Panduan Setup AI Animal

1. Pendahuluan

Dalam dokumentasi ini, Anda akan belajar bagaimana mengimplementasikan AI hewan di Unity menggunakan pendekatan State Machine. Sebagai contoh, kita akan membahas cara membuat AI untuk Goose yang memiliki dua perilaku utama: **Idle** dan **Runaway**.

2. Persyaratan Awal

Pastikan proyek Unity Anda memenuhi persyaratan berikut:

- Unity Engine versi terbaru atau sesuai kebutuhan proyek.
- Paket **AI Navigation** terinstal untuk pergerakan menggunakan NavMesh.
- **Model 3D** dari hewan yang dilengkapi dengan animasi.
- Script AI yang akan dibuat untuk mengatur logika perilaku hewan.

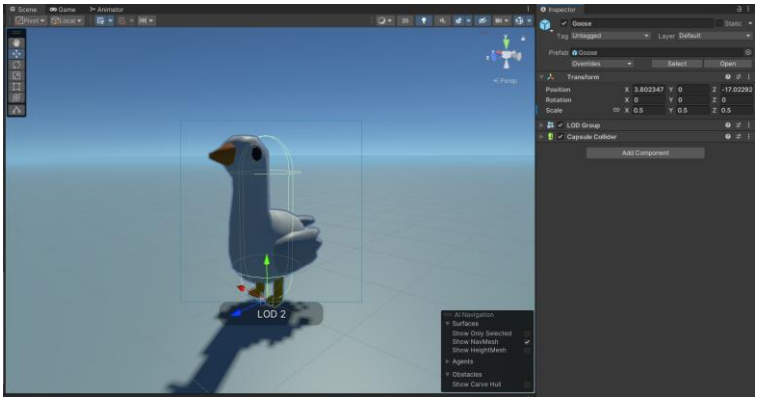
3. Setup Umum untuk Hewan

3.1. Langkah-Langkah Awal

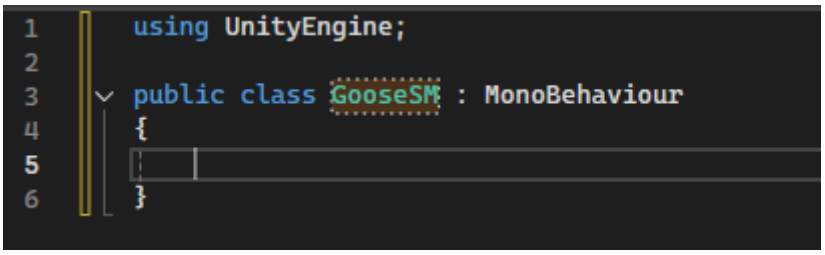
1. **Import Model**: Impor model 3D hewan yang ingin digunakan ke dalam proyek Unity.
2. **Tambahkan Komponen Dasar**:
    - **NavMeshAgent**: Tambahkan komponen NavMeshAgent pada objek hewan untuk mengatur navigasi.
    - **Animator**: Tambahkan komponen Animator untuk mengelola animasi.
    - **Collider dan Trigger**: Tambahkan Collider yang diatur sebagai Trigger untuk mendeteksi pemain atau objek lain.

3.2. Membuat Kelas State Machine

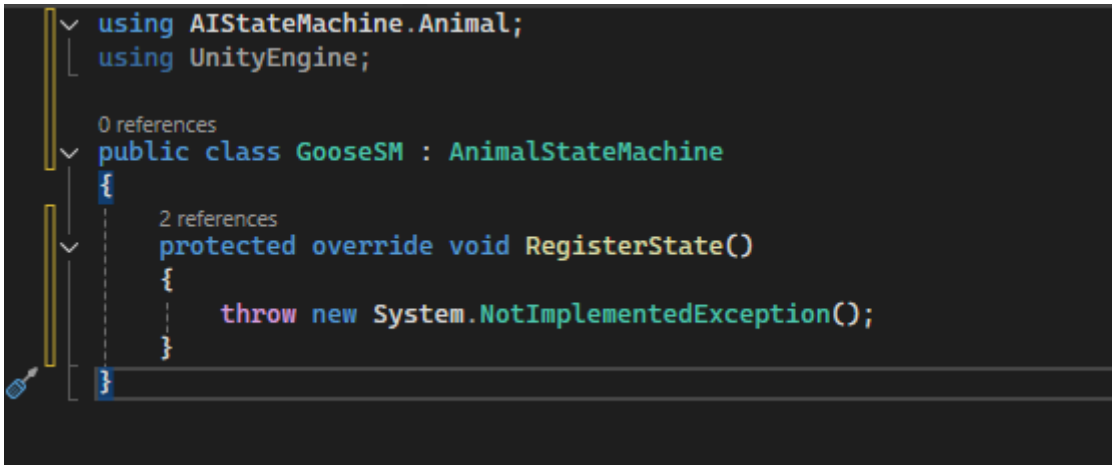
Buatlah kelas State Machine untuk mengatur perilaku hewan. Pada contoh ini, kita akan membuat kelas bernama GooseSM untuk mengatur perilaku Goose. Tambahkan prefab Goose ke dalam Scene.



Buatlah kelas baru dengan nama **GooseSM** (SM adalah singkatan dari StateMachine). Berikut tampilan awal kelas **GooseSM** yang masih kosong.



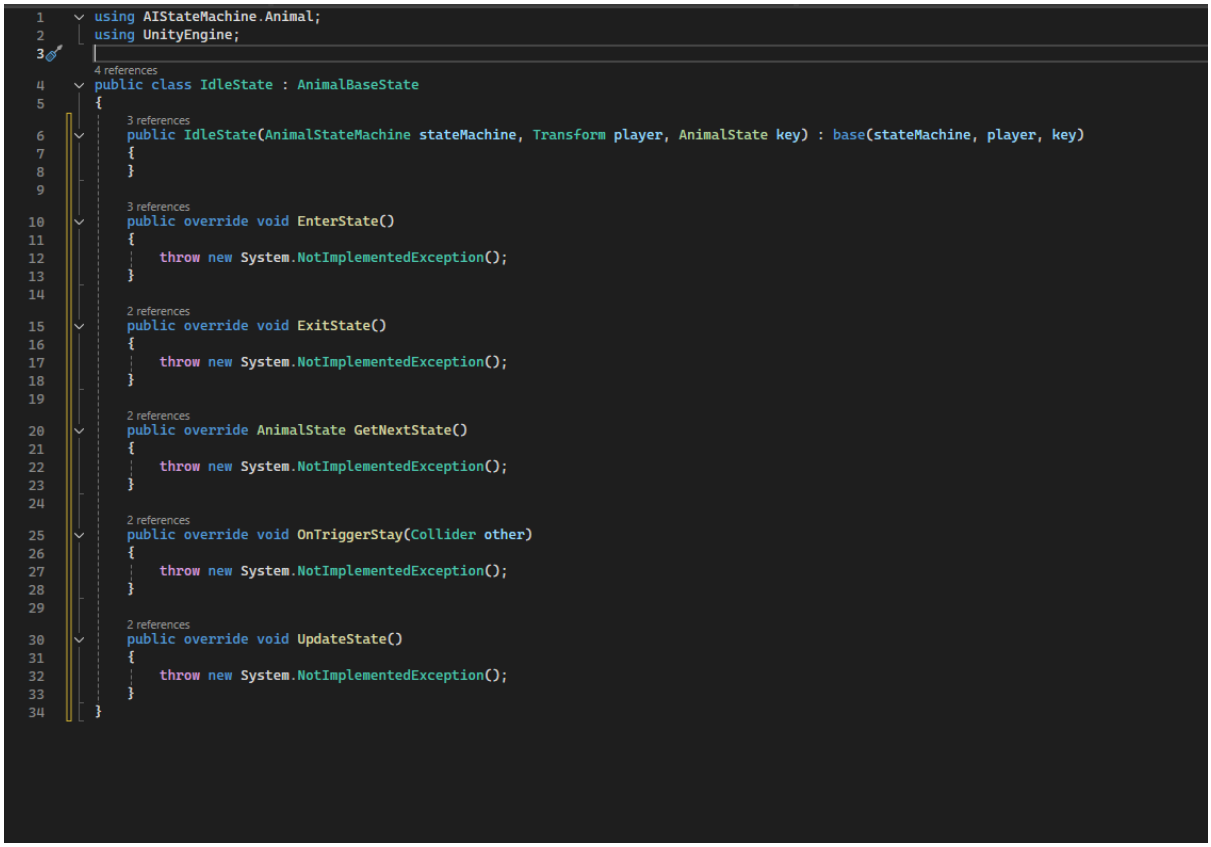
Kita tidak akan menginherit dari MonoBehaviour namun dari kelas parent lain yaitu **AnimalStateMachine**.



AnimalStateMachine membutuhkan method RegisterState sebagai bagian dari kontrak kelas abstract. Method ini berfungsi sebagai pendaftaran behaviour untuk Goose.

3.2. Membuat Kelas Base State

Sebelum melanjutkan, terlebih dahulu buat kelas base state untuk behavior dari animal yang ingin dibuat, dalam contoh kali ini adalah behaviour Goose yaitu Idle dan Runaway. Buatlah kelas dengan nama **IdleState** dan menginherit ke **AnimalBaseState** dan implementasikan seluruh method turunannya.



```
using AIStateMachine.Animal;
using UnityEngine;

4 references
public class IdleState : AnimalBaseState
{
    3 references
    public IdleState(AnimalStateMachine stateMachine, Transform player, AnimalState key) : base(stateMachine, player, key)
    {
    }

    3 references
    public override void EnterState()
    {
        throw new System.NotImplementedException();
    }

    2 references
    public override void ExitState()
    {
        throw new System.NotImplementedException();
    }

    2 references
    public override AnimalState GetNextState()
    {
        throw new System.NotImplementedException();
    }

    2 references
    public override void OnTriggerStay(Collider other)
    {
        throw new System.NotImplementedException();
    }

    2 references
    public override void UpdateState()
    {
        throw new System.NotImplementedException();
    }
}
```

Tambahkan variable float idleTimer untuk logic randomize animations, dimana jika idleTimer ini sudah terlampaui, eksekusi logic randomize nya. Variabel array string randomIdleAnim untuk menyampaikan animasi dari **StateMachine** ke **BaseState.** Lalu ubah parameter dari konstruktor **IdleState** menjadi seperti berikut

```
private float idleTimer, idleTimeInterval;
private string[] randomIdleAnim;

3 references
public IdleState(AnimalStateMachine stateMachine, Transform player, float idleTimer, string[] randomIdleAnim) : base(stateMachine, player, AnimalState.Idle)
{
    this.idleTimer = idleTimer;
    this.randomIdleAnim = randomIdleAnim;
}
```

EnterState() dieksekusi sekali ketika pertama kali memasuki state, disini mainkan animasi seperti Idle_A agar animasi pertama yang dijalankan adalah Goose dalam kondisi idle.

```
3 references
public override void EnterState()
{
    stateMachine.PlayAnimation("Idle_A");
}
```

UpdateState() dieksekusi tiap frame ketika state berjalan, disinilah logic randomize animation dijalankan, perintahnya adalah sebagai berikut.

```
2 references
public override void UpdateState()
{
    idleTimeInterval += Time.deltaTime;
    if (idleTimeInterval < idleTimer) return;

    stateMachine.PlayRandomAnimations(randomIdleAnim);

    idleTimeInterval = 0;
}
```

GetNextState() berisi perintah untuk transisi state. Disini perpindahan state diatur, dan karena behavior yang Goose miliki adalah kabur ketika ada player disekitarnya maka tambahkan seperti berikut.

```
2 references
public override AnimalState GetNextState()
{
    if (stateMachine.playerInSight)
        return AnimalState.Runaway;

    return StateKey;
}
```

Sehingga keseluruhan skrip **IdleState** adalah sebagai berikut.

```
public class IdleState : AnimalBaseState
{
    private float idleTimer, idleTimeInterval;
    private string[] randomIdleAnim;

    3 references
    public IdleState(AnimalStateMachine stateMachine, Transform player, float idleTimer, string[] randomIdleAnim) : base(stateMachine, player, AnimalState.Idle)
    {
        this.idleTimer = idleTimer;
        this.randomIdleAnim = randomIdleAnim;
    }

    3 references
    public override void EnterState()
    {
        stateMachine.PlayAnimation("Idle_A");
    }

    2 references
    public override void UpdateState()
    {
        idleTimeInterval += Time.deltaTime;
        if (idleTimeInterval < idleTimer) return;

        stateMachine.PlayRandomAnimations(randomIdleAnim);

        idleTimeInterval = 0;
    }

    2 references
    public override void ExitState()
    {
    }

    2 references
    public override AnimalState GetNextState()
    {
        if (stateMachine.playerInSight)
            return AnimalState.Runaway;

        return StateKey;
    }

    2 references
    public override void OnTriggerStay(Collider other)
    {
    }
}
```

Setelah membuat skrip **IdleState**, selanjutnya adalah kelas **RunawayState**. Seperti **IdleState**, kelas **RunawayState** juga menginherit dari **AnimalBaseState**. Karena logic dari Runaway sendiri adalah kabur menjauh dari pemain, kita membutuhkan **NavMeshAgent** untuk mencapai tujuan tersebut, buat variable NavMeshAgent dan array string randomAnim untuk merandomize animasi dalam kondisi Runaway.

```
private NavMeshAgent agent;

private string[] randomAnim;

1 reference
public RunawayState(AnimalStateMachine stateMachine, Transform player, NavMeshAgent agent, string[] randomAnim) :
    base(stateMachine, player, AnimalState.Runaway)
{
    this.agent = agent;
    this.randomAnim = randomAnim;
}
```

Buat method dengan nama RunawayPath sebagai logic untuk membuat Goose menjauh dari player dan mengatur animasi Runaway sebagai berikut. RunawayPattern adalah method dari kelas parent **AnimalBaseState** dengan parameter variable NavMeshAgent untuk mengatur pelarian dari animal.

```
1 reference
private void RunawayPath()
{
    RunawayPattern(agent);

    stateMachine.PlayRandomAnimations(randomAnim);
}
```

Panggil RunawayPath dalam EnterState() agar dijalankan langung ketika Goose telah memasuki state Runaway.

```
3 references
public override void EnterState()
{
    RunawayPath();
}
```

Ketika Goose beralih dari RunawayState, tambahkan kondisi untuk mengatur reset pada logic AI NavMesh dari Goose karena telah meninggalkan kondisi Runaway dengan menambahkan NavMeshAgent.ResetPath pada ExitState().

```
2 references
public override void ExitState()
{
    agent.ResetPath();
}
```

Buat variable tipe Boolean dengan nama IsReachedPoint untuk mengcek kondisi apakah Goose sudah sampai dititik lokasi yang telah diatur oleh RunawayPattern atau belum.

```
1 reference
private bool IsReachedPoint()
{
    if (agent.remainingDistance <= agent.stoppingDistance)
        return true;

    return false;
}
```

Variabel tersebut akan digunakan untuk menentukan transisi dari **RunawayState** ini. Pada GetNextState(), jika Goose telah mencapai titik lokasi, kembalikan state ke kondisi Idle seperti berikut.

```
2 references
public override AnimalState GetNextState()
{
    if (IsReachedPoint())
        return AnimalState.Idle;

    return StateKey;
}
```

Maka keseluruhan skrip **RunawayState** adalah sebagai berikut.

```
3 references
public class RunawayState : AnimalBaseState
{
    private NavMeshAgent agent;

    private string[] randomAnim;

    2 references
    public RunawayState(AnimalStateMachine stateMachine, Transform player, NavMeshAgent agent, string[] randomAnim) :
        base(stateMachine, player, AnimalState.Runaway)
    {
        this.agent = agent;
        this.randomAnim = randomAnim;
    }

    3 references
    public override void EnterState()
    {
        RunawayPath();
    }

    2 references
    public override void UpdateState()
    {
    }

    2 references
    public override void ExitState()
    {
        agent.ResetPath();
    }

    2 references
    public override AnimalState GetNextState()
    {
        if (IsReachedPoint())
            return AnimalState.Idle;

        return StateKey;
    }

    2 references
    public override void OnTriggerStay(Collider other)
    {
    }

    1 reference
    private void RunawayPath()
    {
        RunawayPattern(agent);
```

3.3. Merampungkan State Machine

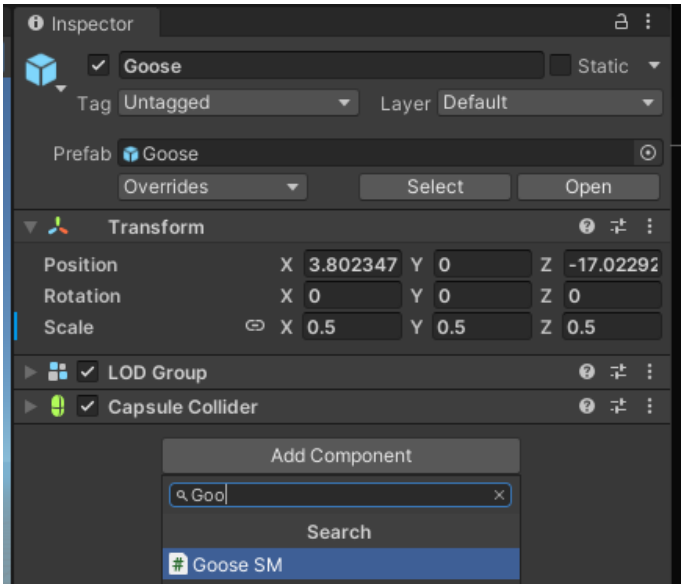
Kembali ke kelas **GooseSM**, tambahkan variable string array untuk menyimpan banyak animasi run, karena Goose memiliki animasi lari sambil terbang mengepakkan sayapnya.

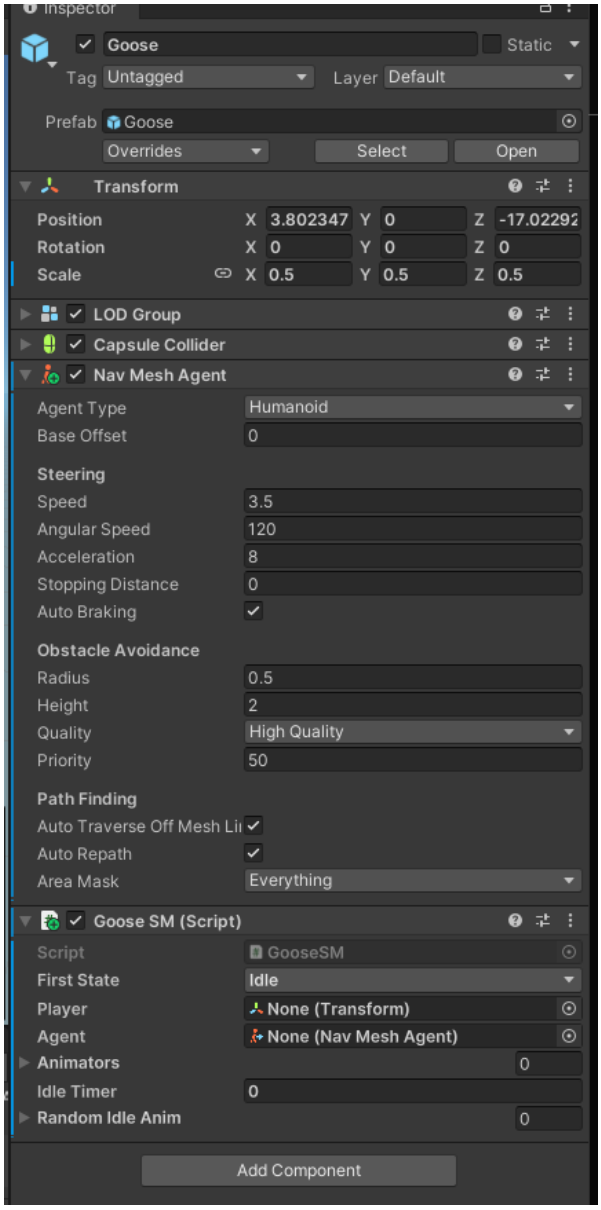
```
using AIStateMachine.Animal;
using UnityEngine;

Unity Script | 0 references
public class GooseSM : AnimalStateMachine
{
    [SerializeField] private string[] randomRunAnimations;

    2 references
    protected override void RegisterState()
    {
        AssignComponent();

    }
}
```

Karena kita sudah punya kelas base behaviour yaitu **IdleState** dan **RunawayState** yang telah dibuat sebelumnya, tambahkan kedua behaviour class tersebut menggunakan konstruktor ke dalam RegisterState(), lalu tambahkan CurrentState ke firstState agar bisa mengatur behaviour pertama Goose saat game berjalan pada Inspector.

```
using AIStateMachine.Animal;
using UnityEngine;

Unity Script | 0 references
public class GooseSM : AnimalStateMachine
{
    [SerializeField] private string[] randomRunAnimations;

    2 references
    protected override void RegisterState()
    {
        AssignComponent();

        States.Add(AnimalState.Idle, new IdleState(this, player, idleTimer, randomIdleAnim));
        States.Add(AnimalState.Runaway, new RunawayState(this, player, agent, randomRunAnimations));

        CurrentState = States[firstState];
    }
}
```

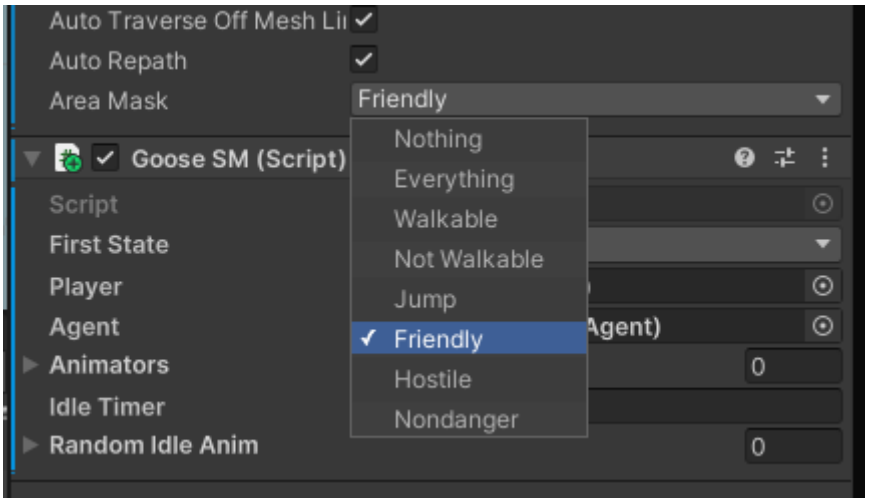
Kelas StateMachine telah rampung dan siap dijalankan. Kembali ke Unity dan tambahkan komponen GooseSM, maka secara otomatis NavMeshAgent juga akan terinput.


Inspector
Goose
Static
Tag Untagged
Layer Default
Prefab Goose
Overrides
Select
Open
Transform
Position X 3.802347 Y 0 Z -17.02292
Rotation X 0 Y 0 Z 0
Scale X 0.5 Y 0.5 Z 0.5
LOD Group
Capsule Collider
Add Component
Goo
Search
Goose SM

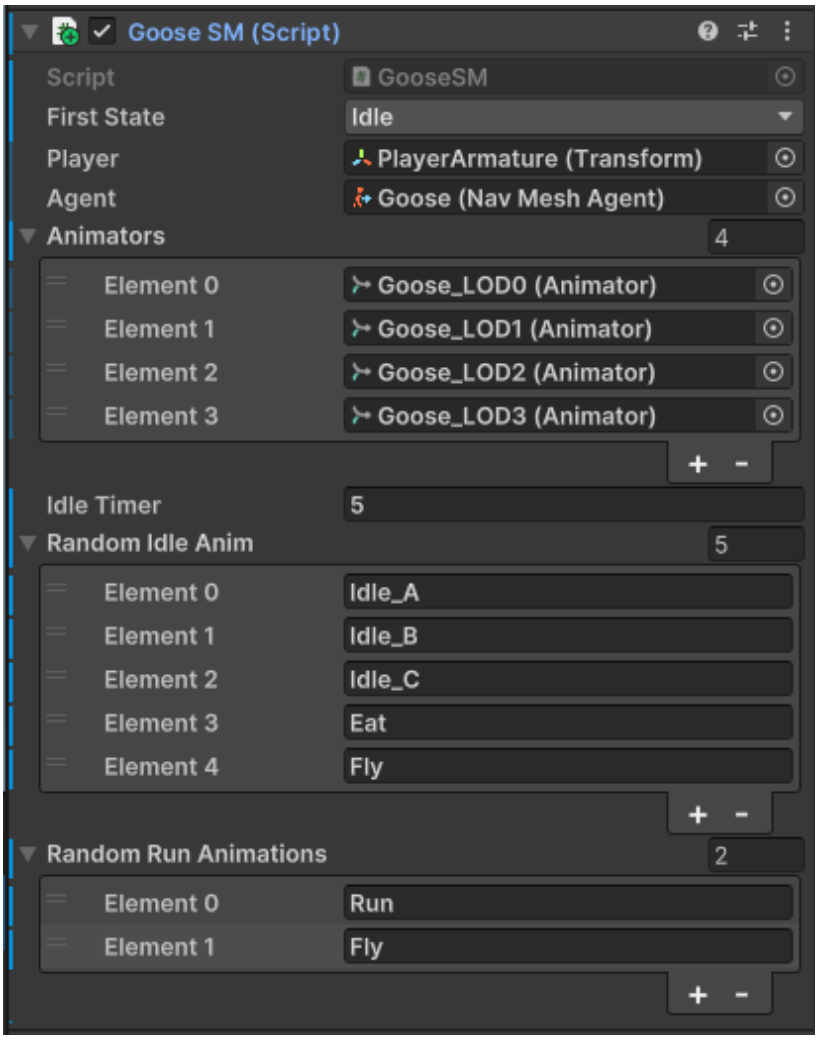

Goose
Static
Tag Untagged
Layer Default
Prefab Goose
Overrides
Select
Open
Transform
Position X 3.802347 Y 0 Z -17.02292
Rotation X 0 Y 0 Z 0
Scale X 0.5 Y 0.5 Z 0.5
LOD Group
Capsule Collider
Nav Mesh Agent
Agent Type Humanoid
Base Offset 0
Steering
Speed 3.5
Angular Speed 120
Acceleration 8
Stopping Distance 0
Auto Braking
Obstacle Avoidance
Radius 0.5
Height 2
Quality High Quality
Priority 50
Path Finding
Auto Traverse Off Mesh Li
Auto Repath
Area Mask Everything
Goose SM (Script)
Script GooseSM
First State Idle
Player None (Transform)
Agent None (Nav Mesh Agent)
Animators 0
Idle Timer 0
Random Idle Anim 0
Add Component

3.4. Mengatur komponen GameObject

Kecepatan Goose dapat diatur pada bagian speed dalam NavMeshAgent, dan AreaMask harus diatur sesuai kelompok hewan, dimana Goose termasuk kedalam kelompok Friendly.

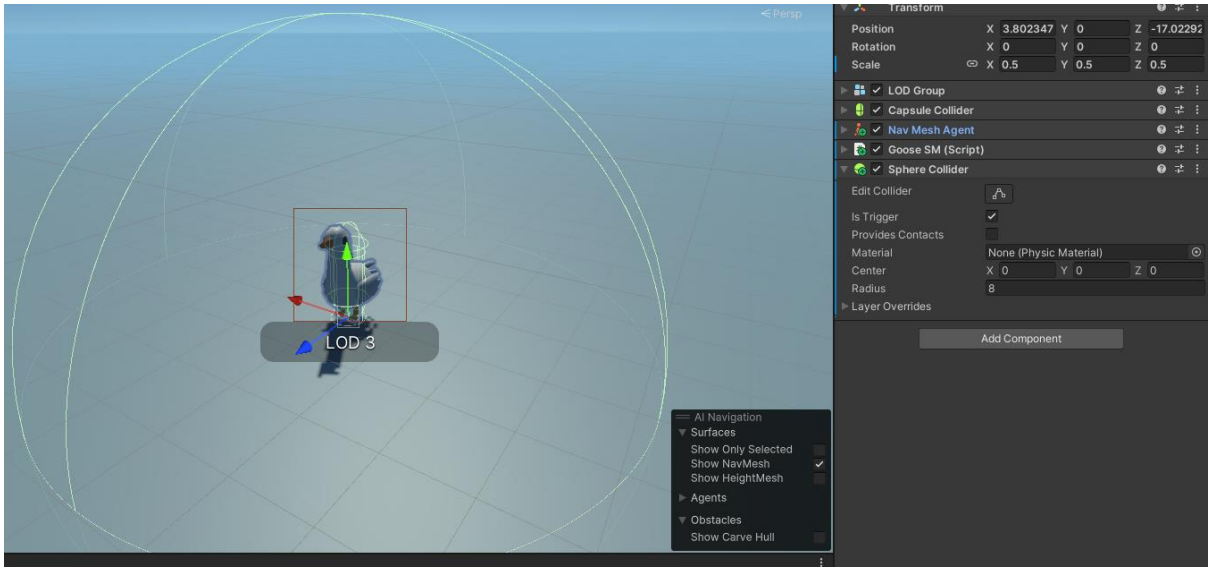


Masukkan semua reference yang dibutuhkan dalam Inspector ke komponen GooseSM, dan ini **WAJIB** jika tidak menambahkan AssignComponent dalam kelas **GooseSM.**



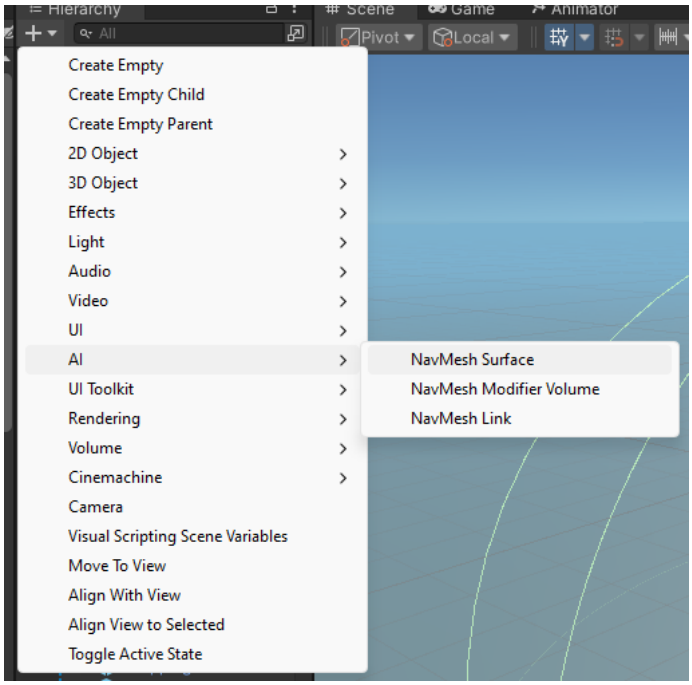
IdleTimer adalah waktu dalam detik yang dibutuhkan untuk kondisi Idle, dalam kasus Goose jika diatur bernilai 5, maka setiap 5 detik animasi idle akan berganti secara acak sesuai yang telah diinput dalam array Random Idle Anim. RandomRunAnimations sama seperti Random Idle Anim, merupakan array untuk menyimpan animasi untuk keperluan Runaway secara acak.

RunawayState membutuhkan trigger collision untuk mendeteksi player, oleh karena itu tambahkan collider apapun (Sphere Collider adalah rekomendasi) dan centang Is Trigger. Atur ukuran trigger sesuai kebutuhan untuk Goose dapat seberapa jauh mendeteksi adanya pemain mendekat.

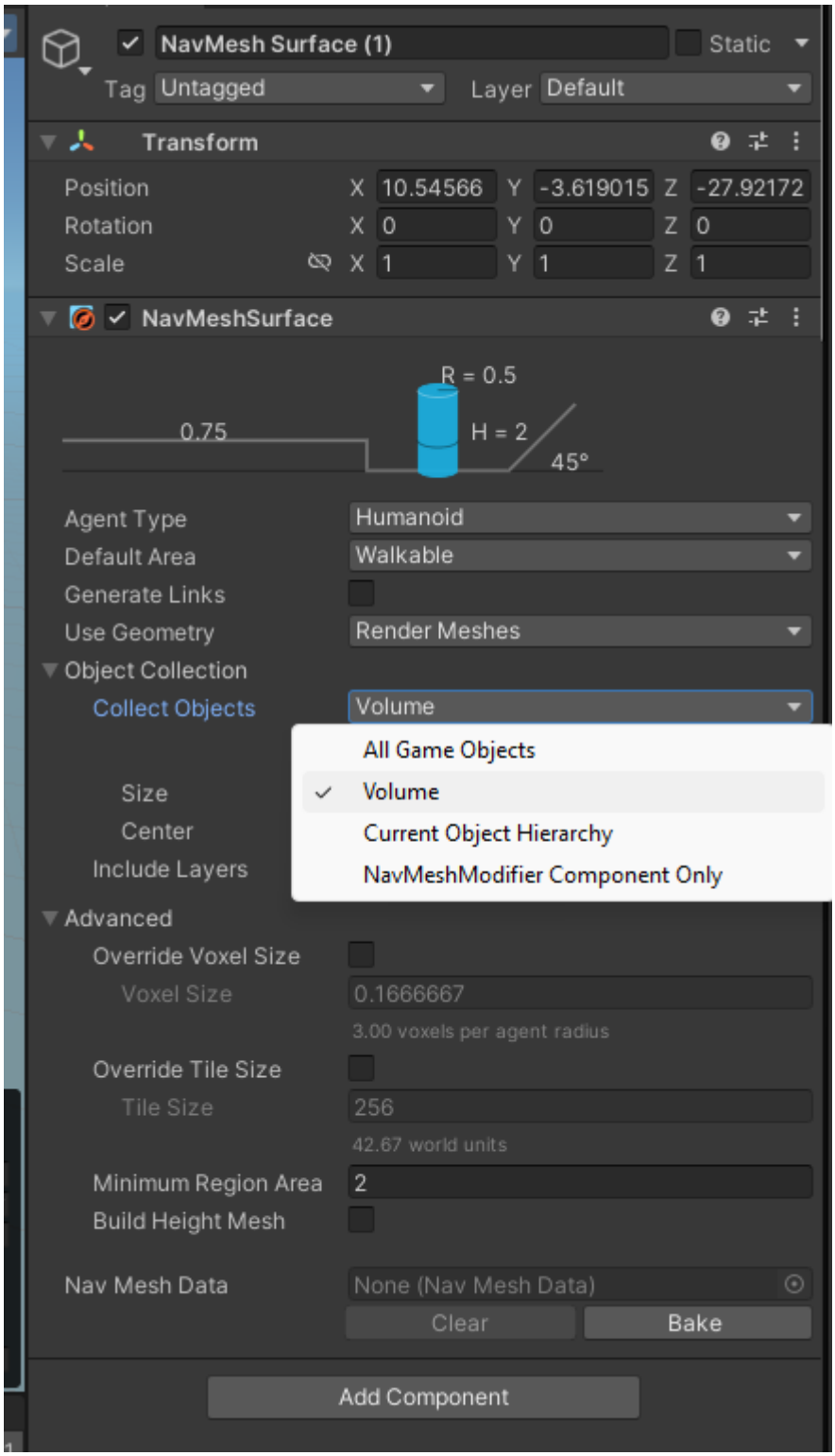


3.5. Mengatur Surface

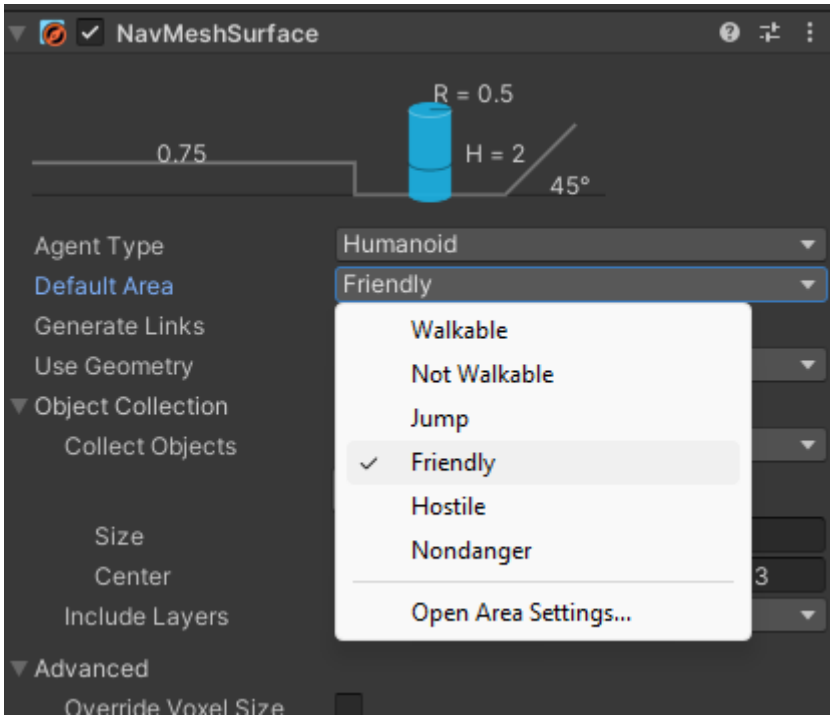
NavMeshAgent bekerja menggunakan surface yang telah disetup. Klik Create GameObject>AI>NavMesh Surface.



Untuk membuat area sendiri bagi Goose, jadi Goose tidak bisa pergi terlalu jauh, kita atur surface untuk Goose menggunakan Volume pada pilihan Collect Objects, lalu posisikan ditempat yang diinginkan.



Atur Default Area menjadi sesuai dengan Area Mask pada agent Goose, yaitu Friendly.



Ketika dibake, jadilah seperti ini. AI Goose sudah siap dijalankan.