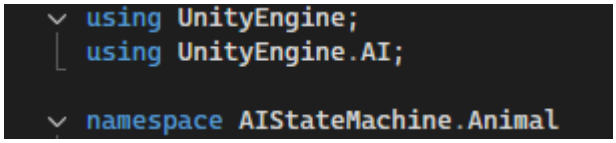
Dokumentasi AI Animal Scripting

Summary

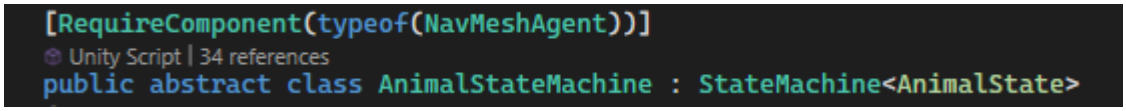
- **AnimalStateMachine** adalah kerangka dasar state machine untuk AI hewan yang mengelola komponen utama seperti NavMeshAgent dan Animator.
- **AnimalBaseState** memberikan logika dasar untuk setiap state dalam state machine, termasuk metode untuk patroli dan melarikan diri dari pemain.

AnimalStateMachine.cs



Namespace

- **namespace AIStateMachine.Animal**
  - Berisi kelas-kelas untuk implementasi state machine AI hewan.



Class

- **AnimalStateMachine**
  - Kelas abstrak yang mewarisi StateMachine<AnimalState>. Digunakan untuk mengelola state machine dari hewan yang berbeda dalam game.
  - Kelas ini membutuhkan komponen NavMeshAgent.

```
[SerializeField] protected AnimalState firstState;

[SerializeField] protected Transform player;
[SerializeField] protected NavMeshAgent agent;
[SerializeField] protected Animator[] animators;

[SerializeField] protected float idleTimer;
[SerializeField] protected string[] randomIdleAnim;

16 references
public bool playerInSight { get; set; }
```

Properties

- **firstState** (protected AnimalState):
  - Menentukan state awal dari hewan.
- **player** (protected Transform):
  - Referensi ke transformasi pemain.
- **agent** (protected NavMeshAgent):
  - Referensi ke komponen NavMeshAgent yang digunakan oleh hewan.
- **animators** (protected Animator[]):
  - Array yang berisi referensi ke komponen Animator untuk mengontrol animasi.
- **idleTimer** (protected float):
  - Timer yang digunakan untuk mengatur waktu idle.
- **randomIdleAnim** (protected string[]):
  - Array yang berisi nama-nama animasi idle yang dapat dipilih secara acak.
- **playerInSight** (public bool):
  - Menyimpan status apakah pemain terlihat oleh hewan.

Methods

```
12 references
protected virtual void AssignComponent()
{
    if (player == null)
        player = GameObject.FindWithTag("Player").transform;

    if (agent == null)
        agent = GetComponent<NavMeshAgent>();

    if (animators.Length == 0 || animators == null)
    {
        if(GetComponent<Animator>() != null)
        {
            animators = new Animator[1];
            animators[0] = GetComponent<Animator>();
        }
        else
        {
            animators = new Animator[transform.childCount];

            for (int i = 0; i < transform.childCount; i++)
            {
                animators[i] = transform.GetChild(i).GetComponent<Animator>();
            }
        }
    }
}
```

- **AssignComponent()** (protected virtual void):
    - Metode untuk meng-assign komponen yang dibutuhkan, seperti player, agent, dan animators. Jika tidak ada komponen yang diassign, metode ini akan mencoba mencarinya secara otomatis di GameObject yang sesuai.

```
23 references
public void PlayAnimation(string stateName)
{
    if (stateName == null)
    {
        Debug.LogError("No Animations to play, probably your state string is null");
        return;
    }

    foreach (var item in animators)
    {
        item.Play(stateName);
    }
}
```

- **PlayAnimation(string stateName)** (public void):
    - Memainkan animasi berdasarkan nama state yang diberikan. Jika tidak ada animasi yang tersedia, maka akan memunculkan pesan error.

```
9 references
public void PlayRandomAnimations(string[] stateNames)
{
    if(stateNames == null)
    {
        Debug.LogError("No Animations to play, probably your Array is null");
        return;
    }

    int x = Random.Range(0, stateNames.Length);
    foreach (var item in animators)
    {
        item.Play(stateNames[x]);
    }
}
```

- **PlayRandomAnimations(string[] stateNames)** (public void):
    - Memainkan animasi secara acak dari array stateNames. Jika array kosong atau null, maka akan memunculkan pesan error.

AnimalBaseState.cs

```
using UnityEngine;
using UnityEngine.AI;

namespace AIStateMachine.Animal
```

Namespace

- **namespace AIStateMachine.Animal**
  - Berisi kelas dasar untuk semua state AI hewan.

```
43 references
public abstract class AnimalBaseState : BaseState<AnimalState>
```

Class

- **AnimalBaseState**
  - Kelas abstrak yang mewarisi BaseState<AnimalState>. Digunakan sebagai basis untuk semua state hewan dalam state machine.

```
21 references
public AnimalBaseState(AnimalStateMachine stateMachine, Transform player, AnimalState key) : base(key)
{
    this.stateMachine = stateMachine;
    this.player = player;
}
```

Konstruktor

- **AnimalBaseState(AnimalStateMachine stateMachine, Transform player, AnimalState key)**:
  - Menginisialisasi state dengan referensi ke stateMachine, player, dan key (state yang diwakili).

```
53 references
protected AnimalStateMachine stateMachine { get; private set; }
7 references
protected Transform player { get; private set; }

protected bool isJumping = false;
protected Vector3 startPos;
protected Vector3 endPos;
protected float jumpHeight = .5f;
protected float jumpDuration = 1f;
protected float jumpTimer = 0.0f;
```

Properti

- **stateMachine** (protected AnimalStateMachine):
  - Referensi ke state machine dari hewan yang sedang dikontrol oleh state ini.
- **player** (protected Transform):
  - Referensi ke transformasi pemain.
- **isJumping** (protected bool):
  - Menyimpan status apakah hewan sedang melompat.
- **startPos, endPos** (protected Vector3):
  - Menyimpan posisi awal dan akhir dari lompatan.
- **jumpHeight** (protected float):
  - Menentukan tinggi lompatan.
- **jumpDuration** (protected float):
  - Menentukan durasi lompatan.
- **jumpTimer** (protected float):
  - Timer untuk mengelola proses lompatan.

Metode

```
3 references
public override void OnTriggerEnter(Collider other)
{
    stateMachine.playerInSight = other.transform.Equals(player);
}
```

- **OnTriggerEnter(Collider other)** (public override void):
    - Memeriksa apakah pemain masuk dalam radius trigger. Jika iya, properti playerInSight akan diatur ke true.

```
2 references
public override void OnTriggerExit(Collider other)
{
    stateMachine.playerInSight = !other.transform.Equals(player);
}
```

- **OnTriggerExit(Collider other)** (public override void):
    - Memeriksa apakah pemain keluar dari radius trigger. Jika iya, properti playerInSight akan diatur ke false.

```
4 references
protected void PatrolPattern(NavMeshAgent agent, float radius)
{
    Vector3 random = Random.insideUnitSphere * radius;
    random += stateMachine.transform.position;

    if (NavMesh.SamplePosition(random, out NavMeshHit hit, radius, agent.areaMask))
        agent.SetDestination(hit.position);
}
```

- **PatrolPattern(NavMeshAgent agent, float radius)** (protected void):
    - Mengatur pola patroli acak untuk agen AI. Menghasilkan posisi acak dalam radius tertentu dan mengarahkan agen ke posisi tersebut.

```
5 references
protected void RunawayPattern(NavMeshAgent agent)
{
    Vector3 bestPoint = Vector3.zero;

    int numChecks = 8;
    float maxDistance = -1f;
    float angleIncrement = 360f / numChecks;

    for (int i = 0; i < numChecks; i++)
    {
        float angle = i * angleIncrement;

        Vector3 direction = Quaternion.Euler(0, angle, 0) * Vector3.forward;
        Vector3 potentialPosition = stateMachine.transform.position + direction * 5f;

        if (NavMesh.SamplePosition(potentialPosition, out NavMeshHit hit, 5f, agent.areaMask))
        {
            float distanceFromPlayer = Vector3.Distance(hit.position, player.position);
            if (distanceFromPlayer > maxDistance)
            {
                bestPoint = hit.position;
                maxDistance = distanceFromPlayer;
            }
        }
    }

    if (maxDistance > -1f)
    {
        agent.SetDestination(bestPoint);
    }
    else
    {
        Vector3 directionFromPlayer = stateMachine.transform.position - player.position;
        Vector3 runToPosition = stateMachine.transform.position + directionFromPlayer.normalized * 5f;

        NavMesh.SamplePosition(runToPosition, out NavMeshHit hit, 5f, agent.areaMask);

        agent.SetDestination(hit.position);
    }
}
```

- **RunawayPattern(NavMeshAgent agent)** (protected void):
    - Mengatur pola lari menjauh dari pemain. AI akan mencari posisi terjauh dari pemain dan mengarahkan agen ke posisi tersebut. Jika tidak ada posisi yang ditemukan, AI akan bergerak menjauh dari pemain dalam garis lurus.