DAFTAR PUSTAKA

Achmad, M. H., Findari, W. S., Ann, N. Q., Pebrianti, D., & Daud, M. R. (2016). Stereo camera—based 3d object reconstruction utilizing semi-global matching algorithm. *2016 2nd International Conference on Science and Technology-Computer (ICST)*.

Apperley, T. H. (2006). Genres and Game Studies: Towards a Critical Approach to Video Game Genres. *Simulation & Gaming 37.1*, 6-23.

Barnouti, N. H., Al-Dabbagh, S. S., & Naser, M. A. (2016). Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm. *Journal of Computer and Communications, 2016, 4, 15-25*.

Cahyadi, M. A., Widhiarso, W., & & Yohannes, Y. (2017). *Perbandingan Algoritma A*, Dijkstra dan Floyd Warshall Untuk Menentukan Jalur Terpendek Pada Permainan "Bacteria Defense".* Palembang: Jurusan Teknik Informatika, STMIK GI MDP.

Fajri, C. (2012). Tantangan Industri Kreatif-Game Online di Indonesia. *Jurnal ASPIKOM 1.5*, 443-454.

Funge, J. D. (2004). Artificial intelligence for computer games: an introduction. *CRC Press*.

Graham, R., McCabe, H., & Sheridan, S. (2003). Pathfinding in Computer Game. *The ITB Journal*.

Guruji, A. K., Agarwal, H., & & Parsediya, D. K. (2016). Time-efficient A* algorithm for robot path planning. *Procedia Technology*, 144-149.

Harsadi, P. (2016). Pathfinding pada Lingkungan Statis Berdasarkan Artificial Potential Field Dengan Flocking Behavior Untuk Non-Player Character Follower Pada Game. *Jurnal Ilmiah SINUS*.

Lasseter, J. (1987). Principle of Traditional Animation Applied To 3D Computer Animation. *Computer Graphics, Volume 21, Number 4.*

Mathew, G. E. (2015). Direction Based Heuristic For Pathfinding In Video Game. *Procedia Computer Science*.

Millington, I., & Funge, J. (2009). *Artificial Intelligence For Games, Second Edition.* Burlington, USA: Morgan Kaufmann Publishers.

Pramono, A. (2015). Algoritma Pathfinding A* pada Game RPG Tanaman Higienis. *Jurnal Edukasi dan Penelitian Informatika (JEPIN) Vol. 1, No. 2, (2015)*, 76.

Reese, B., & Stout, B. (1999). Finding a Pathfinder. *AAAI 99 Spring Symposium on Artificial Intelligence and Computer Games.*

Rifai, W. A. (2015). *Pengembangan Game Edukasi Lingkungan Berbasis Android.* Yogyakarta: Fakultas Teknik Universitas Negeri Yogyakarta.

Sanny, L. (2018). *Potensi Industri Game Indonesia.* Jakarta: Binus University Business School.

Setiawan, A., Harsadi, P., & Siswanti, S. (2019). Penerapan Pathfinding Menggunakan Algoritma A* pada Non Player Character (NPC) Di Game. *Jurnal Ilmiah Sinus (JIS) Vol : 17, No. 2*.

Sharlin, E., Watson, B., Kitamura, Y., Rorabeck, D., Lederer, R., Sutphen, S., & Kishino, F. (2004). he Tangible Pathfinder Design of a Wayfinding Trainer for the Visually Impaired. *Proc. Graphics Interface*.

Shi, X. C., & Hao. (2011). A*-based Pathfinding in Modern Computer Games. *IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011*.

Silver, D. (2005). Cooperative Pathfinding. *AIIDE 1*, 117-122.

Stout, B. (1996). Smart moves: Intelligent pathfinding. *Game developer magazine*, 28-35.

Van Gumster, J. (2020). *Blender for dummies.* John Wiley & Sons.

Wafiqurrahman, N. (2015). Penerapan Algoritma a* (A-Star) Untuk Penentuan Rute Terpendek Game Pramuka Berbasis Android. *Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim*.

Yannakakis, G. N., & Togelius, J. (2014). A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games 7.4*.

LAMPIRAN

Lampiran A. Source Code

EnemyController.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Diagnostics;
using UnityEngine;
using UnityEngine.AI;
using Debug = UnityEngine.Debug;

public class EnemyController : MonoBehaviour {

    //Deklarasi variabel player
    public GameObject player;

    //Deklarasi navmesh agent bagi karakter NPC
    public NavMeshAgent enemy;

    //Deklarasi animasi bagi karakter NPC
    public Animator anim;

    //Deklarasi variabel untuk penghitungan waktu komputasi, waktu running, dan status
player
    public Stopwatch computingTimer,runningTimer;
    public TimeSpan computing_timeSpan,running_timeSpan;
    public int mean,tick,minute,second,milisecond;
    public String status;

    //Deklarasi variabel untuk menampilkan node jalur
    private Vector3 path1;
    public GameObject cube,cubePath;
    public int pathDifference = 0;

    //Inisiasi awal program
    void Start () {

        //Pemanggilan variabel navmesh agent
        enemy = GetComponent<NavMeshAgent>();
        enemy.updateRotation = false;

        //Pemanggilan animator untuk animasi karakter NPC
        anim = GetComponent<Animator>();

        //Inisiasi waktu mulai penghitungan komputasi navmesh dan running program
        Debug.Log("Waktu komputasi mulai");
        computingTimer = Stopwatch.StartNew();
        runningTimer = Stopwatch.StartNew();

        //Inisiasi penghitungan waktu rata-rata komputasi navmesh
        mean = 0;
        tick = 0;
    }
```

```
//Update dijalankan setiap frame pada game (30 frame per detik)
        void Update () {

        //Pengkondisian jika karakter player ditemukan
        if (player != null)
        {
            status = ("player ditemukan");

            //Pemanggilan proses penghitungan jalur berdasarkan posisi NPC, posisi
player dan struktur navmesh yang telah dipanggil
            enemy.SetDestination(player.transform.position);


            //Pengkondisian untuk mengatur rotasi NPC sesuai dengan jalur
            if (enemy.velocity.sqrMagnitude > Mathf.Epsilon)
            {
                transform.rotation =
Quaternion.LookRotation(enemy.velocity.normalized);
            }

            // Pengkondisian untuk menjalankan animasi jalan jika jarak NPC dan jarak
player dibawah dengan jarak yang ditentukan
            if (enemy.remainingDistance < 8)
            {
                anim.SetBool("isWalking", false);
            }
            else
            {
                anim.SetBool("isWalking", true);
            }
        }

        else
        {
            status = ("player tidak ditemukan");
        }

        //Penambahan waktu tick/update setiap kelas update dijalankan
        tick++;

        //Menginisaisi method untuk menampilkan node dalam bentuk kubus
        pathViewer();

        //Penghentian waktu komputasi navmesh
        computingTimer.Stop();
        computing_timeSpan = computingTimer.Elapsed;
        Debug.Log(" Waktu komputasi = " + computing_timeSpan.Minutes + " : " +
computing_timeSpan.Seconds + " : " + computing_timeSpan.Milliseconds);
        //Penambahan waktu komputasi pada variabel mean untuk penghitungan waktu rata-
rata komputasi
        computingTimer = Stopwatch.StartNew();
        mean += computing_timeSpan.Milliseconds;
    }
//Method untuk menampilkan node jalur NPC dan menampilkan koorninat node tersebut
dalam bentuk kubus
    void pathViewer()
    {

        //Mengambil panjang array node jalur
        int pathLegth = enemy.path.corners.Length;

        //Mendeteksi jika jumlah array jalur berubah
        if (pathDifference != pathLegth)
        {

            //Perungalan untuk memunculkan dan menempatkan kubus sesuai koordinat node
            for (int j = 0; j < pathLegth; j++)
            {
                cubePath = Instantiate(cube) as GameObject;
                cubePath.transform.position = enemy.path.corners[j];
```

```
/*Penghapusan objek kubus pada node awal dan node akhir agar tidak terjadi pemunculan
kubus secara
                berkelanjutan ketika karakter NPC dan player bergerak*/
                if (j==0 || j==pathLegth-1)
                {
                    Destroy(cubePath);
                }

                //Debug.log untuk menampiulkan koordinat node
                /*Vector3 path1 = enemy.path.corners[j];
                Debug.Log(path1);*/
            }
        }
        else
        {

        }
    }

    //Kelas dijalankan ketika program dihentikan
    private void OnApplicationQuit()
    {
        //Penghentian waktu running program
        runningTimer.Stop();
        running_timeSpan = runningTimer.Elapsed;

        //Penghitungan waktu rata-rata komputasi navmesh
        mean /= tick;

        Debug.Log("Program dihentikan");
        Debug.Log(" Waktu running program = " + running_timeSpan.Minutes + " Menit, "
+ running_timeSpan.Seconds + " Detik, " + running_timeSpan.Milliseconds + " Milidetik
");
        Debug.Log("Rata-rata waktu komputasi navmesh = " + mean + " mili detik");
        Debug.Log(" Status player : " + status);
    }
}
```

*Player*Controller.cs

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

public class PlayerController : MonoBehaviour {

    //Deklarasi variabel kecepatan jalan dan rotasi player
    public float moveSpeed;
    public float TurnSpeed;

    //Deklarasi button movement bagi player
    Vector2 input;

    //Deklarasi sudut rotasi player
    Quaternion targetRotation;
    float angle;

    //Deklarasi animasi bagi karakter player
    public Animator anim;

//Inisiasi awal program
    void Start ()
    {
        //Pemanggilan animator untuk animasi karakter player
        anim = GetComponent<Animator>();
    }

     //Kelas untuk mengambil nilai x dan y
    void GetInput()
    {
        //Pengambilan nilai x dan y sesuai dengan tombol yang ditekan oleh user (W, A,
S, D) atau (Up, Down, Left, Right)
        input.x = Input.GetAxisRaw("Horizontal");
        input.y = Input.GetAxisRaw("Vertical"); }
    }

           //Kelas untuk menghitung arah pergerakan player sesuai dengan nilai x dan y
    void CalculateDirection()
    {
        angle = Mathf.Atan2(input.x, input.y);
        angle = Mathf.Rad2Deg * angle;
    }

    //Kelas untuk menghitung arah rotasi dan kecepatan rotasi
    void Rotate()
    {
        targetRotation = Quaternion.Euler(0, angle, 0);
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation,
TurnSpeed * Time.deltaTime);

        //Inisiasi animasi jalan pada karakter player
        anim.SetBool("isWalking", true);
    }

    //Kelas untuk menghitung pergerakan karakter player
    void Move()
    {
        transform.position += transform.forward * moveSpeed * Time.deltaTime;

        //Inisiasi animasi jalan pada karakter player
        anim.SetBool("isWalking", true);
    }
```

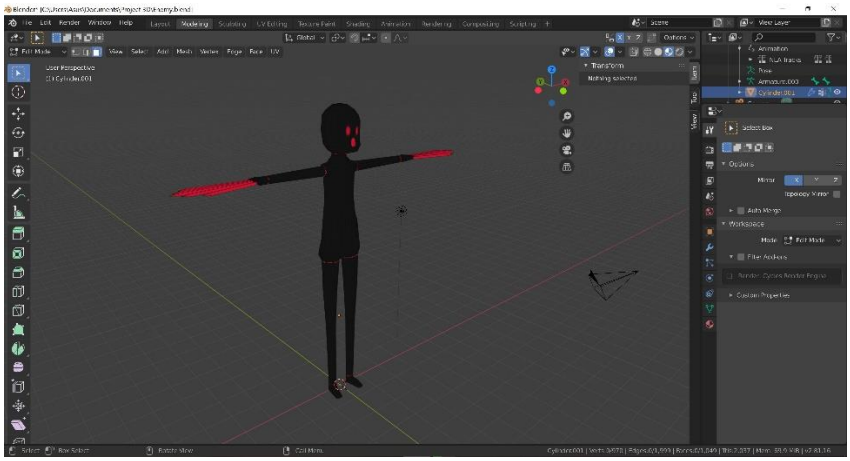
```
    //Update dijalankan setiap frame pada game (30 frame per detik)
    void Update ()
    {
        //Pemanggilan kelas GetInput
        GetInput();

        //Pengkondisian jika karakter diam maka animasi jalan akan dihentikan
        if (Mathf.Abs(input.x) < 1 && Mathf.Abs(input.y) < 1)
        {
            anim.SetBool("isWalking", false);
            return;
        }

        //Pemanggilan kelas CalculateDirection,Rotate dan Move
        CalculateDirection();
        Rotate();
        Move();
    }
}
```

Lampiran B. Model 3 Dimensi

Karakter *Player*

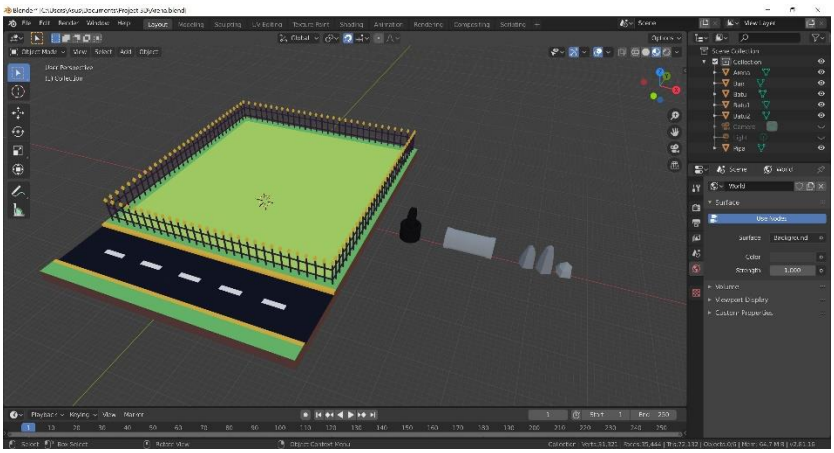
Gambar B.1. Model karakter *Player*

Karakter NPC



Gambar B.2. Model karakter NPC

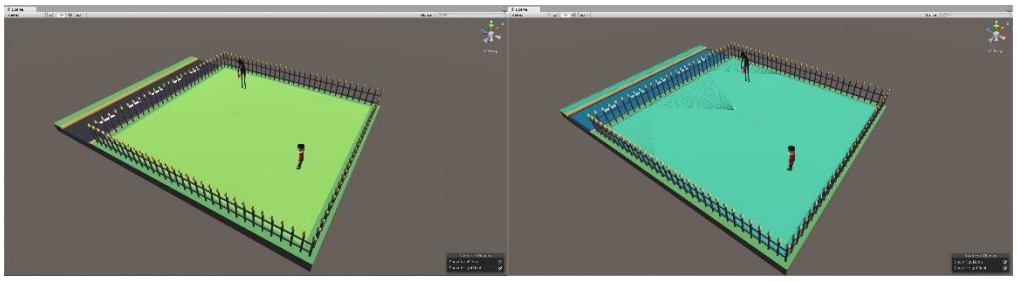
Arena dan Rintangan



Gambar B.3. Model arena dan rintangan

Lampiran C. Scene Game

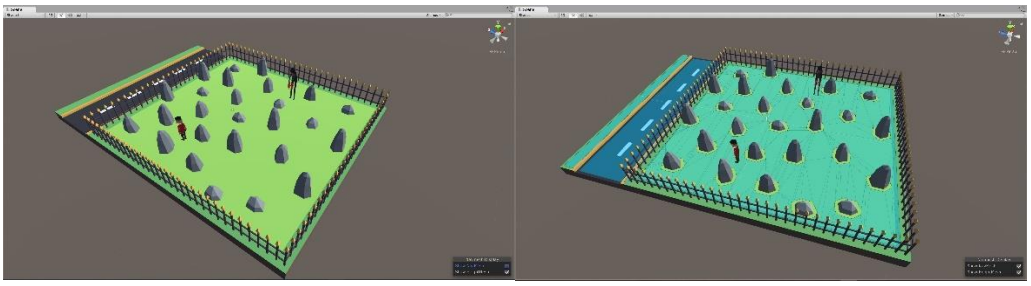
Observer Scene



Gambar C.4. *Observer scene* beserta *navmesh*nya

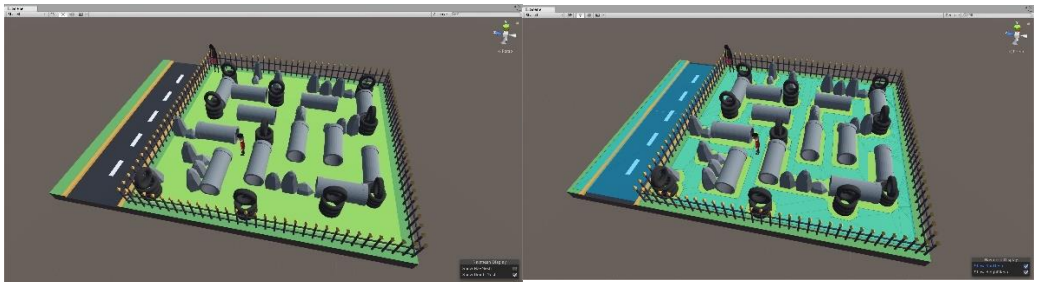
.

Simple Scene



Gambar C.5. *Simple scene* beserta *navmesh*nya

Complex Scene



Gambar C.6. *Complex scene* beserta *navmesh*nya