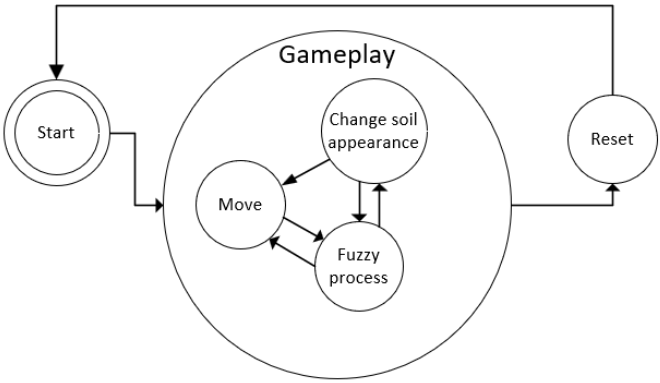
BAB IV

HASIL DAN PEMBAHASAN

4.1 Skenario Permainan

4.1.2 FSM *Gameplay* (*Finite State Machine*)



Gambar 3.3 FSM *Gameplay*

Menekan tombol WASD pada keyboard memungkinkan pemain untuk bergerak bebas saat permainan baru dimulai, lalu Ketika traktor bergerak menuju tanah sawah yang akan di bajak nanti nya 3 *variable* yang telah di *set* oleh *player* akan di proses dengan menggunakan algoritma *fuzzy* .

setelah meperoleh keluaran *porositas* nanti nya akan di lanjutkan dengan perubahan penampilan pada tanah apakah sedikit gembur ,gembur,atau sangat gembur. Lalu jika *player* ingin mengulangi kembali *player* dapat menekan *reset* untuk mengulangi nya kembali.

Skenario dari "*Serious Game* Pengolahan Tanah Menggunakan Bajak Singkal Berbasis *Fuzzy*" ini memiliki konsep tentang mensimulasikan yang diperuntukan untuk orang-orang yang membutuhkan pembelajaran tentang apa saja faktor apa saja yang menyebabkan tanah menjadi tidak gembur, gembur dan sangat gembur. Di dalam game/permainan ini berlatar lokasi di sawah. Dalam permainan ini *Player* memeran kan sebuah traktor tangan yang nanti nya akan berjalan di sawah.

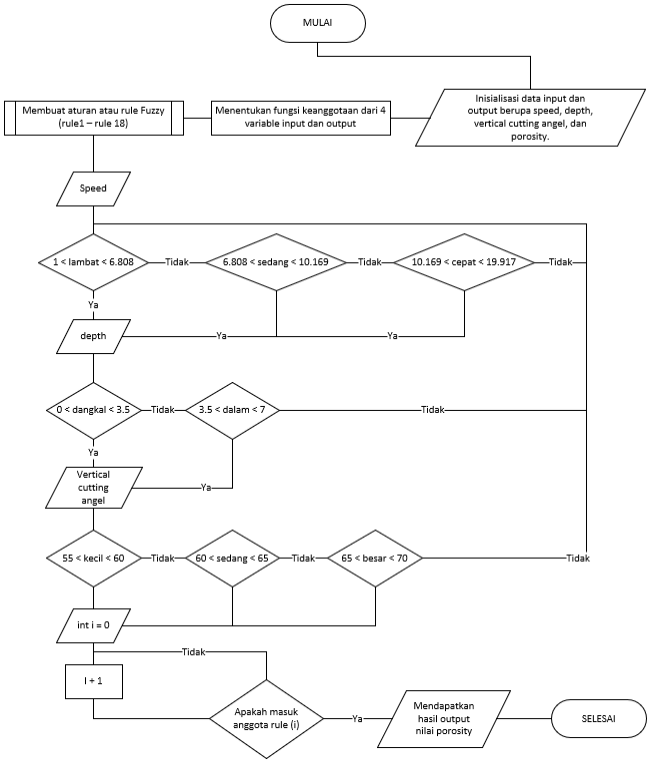
Dan nantinya Player dapat mengontrol 3 *variable* yaitu *Speed* (kecepatan), *Depth* (kedalaman), *vertical cutting angel*(titik potong vertikal).

Permainan ini nantinya akan menghasilkan output tingkat kegemburan tanah, berdasarkan 3 *variable* yang telah di tentukan di dalam game oleh Player. Tingkat kegemburan tanah ditentukan oleh 3 *variable* Input yang diproses oleh algoritma *Fuzzy* yang nantinya akan menentukan tingkat

kegemburan dari tanah tersebut. Nantinya hasil dari gembur atau tidak gembur, gembur dan sangat gembur akan menampilkan perubahan pada tanah yang telah dilewati oleh traktor. Sehingga *Player* dapat melihat perubahan pada tanah secara langsung, tingkat kegemburan akan ditampilkan pada uji *game*.

4.2 Desain dan *flowchart*

4.2.1 *Flowchart Fuzzy Logic*



Gambar 4.1 *Flowchart Fuzzy logic*

Flowchart alur (Gambar 4.1) menunjukkan dari awal proses menggunakan *Fuzzy logic* yaitu pengumpulan data *Input* dan *output*.

Variabel Input

- Kecepatan (*Speed*) : Mengukur tingkat kecepatan bajak singkal.

- Kedalaman Singkal (*Depth*): Parameter ini menentukan seberapa dalam alat bajak memasuki tanah.

- *Vertical Cutting Angle* : sudut mata pisau bajak singkal.

Variabel Output

Kegemburan Tanah: Berdasarkan *Input*, output akan menentukan apakah kondisi tanah ideal untuk diolah atau membutuhkan perubahan parameter.

Algoritma *Fuzzy* Logic

Fuzzy Logic yang diimplementasikan dalam game ini didasarkan pada teori *Fuzzy* set yang diperkenalkan oleh Zadeh pada tahun 1965 (Zadeh, 1965). Algoritma ini terdiri dari beberapa tahapan:

Fuzzification: Mengubah *variabel Input* menjadi tingkat keanggotaan *Fuzzy*. Misalnya, kadar air tanah bisa menjadi "Rendah", "Sedang", atau "Tinggi".

Inference: Menggunakan aturan *Fuzzy* untuk memproses *Input.*

Defuzzification: Mengubah *variable* output *Fuzzy* kembali menjadi angka konkret untuk tindakan selanjutnya dalam *game*.

4.2.2 *Rules*/Aturan *Fuzzy*

1. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Sedikit Gembur."

2. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah "Sedikit Gembur."

3. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Sedikit Gembur."

4. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Sedikit Gembur."

5. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah "Sedikit Gembur."

6. Jika *Speed* adalah "Lambat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Gembur."

7. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Sedikit Gembur."

8. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah "Gembur."

9. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Gembur."

10. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Sedikit Gembur."

11. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah

"Gembur."

12. Jika *Speed* adalah "Sedang" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Sangat Gembur."

13. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Sedikit Gembur."

14. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah "Sedikit Gembur."

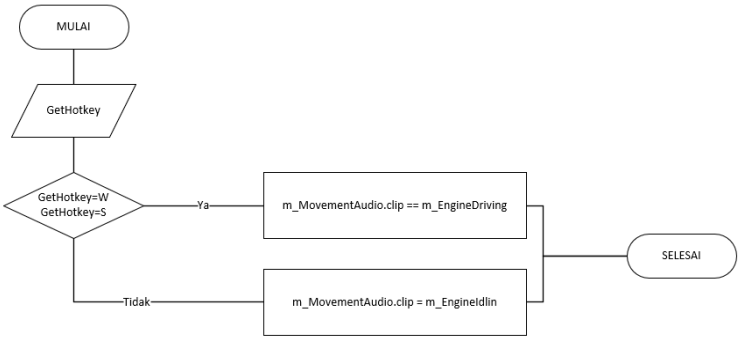
15. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dangkal" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Gembur."

16. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Kecil," maka *Porosity* adalah "Gembur."

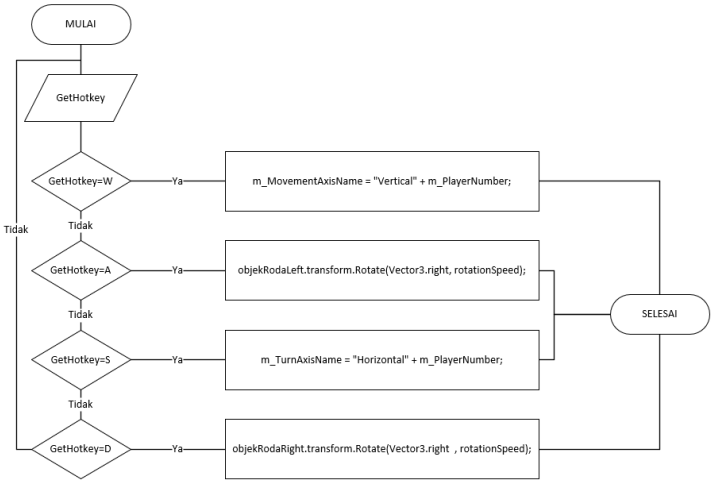
17. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Sedang," maka *Porosity* adalah "Gembur."

18. Jika *Speed* adalah "Cepat" dan *Depth* adalah "Dalam" dan *Vertical Cutting Angle* adalah "Besar," maka *Porosity* adalah "Sangat Gembur."

4.2.3 *Flowchart movement/*pergerakan



Gambar 4.2.1 *Flowchart movement/*pergerakan dan audio



Gambar 4.2.2 *Flowchart movement/*pergerakan dan audio

Pada gambar 4.2.2 merupakan *Flowchart* yang menganalisa masukan *hotkey* pada *keyboard* WASD yang nantinya akan di proses dalam perpindah *vertex* ke traktor pada *Unity engine*. Pada gambar 4.2.1 Jika *hotkey* yang di masukkan berupa W dan S akan berbunyi traktor yang sedang berjalan. Sedangkan jika tidak memasukkan *hotkey* apa apa maka akan terdengar suara traktor yang sedang tidak berjalan.

4.3 *Desain Interface* Permainan

UI, lingkungan, *gameplay*, traktor, dan aset tambahan lainnya semuanya harus dirancang pada saat ini untuk

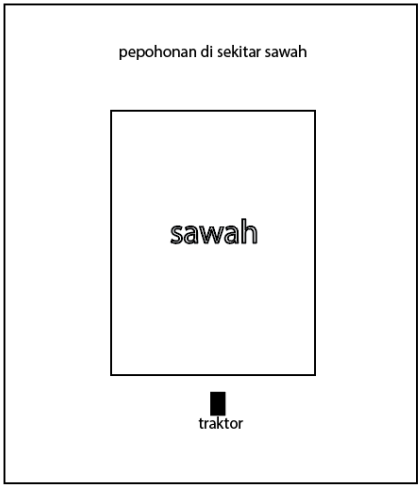
mendukung pengoperasian game. Pada perancangan *Serious Game* ini mengambilan sumber dari daerah di sekitar peneliti yaitu di pulau jawa. Desain area dan asset yang akan di buat akan berupa foto contoh di sekitar peneliti terlebih dahulu yang nanti nya di buat kan versi 3 dimensi nya.

Gambar 4.3 gambar tanah petak

Pada gambar 4.3 menunjukkan tanah kotak dari skitar wilayah peneliti. Ditambah beberapa pohon di sekitar.

Gambar 4.4 Traktor tangan

Lalu pada gambar 4.4 merupakan traktor yang di gunakan setiap hari oleh petani untuk membajak sawah yang berguna untuk mengemburkan tanah.



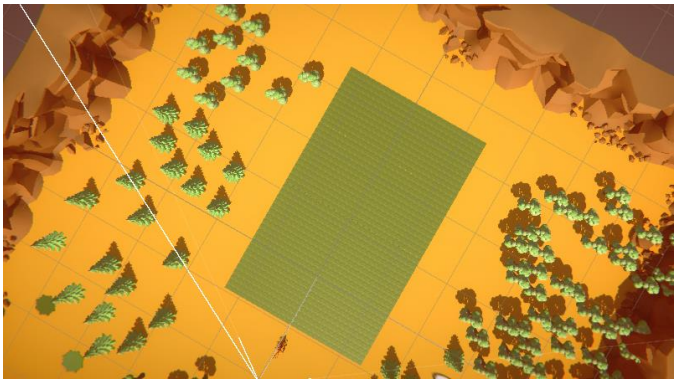
Gambar 4.5 Tata Letak

Kami mulai merancang ruang lingkup game dalam tiga dimensi pada tahap desain tata letak ini dengan menggunakan skenario desain tampilan lingkungan game yang telah ditentukan. Terdapat beberapa tempat untuk persawahan, traktor, dan peralatan pendukung lainnya pada Gambar 4.13.

Gambar 4.6 Desain Tatap Muka atau HUD

Disini tedapat beberapa desain tatap muka dalam *game* yang bisa dikontrol dan ditampilkan ditatap muka *game* nanti yang berupa *Speed Depth* dan traktor kontrol yang dapat dilihat ditatap muka sehingga *Player* dapat memahami hanya dengan melihat tatap muka tanpa menggunakan tutorial.

4.4 Implementasi Objek 3d

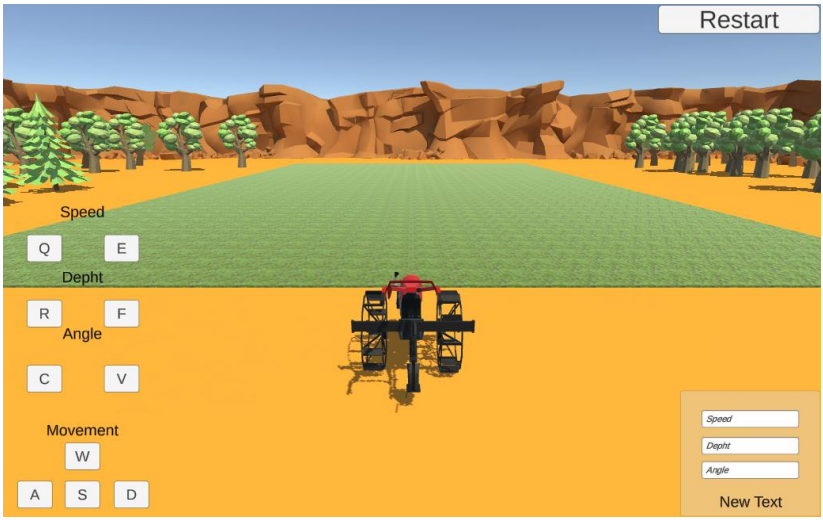


Gambar 4.7 Desain 3D Sawah

Pada Gambar 4.7 sudah dibuatkan nya 1 petak sawah dengan tambahan pohon-pohon dan bebatuan sebagai tambahan inverontment agar lebih terlihat sedikit real.

Gambar 4.8 traktor 3d

Dan pada gambar 4.8 terlihat penggambaran traktor dalam bentuk 3d yang sudah jadi.



Gambar 4.9 di dalam game

Pada Gambar 4.9 terlihat sudah berbentuk game, terdapat *Speed*, *Depth* dan vertical cutting angel yang bisa di atur oleh *Player*. Terdapat juga ui WASD sebagai cara bergeraknya tractor dan di kanan bawah Gambar 4.9 terdapat 3 variable value dan output.

4.5 Implementasi dan Uji Coba *Serious Game*

Implementasi *Game* adalah proses pembuatan game menggunakan desain game dan skenario yang telah ditetapkan.

Pada tahapan ini memulai mengimplementasikan proses pembuatan *game* berupa pembuatan koding yang dibutuhkan agar game dapat berjalan sesuai skenario. Kemudian memasukkan desain 3d dan UI/HUD yang dibutuhkan sesuai desain yang telah direncanakan.

4.5.1 *Source Code Fuzzy* Logic

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.SceneManagement;
public class UseIfLogic : MonoBehaviour
{
    public double TestSpeed;
    public double TestDepth;
    public double TestAngle;
    public string ResultTextPorsity;
    public TextMeshProUGUI ResultTMP;
    public TMP_InputField InputFieldSpeed;
    public TMP_InputField InputFieldDepth;
    public TMP_InputField InputFieldAngle;
    public Material MatSedikitGembur;
```

```
public Material MatGembur;
public Material MatSangatGembur;
public void RestartScene()
{
    SceneManager.LoadScene(0);
}
private void Start()
{
    TestSpeed = 6.808;
    TestDepth = 3;
    TestAngle = 60;
}
private void Update()
{
    SpeedUpdate();
    dephtUpdate();
    angleUpdate();
    OnCalculate();
}
private void SpeedUpdate()
{
    if (Input.GetKey(KeyCode.Q))
```

```
    {
        TestSpeed -= 0.01;
        if (TestSpeed <= 1) TestSpeed = 1;
    }
    if (Input.GetKey(KeyCode.E))
    {
        TestSpeed += 0.01;
        if (TestSpeed >= 19.917) TestSpeed = 19.917;
    }
    InputFieldSpeed.text = TestSpeed.ToString();
}
private void dephtUpdate()
{
    if (Input.GetKey(KeyCode.R))
    {
        TestDepth -= 0.01;
        if (TestDepth <= 0) TestDepth = 0;
    }
    if (Input.GetKey(KeyCode.F))
    {
        TestDepth += 0.01;
        if (TestDepth >= 7) TestDepth = 7;
```

```
    }
    InputFieldDepth.text = TestDepth.ToString();
}
private void angleUpdate()
{
    if (Input.GetKey(KeyCode.C))
    {
        TestAngle -= 0.01;
        if (TestAngle <= 55) TestAngle = 55;
    }
    if (Input.GetKey(KeyCode.V))
    {
        TestAngle += 0.01;
        if (TestAngle >= 70) TestAngle = 70;
    }
    InputFieldAngle.text = TestAngle.ToString();
}
public string ResultSpeed(double value)
{
    string result = "";
    if (value >= 1 && value <= 6.808)
    {
```

```
        result = "Lambat";
    }
    else if (value > 6.808 && value <= 10.169)
    {
        result = "Sedang";
    }
    else if (value > 10.169 && value <= 19.917)
    {
        result = "Cepat";
    }
    return result;
}
public string ResultDepth(double value)
{
    string result = "";
    if (value >= 0 && value <= 3.5)
    {
        result = "Dangkal";
    }
    else if (value > 3.5 && value <= 7)
    {
        result = "Dalam";
```

```
    }
    return result;
}
public string ResultAngle(double value)
{
    string result = "";
    if (value >= 55 && value <= 60)
    {
        result = "Kecil";
    }
    else if (value > 60 && value <= 65)
    {
        result = "Sedang";
    }
    else if (value > 65 && value <= 70)
    {
        result = "Besar";
    }
    return result;
}
public float raycastDistance = 5f;
public void OnCalculate()
```

```
    {
        ResultTextPorsity = "";

        CalculationPorosity();
        Ray ray = new Ray(transform.position, Vector3.down);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit, raycastDistance))
        {
            if (hit.collider.CompareTag("box"))
            {
                MeshRenderer boxRenderer =
hit.collider.GetComponent<MeshRenderer>();
                if (boxRenderer != null)
                {
                    switch (ResultTextPorsity)
                    {
                        case "Sedikit Gembur":
                            boxRenderer.material =
MatSedikitGembur;
                            break;
                        case "Gembur":
                            boxRenderer.material = MatGembur;
```

```
                    break;
                case "Sangat Gembur":
                    boxRenderer.material                    =
MatSangatGembur;
                    break;
                default:
                    break;
            }
        }
    }
}
[ContextMenu ("Test Result")]
public void CalculationPorosity()
{
    string Speed = ResultSpeed(TestSpeed);
    string Depth = ResultDepth(TestDepth);
    string angle = ResultAngle(TestAngle);
    if (Speed == "Lambat" && Depth == "Dangkal" &&
angle == "Kecil")
    {
        ResultTextPorsity = "Sedikit Gembur";
```

```
    }
    else if (Speed == "Lambat" && Depth == "Dangkal"
&& angle == "Sedang")
    {
        ResultTextPorsity = "Sedikit Gembur";
    }
    else if (Speed == "Lambat" && Depth == "Dangkal"
&& angle == "Besar")
    {
        ResultTextPorsity = "Sedikit Gembur";
    }
    else if (Speed == "Lambat" && Depth == "Dalam" &&
angle == "Kecil")
    {
        ResultTextPorsity = "Sedikit Gembur";
    }
    else if (Speed == "Lambat" && Depth == "Dalam" &&
angle == "Sedang")
    {
        ResultTextPorsity = "Sedikit Gembur";
    }
    else if (Speed == "Lambat" && Depth == "Dalam" &&
```

```
angle == "Besar")
      {
         ResultTextPorsity = "Gembur";
      }
      else if (Speed == "Sedang" && Depth == "Dangkal"
&& angle == "Kecil")
      {
         ResultTextPorsity = "Sedikit Gembur";
      }
      else if (Speed == "Sedang" && Depth == "Dangkal"
&& angle == "Sedang")
      {
         ResultTextPorsity = "Gembur";
      }
      else if (Speed == "Sedang" && Depth == "Dangkal"
&& angle == "Besar")
      {
         ResultTextPorsity = "Gembur";
      }
      else if (Speed == "Sedang" && Depth == "Dalam" &&
angle == "Kecil")
      {
```

```
            ResultTextPorsity = "Sedikit Gembur";
        }
        else if (Speed == "Sedang" && Depth == "Dalam" &&
angle == "Sedang")
        {
            ResultTextPorsity = "Gembur";
        }
        else if (Speed == "Sedang" && Depth == "Dalam" &&
angle == "Besar")
        {
            ResultTextPorsity = "Sangat Gembur";
        }
        else if (Speed == "Cepat" && Depth == "Dangkal" &&
angle == "Kecil")
        {
            ResultTextPorsity = "Sedikit Gembur";
        }
        else if (Speed == "Cepat" && Depth == "Dangkal" &&
angle == "Sedang")
        {
            ResultTextPorsity = "Sedikit Gembur";
        }
```

```
        else if (Speed == "Cepat" && Depth == "Dangkal" &&
angle == "Besar")
        {
            ResultTextPorsity = "Gembur";
        }
        else if (Speed == "Cepat" && Depth == "Dalam" &&
angle == "Kecil")
        {
            ResultTextPorsity = "Gembur";
        }
        else if (Speed == "Cepat" && Depth == "Dalam" &&
angle == "Sedang")
        {
            ResultTextPorsity = "Gembur";
        }
        else if (Speed == "Cepat" && Depth == "Dalam" &&
angle == "Besar")
        {
            ResultTextPorsity = "Sangat Gembur";
        }
        ResultTMP.text = ResultTextPorsity;
    }
```

```
}
```

Gambar 5.0 *source code Fuzzy* beserta *Input keybind*

Pada gambar 5.0 merupakan *source code* yang berguna menjalan kan algoritma *Fuzzy* pada game sehingga game dapat berjalan dengan baik, dari mulai pengelompokan keanggotaan hingga penentuan aturan/*Rules* hingga hasil/*Output*. Pada *source code* tersebut juga terdapat tedapat keybind atau tombol yang dapat di klik oleh *Player* untuk menaikkan dan menurunkan *variable* dari *Speed, Depth, dan vertical cutting angel*, sehingga *Player* dapat melihat secara langsung perubahan angka *Input* yang berubah dan dapat disesuaikan dengan *output* apa yang diinginkan oleh *Player*, *Source code* tersebut juga dapat menampilkan secara langsung kepada *Player* dalam bentuk perubahan pada tanah, karena *source code* tersebut tersambung dengan *assets* tanah pada game, apakah sedikit gembur, gembur, atau sangat gembur, seperti yang di tampilkan pada gambar 5.2 sampai gambar 5.5.

4.5.2 *Source code* Pergerakan traktor dan *Input* suara

```
using UnityEngine;
```

```
namespace Complete
{
    public class TankMovement : MonoBehaviour
    {
        public int m_PlayerNumber = 1;
        public float m_Speed = 12f;
        public float m_TurnSpeed = 180f;
        public AudioSource m_MovementAudio;
        public AudioClip m_EngineIdling;
        public AudioClip m_EngineDriving;


        private string m_MovementAxisName;
        private string m_TurnAxisName;
        private Rigidbody m_Rigidbody;
        private float m_MovementInputValue;
        private float m_TurnInputValue;
```

```
private ParticleSystem[] m_particleSystems;
public GameObject objekRodaLeft;
public GameObject objekRodaRight;
private void Awake ()
{
    m_Rigidbody = GetComponent<Rigidbody> ();
}
private void OnEnable ()
{
    m_Rigidbody.isKinematic = false;

    m_MovementInputValue = 0f;
    m_TurnInputValue = 0f;

    m_particleSystems =
```

```
GetComponentsInChildren<ParticleSystem>();

        for (int i = 0; i < m_particleSystems.Length; ++i)

        {

            m_particleSystems[i].Play();

        }

    }

    private void OnDisable ()

    {

        m_Rigidbody.isKinematic = true;

        for(int i = 0; i < m_particleSystems.Length; ++i)

        {

            m_particleSystems[i].Stop();

        }

    }

    private void Start ()

    {
```

```
        m_MovementAxisName = "Vertical" +
m_PlayerNumber;

        m_TurnAxisName = "Horizontal" +
m_PlayerNumber;

    }

    public float rotationSpeed ;

    private void Update ()

    {

        m_MovementInputValue = Input.GetAxis
(m_MovementAxisName);

        if (m_MovementInputValue > 0)

        {

            objekRodaRight.transform.Rotate(Vector3.right  ,
rotationSpeed);



            objekRodaLeft.transform.Rotate(Vector3.right,
rotationSpeed);
```

```
        }
        m_TurnInputValue = Input.GetAxis
(m_TurnAxisName);
        EngineAudio ();
    }
    private void EngineAudio ()
    {
        if (Mathf.Abs (m_MovementInputValue) < 0.1f &&
Mathf.Abs (m_TurnInputValue) < 0.1f)
        {
            if (m_MovementAudio.clip == m_EngineDriving)
            {
                m_MovementAudio.clip = m_EngineIdling;
                m_MovementAudio.Play ();
            }
        }
        else
```

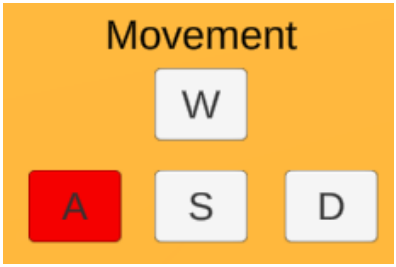
```
        {
            if (m_MovementAudio.clip == m_EngineIdling)
            {
                m_MovementAudio.clip = m_EngineDriving;
                m_MovementAudio.Play();
            }
        }
    }
    private void FixedUpdate ()
    {
        Move ();
        Turn ();
    }
    private void Move ()
    {
        Vector3 movement = transform.forward *
```

```
m_MovementInputValue * m_Speed * Time.deltaTime;

        m_Rigidbody.MovePosition(m_Rigidbody.position +
movement);

    }

    private void Turn ()

    {

        float turn = m_TurnInputValue * m_TurnSpeed *
Time.deltaTime;



        Quaternion turnRotation = Quaternion.Euler (0f, turn,
0f);

        m_Rigidbody.MoveRotation (m_Rigidbody.rotation
* turnRotation);

    }

  }

}
```

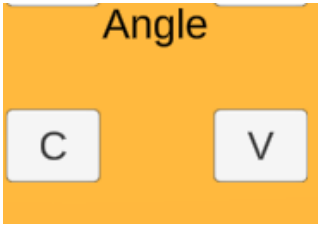
Gambar 5.1 *Source code* Pergerakan traktor dan *Input* suara

Pada gambar 5.1 yang merupakan *Source code* pergerakan dari traktor sekaligus merupakan tempat dimana suara traktor saat sedang diam dan sedang berjalan sehingga *Player* dapat merasakan secara nyata mungkin dengan ada nya suara traktor.

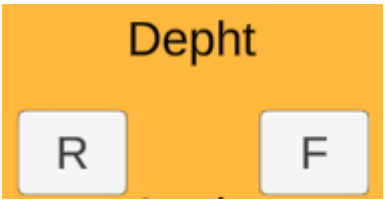
4.6 Implementasi *UI/HUD* dan Uji Coba *Serious Game*



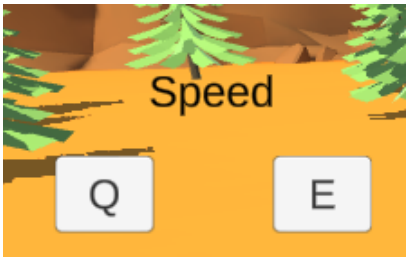
Gambar 5.2 *movement*



Gambar 5.3 *vertical cutting angel*

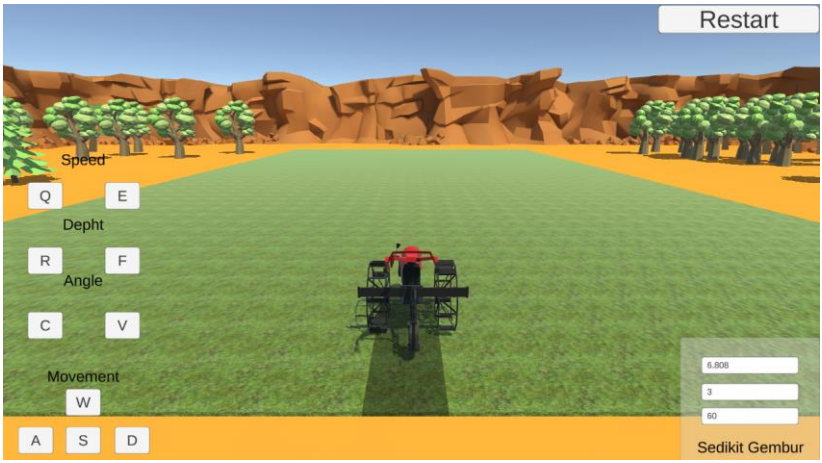


Gambar 5.4 *Depth*



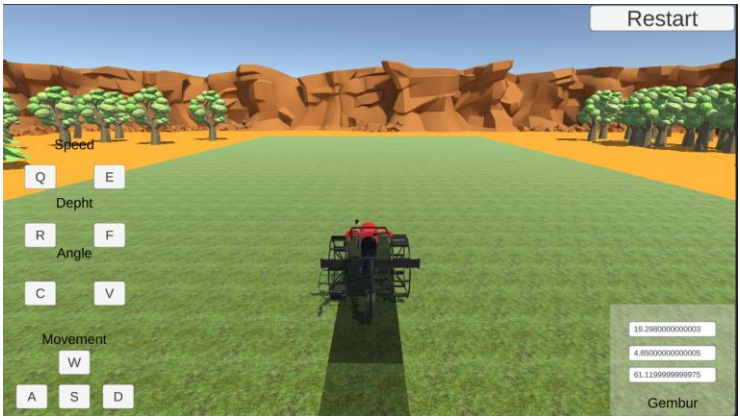
Gambar 5.5 *Speed*

Bisa kita lihat pada gambar 5.2 kita menggunakan huruf WASD pada *keyboard* untuk menggerakan traktor, W untuk maju, A untuk belok kiri, D untuk belok kanan dan S untuk membuat traktor mundur. Terdapat juga gambar 5.3 C dan V, C untuk menurunkan *vertical cutting angel* dan V untuk menaikkan *vertical cutting angel*. Lalu gambar 5.4 R dan F, huruf R untuk menurunkan nilai *Depth* dan huruf F untuk menaikkan nilai *Depth.* Yang terakhir pada gambar 5.5 terdapat huruf Q dan E, huruf Q untuk menurunkan nilai *Speed* dan huruf E untuk menaikkan nilai *Speed.*



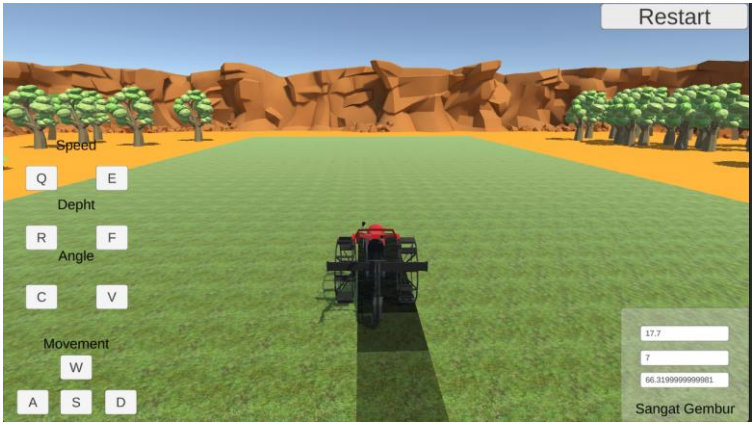
Gambar 5.6 tidak gembur

Pada gambar 5.6 pada kanan bawah gambar ketika diproses *Fuzzy* dan hasil keluarannya tidak gembur maka tanah akan sedikit tidak gelap warna nya.



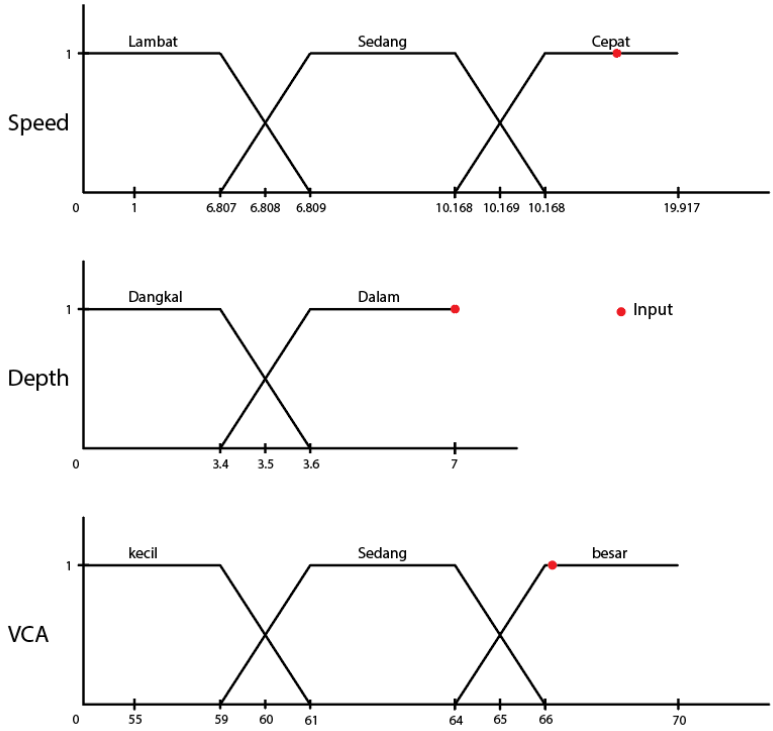
Gambar 5.7 gembur

Pada gambar 5.7 pada kanan bawah gambar ketika diproses *Fuzzy* dan hasil keluarannya tidak gembur maka tanah akan sedikit gelap warnanya. Berbeda dengan warna sebelum nya yang tidak terlalu gelap.



Gambar 5.8 sangat gembur

Pada gambar 5.8 pada kanan bawah gambar ketika diproses *Fuzzy* dan hasil keluarannya tidak gembur maka tanah akan sangat gelap warna nya. Berbeda dengan 2 warna sebelum nya yang tidak terlalu gelap.



Gambar 5.9 Grafik proses *fuzzy* 5.8 manual

$$[X]=\begin{cases}0, & X \le X\min \\ \dfrac{X-X\min}{X\max-X\min}, & X\min \le X \le X\max \\ 1, & X \ge X\max\end{cases}$$

Gambar 6.0 Rumus *Fuzzy MIN MAX*

Terlihat pada Gambar grafik pada Gambar 5.9 merupakan proses *fuzzyfikasi* dari gambar 5.8. titik merah merupakan nilai *input* yang dicoba pada percobaan Gambar 5.8 . Dari grafik pada Gambar 5.9 tersebut nilai dari *variable speed* adalah 17.7, 17.7 masuk dalam fungsi keanggotaan dari cepat yang merupakan *MIN*=10.168 sampai *MAX*=19.917. pada nilai *variable depth* adalah 7, 7 masuk dalam fungsi keanggotaan dari dalam yang merupakan *MIN*=3.4 sampai *MAX*=7. Pada *variable* terakhir yaitu *vertical cutting angel* nilainya adalah 66.32, 66.32 masuk dalam fungsi keanggotaan dari besar yang merupakan *MIN*=64 sampai *MAX*=70. Sesuai dengan rumus pada Gambar 6.0. Setelah menganalisa nilai terhadap fungsi keanggotaan dari masing-masing *variable* nantinya akan dicari kesamaan dari rules yang telah di buat.

Tabel 4.1 Hasil Ujicoba *fuzzy* pada *game*

NO	Input			Output	
	speed	depth	Vertical Cutting Angel	Manual	Serious Game
1	7.35	3.12	58.2	Sedikit gembur	Sedikit gembur
2	12.25	4	62	Gembur	Gembur
3	8.5	5.4	64.5	Gembur	Gembur
4	13.4	6.1	66.8	Sangat Gembur	Sangat Gembur
5	5.3	4.6	67.4	Gembur	Gembur
6	4.9	5	62.2	Sedikit gembur	Sedikit gembur
7	10.4	2.5	68.8	Gembur	Gembur
8	7.3	6.2	66.6	Sangat Gembur	Sangat Gembur
9	4.4	5.7	60.5	Sedikit gembur	Sedikit gembur
10	8.1	6	67.7	Sangat Gembur	Sangat Gembur

Pada Tabel 4.1 merupakan 10 kali percobaan dari permainan bajak singkal yang telah dibuat, dengan memasukkan masukan yang ingin dicari dari *speed*, *depth*, dan *vertical cutting angel*. Pada tabel 4.1 dari 10 kali percobaan semuanya mendapatkan hasil keluaran yang sama dengan perhitungan manual yang telah dicoba sebelumnya.

Jika ingin mengulangi *game Player* dapat mengeklik *restart* pada kanan atas gambar 5.8. untuk mengulangi dari awal.

Dari percobaan di atas dapat dilihat hasil dari *Input* yang berupa 3 variable tersebut dapat berjalan dan menghasilkan hasil seperti pada gambar 5.7 sampai gambar 5.8 , Dimana tanah akan berubah sesuai dengan kegemburan tanah yang dihasilkan oleh algoritma *Fuzzy* yang dijalankan oleh *game* tersebut.

Kinerja algoritma *Fuzzy* Logic dievaluasi berdasarkan sejauh mana algoritma ini mampu menghasilkan keputusan yang akurat dalam konteks *game* pengolahan tanah. Beberapa indikator yang digunakan untuk mengukur keakuratan meliputi:

- Kecocokan dengan Data Empiris: Algoritma diuji dengan

data empiris yang mencerminkan situasi dalam game. Hasil dari algoritma harus sesuai dengan hasil yang diharapkan berdasarkan data tersebut.

- Kemampuan Menangani Variabilitas: Algoritma harus mampu menangani variasi *Input*, seperti kecepatan, kedalam, dan faktor lainnya.

- Kesesuaian dengan Aturan *Fuzzy*: Keputusan yang dihasilkan oleh algoritma harus sesuai dengan aturan *Fuzzy* yang telah ditentukan.

Selain keakuratan, efisiensi algoritma juga sangat penting dalam game. Evaluasi efisiensi melibatkan beberapa aspek berikut:

Waktu Eksekusi: Algoritma harus bekerja dengan cepat dan memberikan respons yang hampir instan kepada pemain. Lamanya waktu eksekusi algoritma harus sesuai dengan ekspektasi pengguna.

Penggunaan Sumber Daya: Algoritma harus dapat beroperasi tanpa menghabiskan terlalu banyak sumber daya komputer seperti CPU dan RAM.

Kemampuan Penyesuaian: Algoritma harus dapat mengatur

dirinya sendiri sesuai dengan tingkat kesulitan atau variasi yang ada dalam permainan.