3. *Recursive Depth First Search*

Algoritma *Depth First Search* (DFS) adalah teknik pencarian yang dimulai dari titik awal dan terus melanjutkan ke node anak paling kiri di setiap *level*. Cara kerjanya adalah dengan memasukkan node awal ke dalam sebuah tumpukan[6]. Selanjutnya, ambil *node* pertama dari tumpukan tersebut pada *level* paling atas. Jika *node* tersebut merupakan solusi dari pencarian, proses berakhir dan hasilnya dikembalikan. Namun, jika *node* tersebut bukan solusi, seluruh *node* yang berhubungan dengan *node* tersebut dimasukkan ke dalam tumpukan. Proses ini terus diulang hingga semua *node* telah diperiksa dan tumpukan kosong. Jika tidak ditemukan solusi, pencarian dimulai kembali dari *node* awal.

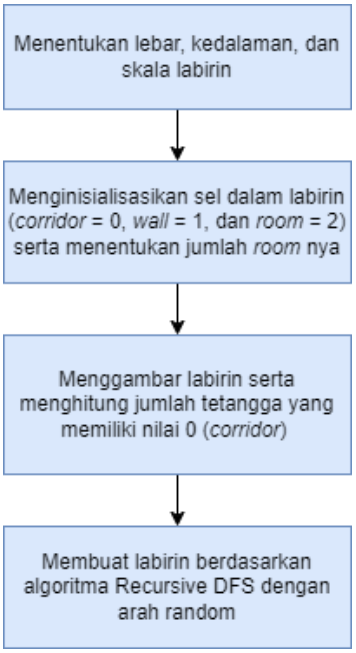
2.4. GDLC

GDLC *(Game Development Life Cycle)* adalah metode pengembangan game yang menggunakan pendekatan *iterative* dengan 6 *fase* pengembangan, antara lain *fase* inisialisasi/pembuatan konsep, *pre-production*, *production*, pengujian terdiri dari dua pengujian, yaitu *Alpha* secara internal dan *Beta* secara eksternal, serta terakhir adalah fase *release* [7]. GDLC merupakan *metode* yang sering dipilih dalam membuat game karena prosesnya yang runtut dan jelas.

3. METODE PENELITIAN

3.1. Alur Pembuatan Labirin

Alur pembuatan labirin dapat dilihat pada Gambar 1 di bawah ini.

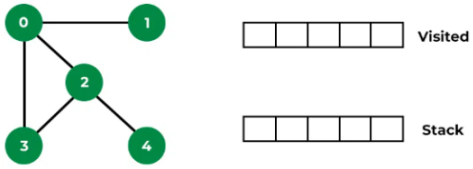


Gambar 1. Alur Skenario Pembuatan Labirin

3.2. Recursive Depth First Search Algorithm

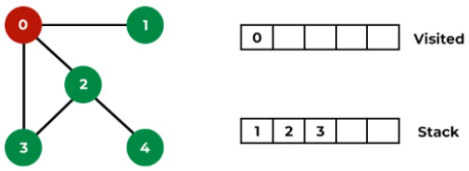
Berikut adalah tahapan dari algoritma *Recursive Depth First Search Algorithm* yang akan diterapkan untuk membuat labirin pada *game LIKE*.[8] Algoritma ini berbasis *LIFO (Last In First Out).*

a) Simpul akan dibagi menjadi dua yaitu *visited* (simpul yang telah dikunjungi) dan *stack* (simpul yang belum dikunjungi) dapat dilihat di Gambar 2.



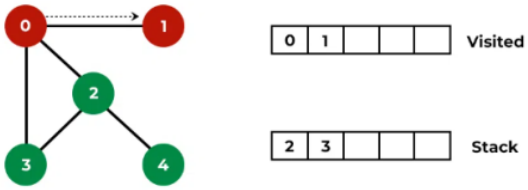
Gambar 2. Membagi Simpul Menjadi *Visited* Dan *Stack*

b) Menetapkan simpul awal sebagai akar. Pada Gambar 3. *Node* 0 ditetapkan sebagai simpul awal dan masuk ke dalam simpul *visited*, sedangkan *node* 1, 2, 3, 4 masuk ke dalam simpul *stack*.

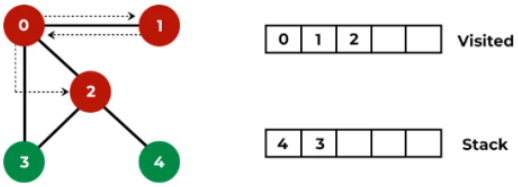


Gambar 3. Node 0 Sebagai Simpul Awal

c) Mengunjungi semua node tetangganya hingga sampai ke tujuan. Namun, apabila node yang dikunjungi sudah paling ujung *(leaves)* dan belum mencapai tujuan maka penelusuran akan kembali ke node sebelumnya seperti pada Gambar 2, dan 3

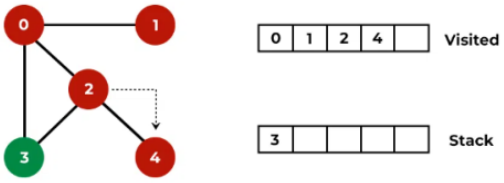


Gambar 4. Menelusuri Node 1

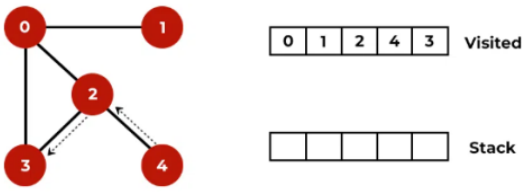


Gambar 5. Kembali Ke Node 0 Dan Telusuri Node 2

Pada Gambar 5. Penelusuran akan kembali ke node 0 karena node 1 adalah node paling ujung *(leaves)*. Kemudian penelusuran dilanjut ke node 2.



Gambar 5. Telusuri Node 4

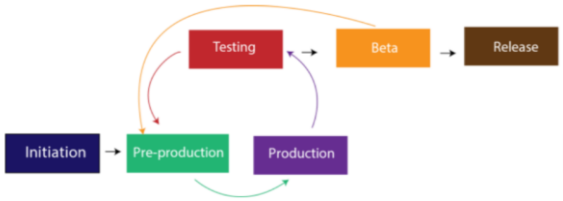


Gambar 6. Kembali Ke Node 2 Dan Telusuri Node 4

Pada Gambar 6. Algoritma *Recursive* DFS sudah selesai karena semua node telah dikunjungi. [9]

3.3. Game Development Life Cycle (GDLC)

Pada metode GDLC terdiri dari enam tahapan yang mengatur jalannya proses membuat Game, yaitu sebagai berikut.[10]



Gambar 7. Metode GDLC

1) *Initiation*

Tahap ini berisi konsep kasar dari game yang dibuat, misalnya judul, target pengguna, *genre*, tujuan *game*, dan pemilihan *platform*.

2) *Pre-Production*

Ini adalah tahap berlangsungnya proses pembuatan *Gameplay* biasanya dengan membuat *Game Design Document (GDD)* yang berisi konsep game secara keseluruhan.

3) *Production*

Tahap production adalah tahap implementasi dari GDD yang telah dibuat sebelumnya, seperti membuat asset yang dibutuhkan contohnya karakter pada game (NPC dan *Player*), *User Interface*, dan merancang *source code* agar sesuai dengan alur permainan.[11]

4) *Testing*

Tahapan ini berguna untuk menguji apakah masih terjadi bug atau tidak. Pada penelitian ini peneliti akan menggunakan metode *Blackbox Testing*.

4. HASIL DAN PEMBAHASAN

1. Alur Pembuatan Labirin

Pertama untuk membuat sebuah labirin dalam game ada beberapa metode salah satu metode

yang digunakan adalah *Recursive Depth Firs Search Algoritm* merupakan algoritma yang memulai pencarian dari suatu simpul tertentu, kamudian menjelajahi sejauh mungkin di setiap cabang sebelum melakukan backtracking. [1]Berikut yang akan dilakukan pertama adalah menginisialisasi kedalaman, lebar, dan skala pada labirinya, seperti pada Gambar 8. Inisialisasi berikut ini:

```
public class MazeLogic : MonoBehaviour
{
    public int width = 30; //x length
    public int depth = 30; //z length
    public int scale = 6;
    public GameObject character;
```

Gambar 8. Inisialisai

Selanjutmya, setelah menginisialisasi, akan dibuat berapa jumlah room yang diinginkan, seperti pada Gambar 9. Jumlah *Room* berikut ini:

```
1 reference
public virtual void AddRooms(int count, int minSIze, int maxSize)
{
    for (int c = 0; c < count; c++)
    {
        int startX = Random.Range(3, width - 3);
        int startz = Random.Range(3, depth - 3);
        int roomWidth = Random.Range(minSIze, maxSize);
        int roomDepth = Random.Range(minSIze, maxSize);

        for (int x = startX; x < width - 3 && x < startX + roomWidth; x++)
        {
            for (int z = startz; z < depth - 3 && z < startz + roomDepth; z++)
            {
                map[x, z] = 2;
            }
        }
    }
}
```

Gambar 9. Jumlah Room

Setelah semuanya selesai maka akan Digambar labirinya sesuai penjelasan Gambar 10. Pembuatan Labirin:

```
1 reference
void DrawMaps()
{
    for (int z = 0; z < depth; z++)
    {
        for (int x = 0; x < width; x++)
        {
            if (map[x, z] == 1)
            {
                Vector3 pos = new Vector3(x * scale, 0, z * scale);
                GameObject wall = Instantiate(Cube[Random.Range(0, Cube.Count)], pos, Quaternion.identity);
                wall.transform.localScale = new Vector3(scale, scale, scale);
                wall.transform.position = pos;
            }
        }
    }
}
```

Gambar 10. Pembuatan Labirin

Selanjutnya akan di implementasikan *Recursive Depth Firs Search Algoritm* pada algoritma tersebut akan dimasukkan *Shuffle* untuk membentuk pola arah yang random, supaya pola yang terbentuk tidak monoton.

```
Unity Script (7 asset references) | 0 references
public class RecursiveDFS : MazeLogic
{
    public List<MapLocation> directions = new List<MapLocation>()
    {
        new MapLocation(1,0),
        new MapLocation(0,1),
        new MapLocation(-1,0),
        new MapLocation(0,-1)
    };
    2 references
    public override void GenerateMaps()
    {
        Generate(5, 5);
    }
```

Gambar 11. Recursive DFS

Pada script diatas akan ditambahkan *script* baru untuk memasukkan *shuffle*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public static class Extentions
{
    private static System.Random rng = new System.Random();

    1 reference
    public static void Shuffle<T>(this IList<T> list)
    {
        int n = list.Count;
        while (n > 1)
        {
            n--;
            int k = rng.Next(n + 1);
            T value = list[k];
            list[k] = list[n];
            list[n] = value;
        }
    }
}
```

Gambar 12. Shuffle

Menambahkan *methodGenerateMaps()* untuk membentuk labirinnya

```
void Generate(int x, int z)
{
    if (CountSquareNeighbours(x, z) >= 2) return;
    map[x, z] = 0;

    directions.Shuffle();

    Generate(x + directions[0].x, z + directions[0].z);
    Generate(x + directions[1].x, z + directions[1].z);
    Generate(x + directions[2].x, z + directions[2].z);
    Generate(x + directions[3].x, z + directions[3].z);

}
```

Gambar 13. Generate

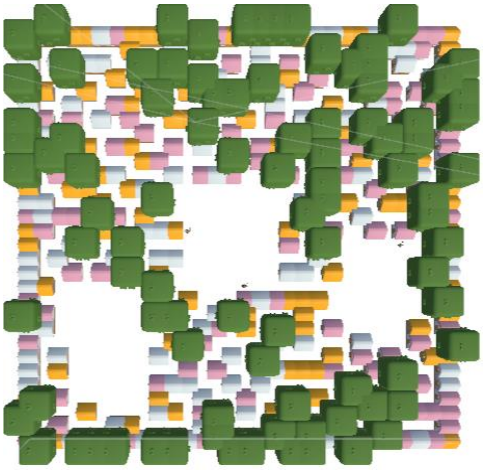
Penjelasan alur program metode rekrusif algoritma DFS [9] untuk membuat labirin. Parameter ‘x’ dan ‘z’ merepresentasikan koordinat, melakukan beberapa Langkah berikut:

- Memeriksa apakah jumlah tetangga persegi saat ini yang sudah diisi dengan nilai non-0 sudah mencapai atau melebihi 2. Jika iya, maka metode berhenti dan tidak melanjutkan pengisian labirin di koordinat ini.

- Jika belum, metode akan mengatur nilai peta koordinat (x, y) menjadi 0(artinya persegi diisi).
- Kemudian, arah acak *('direction')* diacak untuk memilih urutan acak langkah berikutnya.
- Metode akan melakukan pemanggilan dirinya sendiri *(Generate)* untuk keempat arah yang telah diacak, yaitu melalui Generate(x + directions [i].x, z + directions [i].z) di mana i adalah *indeks* dari arah yang dipilih.

  Kode ini akan terus memanggil dirinya sendiri secara rekursif untuk membuat jalur-jalur dalam labirin sampai kondisi berhenti terpenuhi (yaitu, jumlah tetangga dari suatu titik yang sudah diisi sudah mencapai atau melebihi 2). Algoritma ini akan secara acak mengisi persegi pada peta, menciptakan struktur labirin yang berbeda setiap kali dijalankan.

  Berikut hasil dari labirin yang telah dibuat:



Gambar 14 *labyrinth*

2. Alur Pembangunan *Game*

Berikut hasil dari implementasi *metode* GDLC yang digunakan:

a. Initiation

Pada *initiation* ini berisi gambaran *Game* yang akan dibuat judul *game* nya *Labyrinth of Islamic Knowledge* (LIKE) *game* ini bertema Islami, dan pembelajaran tentang keagamaan, unsur keagamaannya terletak pada soal-soal pembelajaran yang ada unsur agamanya. Target *audience* anak-anak berusia 7 tahun hingga umum. Tujuan akhir dari permainan ini adalah untuk mengumpulkan bonus koin. Kunci tersebut didapatkan jika berhasil mengalahkan semua monster.

b. Pre-Production

- ***Game Overview***

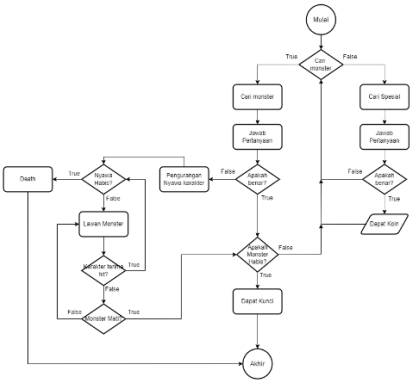
***Title**: Labyrinth of Islamic Knowledge (LIKE)*

***Genre**: islamic, maze, and education*

***Target Audience* :** Anak - anak berusia 7 tahun hingga umum

- ***Core Gameplay***

Labyrinth of Islamic Knowledge (LIKE) adalah game 3D bernuansa Islami dimana pemain akan bermain sebagai karakter utama yang harus menjelajahi labirin. Dalam labirin ini, pemain akan mencari dan berhadapan langsung dengan musuh (monster) untuk mengalahkannya dengan cara menjawab kuis-kuis Islami dengan benar. Jika pemain berhasil menjawab kuis dengan benar maka musuh akan mati, sebaliknya jika jawaban salah maka pemain tersebut akan kekurangan nyawa dan masuk ke *scene* pertarungan antara monster dan player. Musuh akan mati jika kalah dalam pertarungan lalu player akan kekurangan nyawa apabila terkena serangan dan akan mati jika nyawanya habis. Tujuan akhir permainan ini adalah untuk mengumpulkan kunci yang digunakan sebagai syarat masuk ke babak berikutnya dan mendapatkan bonus koin. Kunci tersebut didapatkan jika berhasil mengalahkan semua monster.



Gambar 15. Alur *game*

- **Karakter dan Tujuan**

Player: Tujuan utama *player* adalah mendapatkan kunci untuk masuk ke level berikutnya dengan melawan *monster* serta mendapat koin untu k mengganti skin pada karakter.

***Monster*:** Tujuan utama *monster* adalah sebagai musuh yang harus dikalahkan oleh *player* dengan cara menjawab pertanyaan-

pertanyaan bertema islami secara benar atau berhasil menang dalam pertarungan.

- **Konsep Desain *Environment***

Map/worlds references, Realism Art Style (PBR) Iconic map location (Maps are designed to reflect-real-world environments like old house, dll)

Referensi Desain Reference

Gambar 16. *Environment*

Player

Gambar 17. *Player*

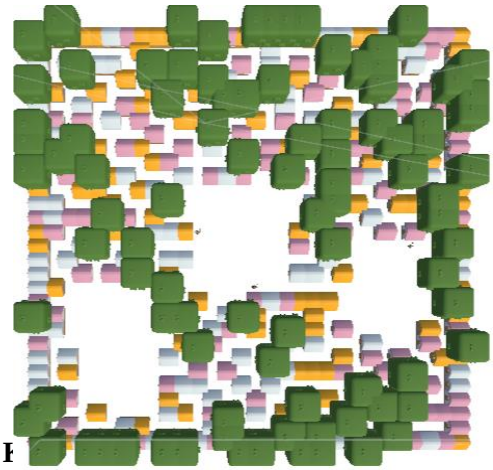
Monster

Gambar 18. *Enemy*

c. Production Environment

Gambar 19. Mencari *Enemy*

Labyrinth



Gambar 20. *Labyrinth*

d. Testing

Testing dilakukan dengan *Black Box Testing.* Menguji pada *input* dan *output* dari *system.* Hasil pengujian ini dijelaskan dalam Tabel 1. *Pengujian Black Box Halaman Menu Utama*

Gambar 21. *Main Memu Testing*

Pengujian *Black Box* dari halaman menu utama				
Tombol	Aksi Pemain	Reaksi Sistem	Harapan	Status
Menu Main	Pemain menekan tombol *play*	Sistem menampilkan menu pengenalan *game*	Sesuai harapan	*Valid*
Menu Petunjuk	Pemain menekan tombol petunjuk cara memainkan *game*	Sistem menampilkan halaman menu petunjuk permainan	Sesuai harapan	Valid
Menu Settting	Pemain menekan tombol menu setting	Sistem akan menampilkan halaman menu setting	Sesuai harapan	Valid
Menu About	Pemain menekan tombol about	System menampilkan halaman about *game*	Sesuai harapan	Valid
Menu Keluar	Pemain menekan tombol keluar	Permainan berhenti	Sesuai harapan	Valid

Tabel 1. Pengujian

5. KESIMPULAN

Dari Penelitian yang telah dilakukan bahwa dengan menggunakan algoritma *Depth First Search* (DFS) untuk membuat *labyrinth* dan juga algoritma *Game Developtment Life Cycle* sangat membantu dalam membangun alur dari *game* ini, maka dapat disimpulkan bahwa:

1. Dengan memanfaatkan kedua algoritma tersebut penulis dapat membangun *game* ritma tersebut penulis dapat membangun *game* yang dinamis.
2. *Model* pengembangan *game* menggunakan algoritma *GDLC* sangat membantu dalam membuat sebuah *game*.
3. Dengan dimasukkannya soal-soal bernuansa Islami dapat memberikan *edukasi* yang baik untuk anak-anak.

UCAPAN TERIMA KASIH

Ucapan terima kasih kami sampaikan kepada pihak-pihak yang telah ikut berkontribusi dalam penelitian ini dari proses penulisan hingga proses submit, orang tua kami, dosen pengampu matakuliah multimedia dan game baik praktikum maupun teori, rekan-rekan kami, dan masih banyak lagi yang tidak bisa kami sebutkan satu persatu. Selain itu kami juga mengucapkan terima kasih kepada pihak-pihak dari jurnal Informatika dan Teknik Elektro Terapan yang mengevaluasi dan membaca penulisan kami. Semoga karya ini dapat menjadi referensi yang bermanfaat bagi para pembaca serta bagi Masyarakat dalam arti luas.

DAFTAR PUSTAKA

[1] E. Setiadharma, L. Husniah, and A. S. Kholimi, “Algoritma Maze Generator Recursive Backtracking Untuk Membuat Prosedural Labirin Pada Game Petualangan Labirin 3D,” *J. Repos.*, vol. 2, no. 3, pp. 373–384, 2020, doi: 10.22219/repositor.v2i3.397.

[2] L. Husniah, B. F. Pratama, and H. Wibowo, “Gamification And GDLC (Game Development Life Cycle) Application For Designing The Sumbawa Folklore Game ”The Legend Of Tanjung Menangis (Crying Cape)”,” *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 3, no. 4, pp. 351–358, 2018, doi: 10.22219/kinetik.v3i4.721.

[3] A. Hadisopiyan, C. D. Suhendra, and P. H. Rantelinggi, “Membuat Game 3d Survival Horror ‘Suanggi Survival Papua’ Berbasis Desktop Menggunakan Unity,” *INFORMAL Informatics J.*, vol. 5, no. 3, p. 96, 2020, doi: 10.19184/isj.v5i3.21235.

[4] L. S. Mongi, A. S. M. Lumenta, and A. M. Sambul, “Rancang Bangun Game Adventure of Unsrat Menggunakan Game Engine Unity,” *J. Tek. Inform.*, vol. 13, no.

1, 2018, doi: 10.35793/jti.13.1.2018.20191.
[5] A. N. Putri, "Optimasi Alghoritma Breadth First Search Pada Game Engine 3D Third Person Shooter Maze Berbasis Agen Cerdas Android," *J. Transform.*, vol. 14, no. 1, p. 50, 2016, doi: 10.26623/transformatika.v14i1.349.
[6] E. G. Masala, I. P. Saputro, and R. T. B. Turang, "Perbandingan Algoritma Breadth First Search Dan Depth First Search Pada Game Mummy Maze Deluxe," *J. Ilm. Realt.*, vol. 14, no. 2, pp. 143–148, 2018, doi: 10.52159/realtech.v14i2.46.
[7] R. A. Krisdiawan and Rio, "Penerapan Model Pengembangan Game Gdlc (Game Development Life Cycle)Dalam Membangun Game Platform Berbasis Mobile," *Teknokom*, vol. 2, no. 1, pp. 31–40, 2019.
[8] O. Pribadi, "Maze Generator Dengan Menggunakan Algoritma Depth-First-Search," *J. TIMES*, vol. 4, no. 1, pp. 1–5, 2015, [Online]. Available: http://www.stmik-time.ac.id/ejournal/index.php/jurnalTIMES/article/view/213
[9] B. V. Indriyono, N. Pamungkas, Z. Pratama, E. Mintorini, I. Dimentieva, and P. Mellati, "Comparative Analysis of the Performance Testing Results of the Backtracking and Genetics Algorithm in Solving Sudoku Games," vol. 5, no. 1, pp. 29–35, 2023.
[10] M. R. Siregar and N. Nelmiawati, "Game 3D 'Lawan Narkoba' Menggunakan Metode Game Development Life Cycle (GDLC)," *J. Appl. Multimed. Netw.*, vol. 4, no. 1, pp. 24–31, 2020, doi: 10.30871/jamn.v4i1.1634.
[11] R. A. Krisdiawan, "Implementasi Model Pengembangan Sistem Gdlc Dan Algoritma Linear Congruential Generator Pada Game Puzzle," *Nuansa Inform.*, vol. 12, no. 2, pp. 1–9, 2018, [Online]. Available: https://journal.uniku.ac.id/index.php/ilkom/article/view/1634/1211