Candice AI for Games

User Manual

Contents

Getting Started

In this system, there are three main parts.

- Candice AI Manager
  - This houses all the higher level logic, such as calculating shortest paths using Pathfinding algorithms.
  - It also handles the registration of AI agents and issues each agent a unique Agent ID.
  - An instance of Candice AI Manager is required for **Pathfinding**, **Obstacle Avoidance** and **Object Detection** functionality.
- Candice AI Controller
  - This script controls the actual AI agents. It supports Finite State Machines and Behaviour tress and even includes an attack and Damage System.
- Simple AI Controller
  - This basic AI logic such as moving from one point to another, destroying on collision etc. It is best used for bullets or missile type systems. Normally, it would be attached to the projectiles fired by the AI Agent.
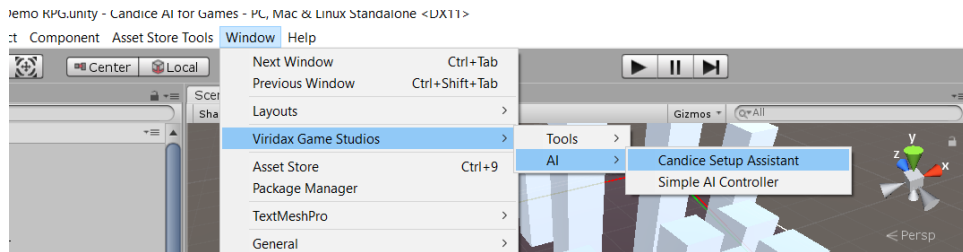
Candice AI Controller

Before you begin using the AI Controller script, make sure you have done the following first:

1. If your agent uses animations, then ensure you have for Idle, Running (and/or walking), receiving damage and attacking. Idle should be the default animation state.
2. **Note: It will still work without animations. Just ensure that "has Animations" is unchecked in the AI Settings tab.**
3. Set the triggers and Boolean conditions accordingly:
   a. You can use the conditions in the script or use your own.
   b. If you use your own conditions, you will have to change them in the script as well. (Where it says anim.SetBool() and anim.SetTrigger())

Adding the Components

In order to add the controller, click on **Window -> Viridax Game Studios -> AI -> Candice AI Manager Window -> Viridax Game Studios -> AI -> Candice Setup Assistant,** or **Window -> Viridax Game Studios -> AI -> Simple AI Controller**.

When adding the AI Controller, a Setup Assistant will popup, allowing you to either quickly generate an instance of Candice AI Manager or create a Candice AI Controller.

Using the Controllers

Simple AI Controller

This script is straight-forward and easy to use. Once the script has been added to the game object, assign a target and it will automatically move to the target. Any game object can be assigned as a target.

▼ 🟩 ☑ **Box Collider**   🔲 🔧 ⚙,

	🔼 Edit Collider
Is Trigger	☐
Material	None (Physic Material) ⊙
Center	X 0 Y 0 Z 0
Size	X 1 Y 1 Z 1

▼ 🔷 ☑ **Simple AI Controller (Script)**   🔲 🔧 ⚙,

Script	⊡ SimpleAIController ⊙
Target	🔷 Player ⊙
Rb	None (Rigidbody) ⊙
Follow Target	☑
Obstacle Avoidance	☑
Speed	10
Stopping Distance	0
Destroy On Collision	☑
Colision Delay	3
Destroy After Delay	☐
Delay	3
Stop On Collision	☐
Orbit	☐
Fire True	☑

▼ ⚛ **Rigidbody**   🔲 🔧 ⚙,

Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	☑
Is Kinematic	☐
Interpolate	None ⬍
Collision Detection	Discrete ⬍
▼ Constraints	
Freeze Position	☐ X ☐ Y ☐ Z
Freeze Rotation	☑ X ☐ Y ☑ Z

Candice AI Controller

The AI Controller works best with animations. However, if you do not have animations, uncheck Has Animations in the AI Settings tab, as well as **Can Attack** in the Combat tab in order to prevent a bug when it tries to attack.

*Note: you will require some basic knowledge on how to set up animator controllers with animations.

To use the script with animations, ensure you have animations for Idle, Walk, Run, Dame and Attack.

Make sure walking and running animations have "Loop Time" and "Loop Pose" enabled in order for the animations to loop seamlessly. Also, depending on what animations you use, you might have to check some other settings and also ensure that the 3D model is using the correct rig (usually it is Humanoid).

When you add the script from the Menu, it will automatically add a Rigidbody, a Collider and Animator. After that, follow these steps:
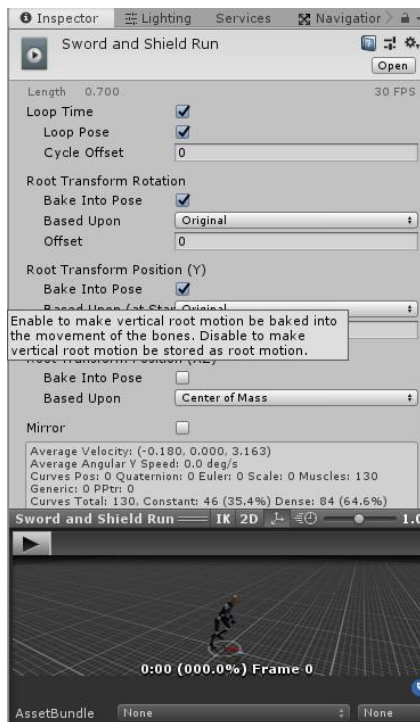
1. In the Animator component, uncheck "Apply Root Motion."
2. In the Rigidbody, ensure that "Use Gravity" is checked.
3. Also in the Rigidbody go to the constraints section and check Freeze Rotation for X and Z.
4. In the Collider, adjust the height, radius and centre according to your character.
5. These settings are just a guideline and you can play around with them to suit your situation.

Note: The Setup Assistant does these things automatically.
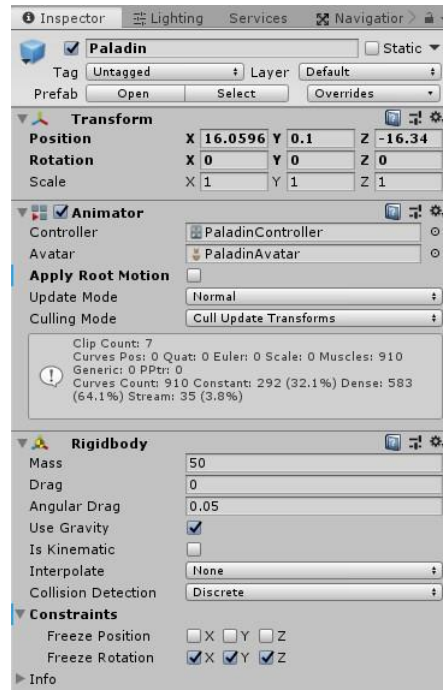
Once all of these have been set up, we can now set up the animations. This animation controller set up works with the default Finite State Machine implementation.

Default FSM Implementation

1. Create a new Animator Controller and add it to the Controller section in the Animator. In the case of the demo scene it is **Paladin Controller**.
2. Double click on your Animator Controller and then drag all the required animations into it.
3. By now you should have something looking similar to this:
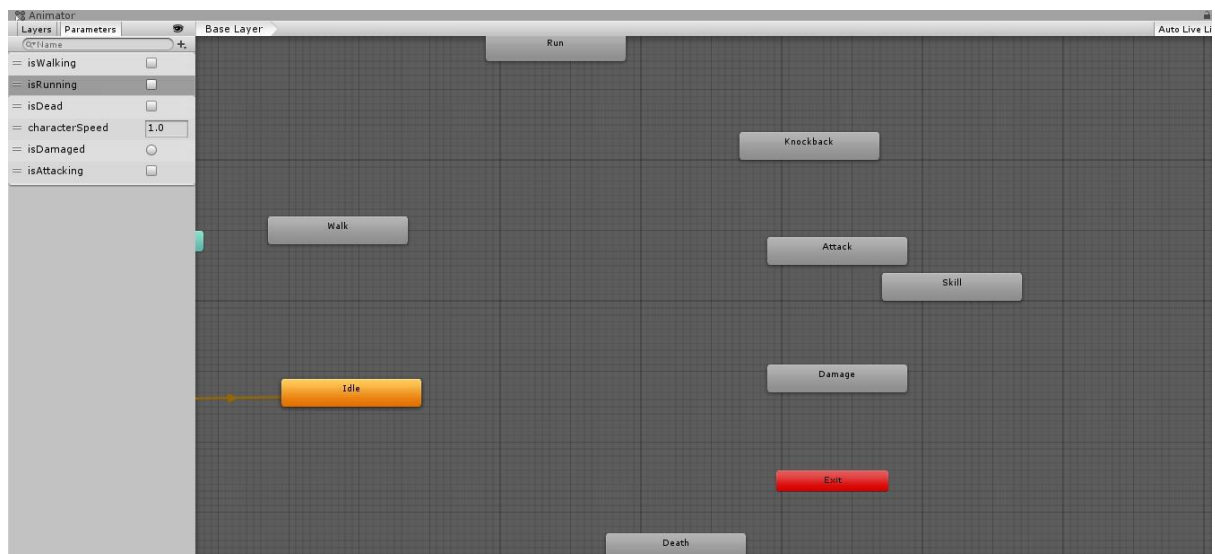
4. You can now add transitions between the different animations. You will need the following conditions:

a. isWalking (Boolean), isRunning (Boolean), isDead (Boolean), isDamaged (Trigger), isAttacking (Boolean).

b. The demo scene will show you where each transition needs to be and with what condition. Also take note that some transitions have "Has Exit Time" unchecked.

5. Your animator controller should look something like this by now:
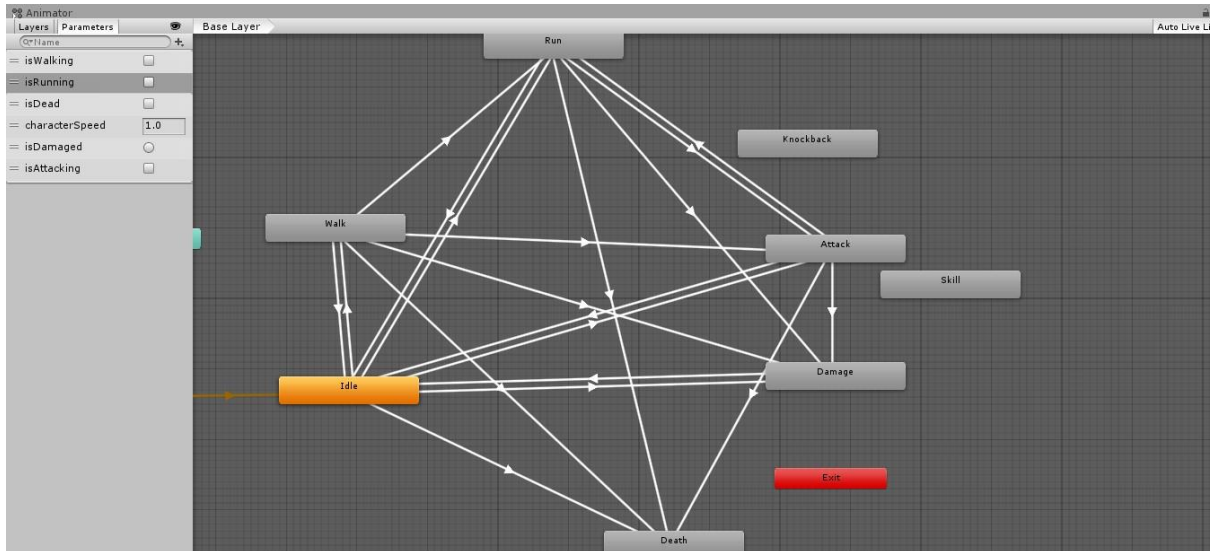
*Note: The AI Controller uses Behaviour Tress by default. So you will have to set up one of you own and attach it to the Game Object. A more detailed explanation can be found under the features section.

Features

Character Class

The Basic AI Controller inherits from a class called "Character". This class stores information/attributes about the character such as Strength, Intelligence, Faith, Attack Damage and more. It also keeps track of the Player Level.

Character Stats classes

There are several statistics but all of them are controlled by 3 primary attributes:

- Strength.
- Intelligence.
- Faith.

Each of these attributes directly affect three forms of damage.

- Physical Damage (Strength)
- Magical Damage (Intelligence)
- Spiritual Damage (Faith)

Levelling Up

The Character Stats class supports levelling up by calling the LevelUp() method. This will automatically adjust the character attributes. This feature enables the AI characters to level up and become stronger as the game progresses.

Finite State Machines

The controller supports different states that the character can be in, using Finite State Machines. The states are

- Idle
- Follow
- Attack
- Patrol

By default, the controller should change between states automatically. More states can be easily added.

*Note: The character can only be in one state at a time. <mark>Finite state machines are the default implementation if the agent is controlled by the player.</mark>

In order for a state to work, you must create an Action class that inherits from (FSMAction) and then assign it to the state you want. For example, create a MoveAction Action that can be assigned to the Follow State and Patrol State (since they both require the agent to move).

Patrol State

For the character to patrol between several points, add the required Patrol Points to the controller in the inspector:

The patrol points can be anything from a simple empty GameObject to more complex things. Make sure each Patrol Point is tagged as 'PatrolPoint'.

Attack State

The AI unit can also attack and deal damage to objects around. You will need to use an animation Event to call the Attack() method in the script. To do so, follow these steps:

1. Select the 3D Model, in this case it is **Paladin**.
2. Select the **Animation Window**.
3. Select the Attack animation.
4. Add an animation Event at the point where you feel the damage should be dealt. Usually where the sword comes down and makes contact with the player.
5. Choose Attack() as the function to call.
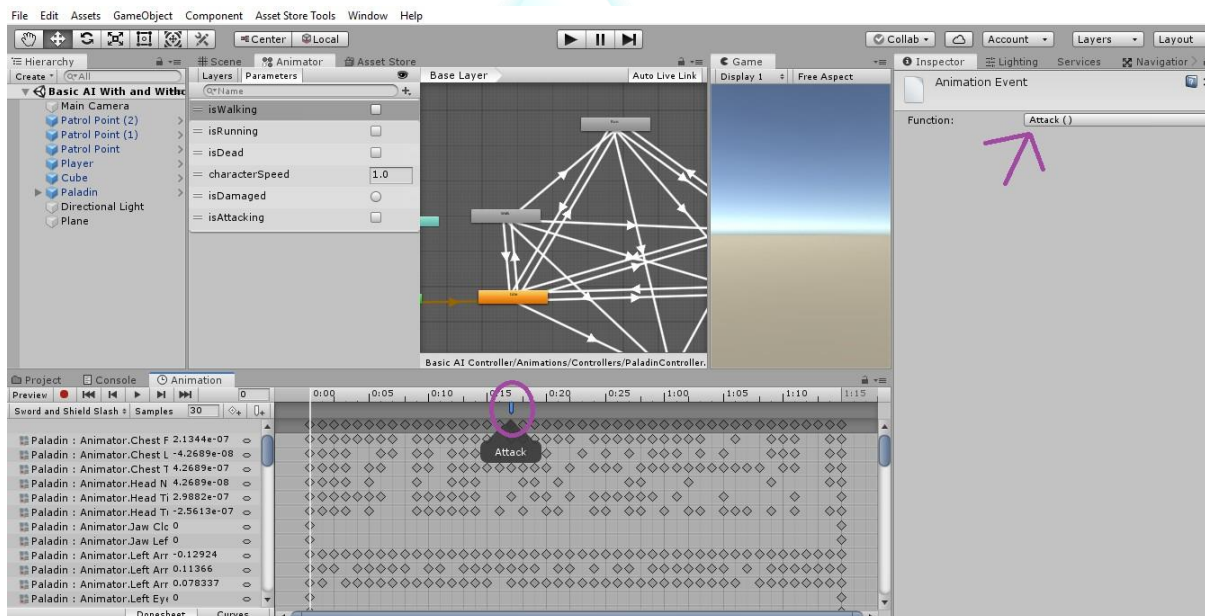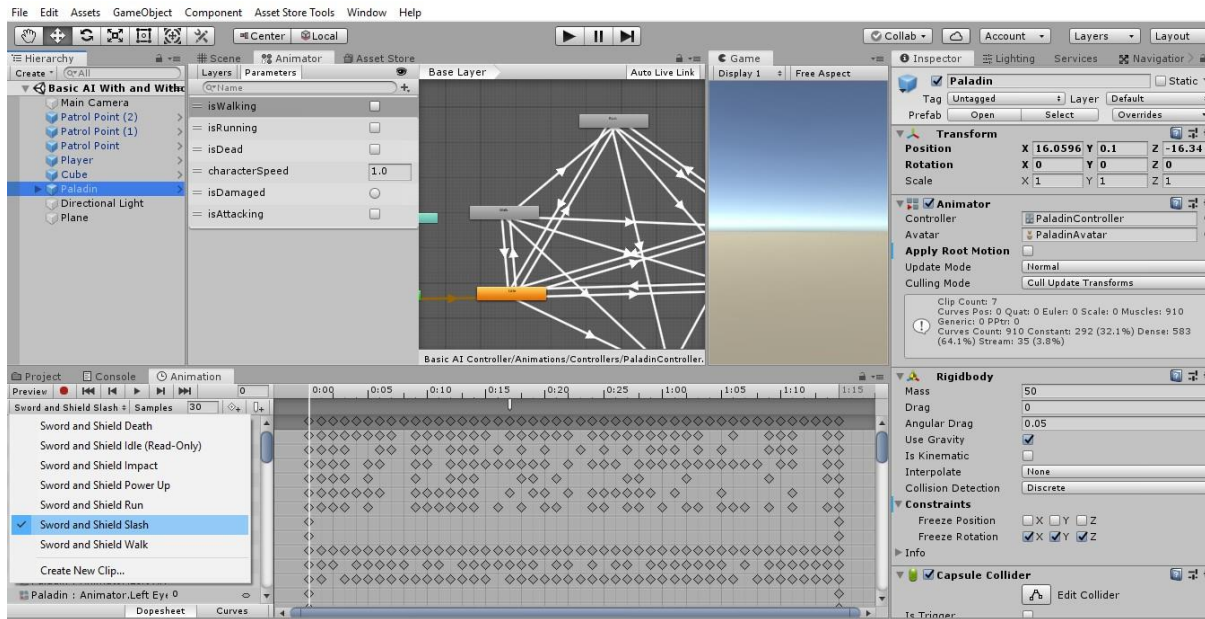
Behaviour Trees

The AI Controller also supports behaviour trees. They are the default implementation if the agent is AI controlled.

You will have to create a script that inherits from MonoBehaviour. In the script, you will have to create your various nodes, starting from the root node. This will require a bit of knowledge on how behaviour trees work. A link is provided at the end explaining Behaviour Trees. You can also Download Candice Behavior Designer here and easily create entire Behavior Trees while just dragging and dropping.

Damage System

The Damage System works by sending a Raycast as a sphere around the character. If it collides with an object marked with any of the Enemy Tags, a message will be sent to the enemy object in

order to receive damage. Ensure that the receiving object has a method with the name ReceiveDamage(). It should have the following method signature

Public void ReceiveDamage(float damage) { }

Head Look

The Basic AI Controller also has a Head Look feature. This allows the character to dynamically look at any object in the game. To use this feature follow these steps:

1. Click on the Movement tab.
2. Check the "Enable Head Look" option.
3. Select a Target, you can drag and drop any gameobject into the slot.

   The intensity determines how intensely the character will focus on the object.
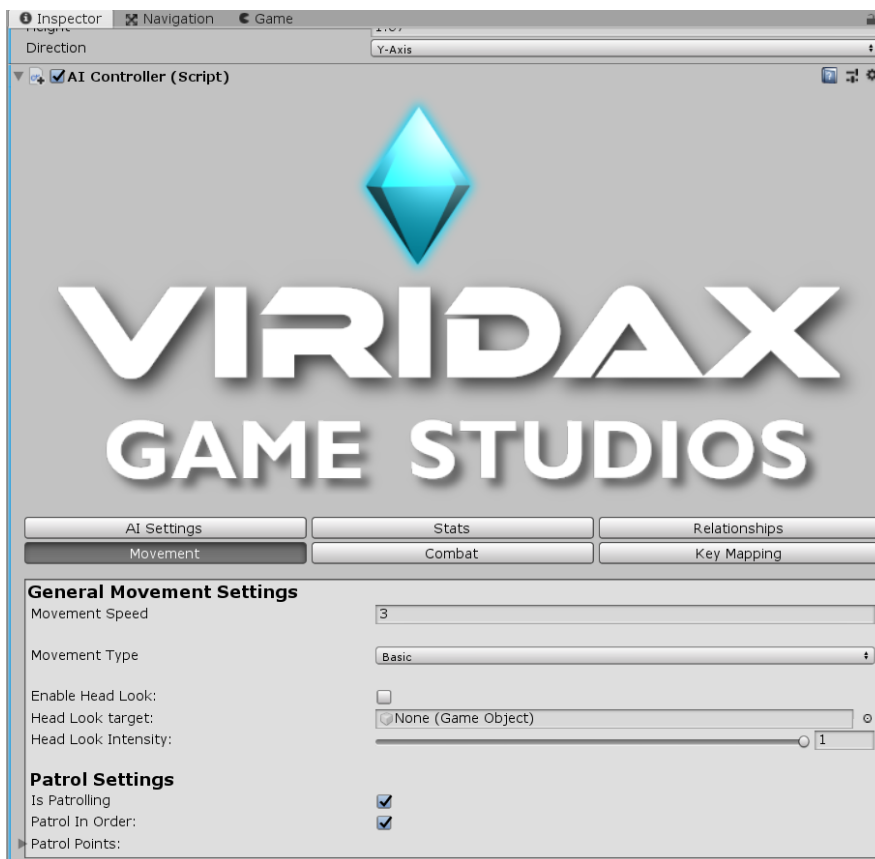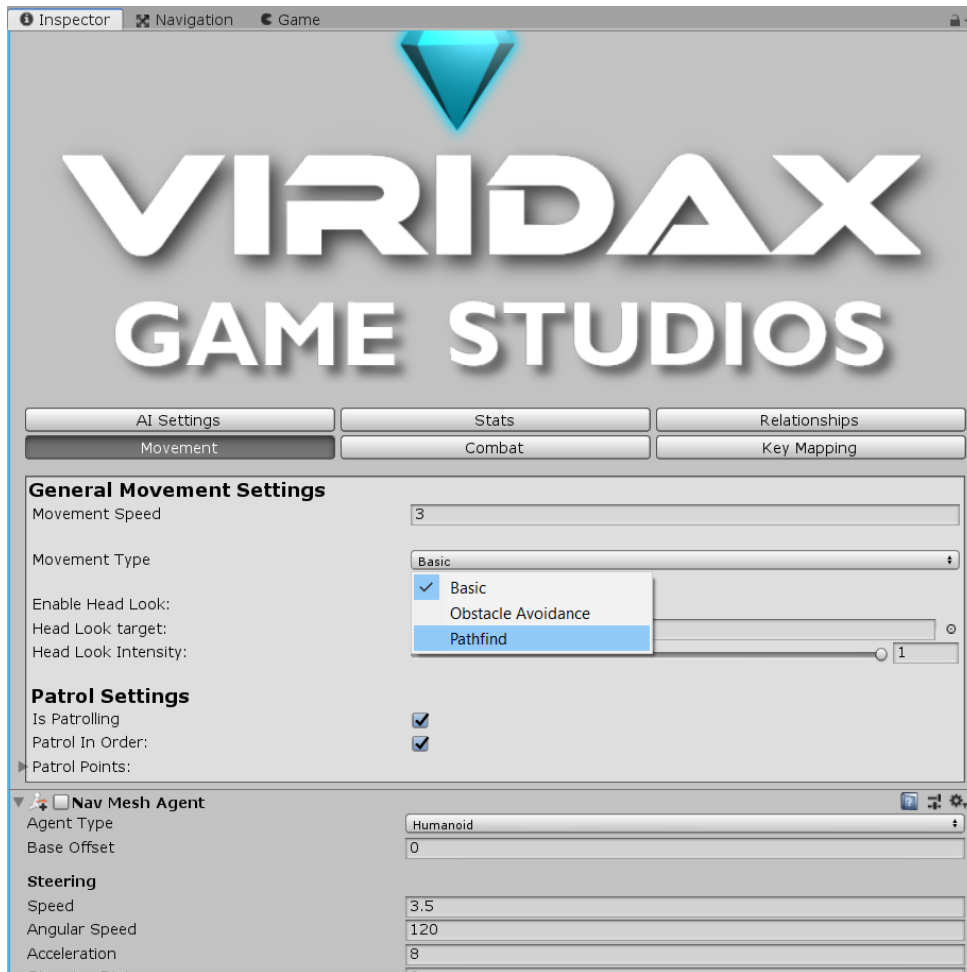

Movement

The AI controller supports three types of movement:

- Simple.
- Obstacle Avoidance
    - o This evades immediate obstacles.
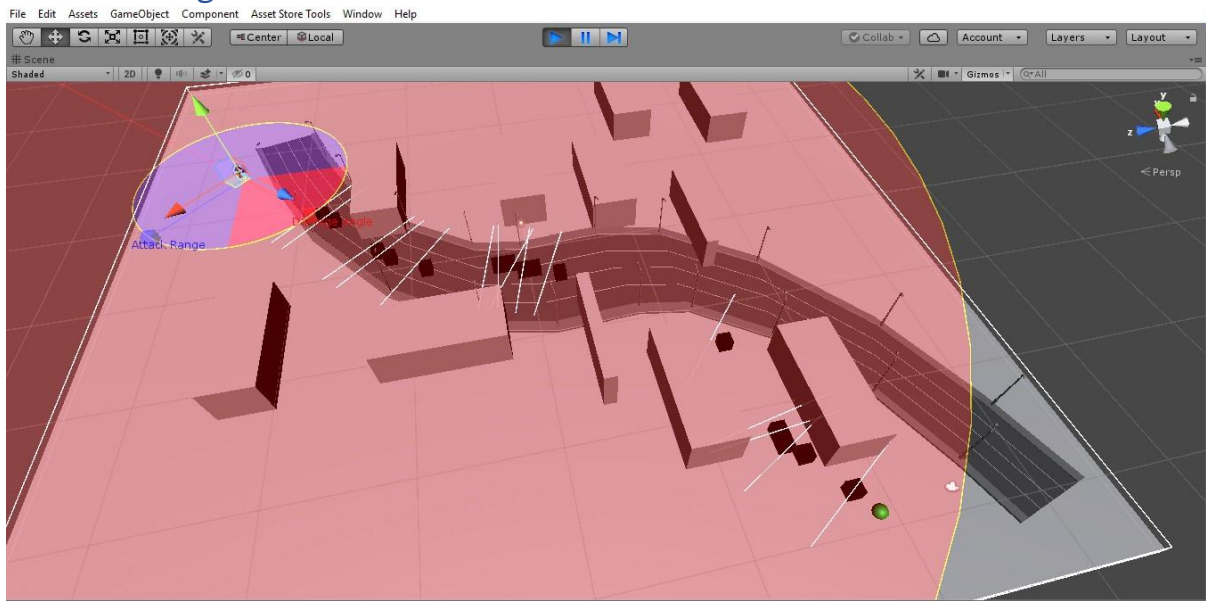- Pathfinding

o   This uses the A* algorithm to find the shortest path to the target.

Obstacle Avoidance

This allows the AI to evade immediate obstacles, within a certain range. This algorithm is most efficient in straight line paths with a few random obstacles. However, it can also be used in more complex scenes, but efficiency is not guaranteed.

A* Pathfinding

This algorithm allows the agent to find the shortest path to the target.

To make it work in your own scenarios, follow these steps:

1. Ensure you have an Candice AI Manager script attached to a GameObject
   a. Click on **Window -> Viridax Game Studios -> AI -> Candice AI Manager and it will automatically create one for you, with a Grid component as well**.
2. Tweak the Grid settings to fit your terrain and define an unwalkable mask. Unwalkable mask is the Layer which the algorithm will define as unwalkable, and thus the AI will never walk on terrain/gameobjects with that Layer. (e.g a layer with the name "Obstacles")
3. Define Walkable Regions.
   a. For each walkable region, you must specify a Terrain Mask (basically a Layer that will be assigned to a terrain/gameobject) and a movement penalty for that terrain. This allows the AI to "prefer" certain terrain types over others. E.g, it might prefer moving on the road layer as opposed to the sand layer.

Now you can call **StartFindingPath()** and **StopFindingPath()** respectively and it will automatically calculate the best path and move the agent.

Editor and SceneView

Basic A.I Controller also has powerful editor and SceneView controls. You can adjust the properties including attack range and detection radius right from the SceneView. You can also

adjust all other properties, like enemy tags, move speed and more from the inspector. There's no need to touch the script if you don't want to.
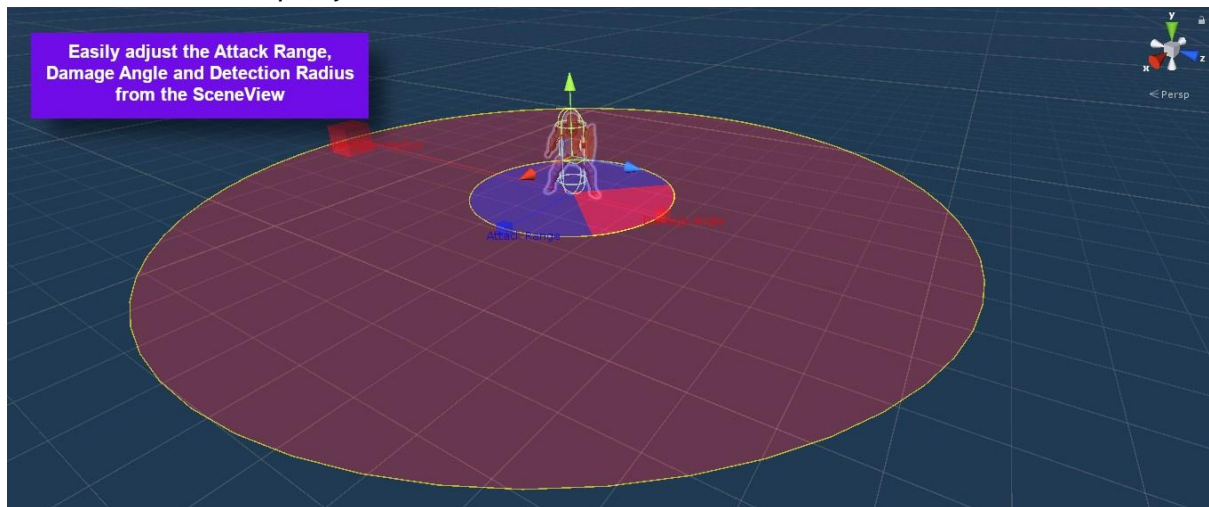
Support

We at Viridax Game Studios strive to provide the best AAA quality video game script assets to the world. If you have any issues with this asset, suggestions, or just want to send some nice feedback, then feel free to contact us on one of our channels.

Support email: natubeast@gmail.com

Discord Server: https://discord.gg/GUtK6EH

Facebook: https://www.facebook.com/viridaxgamestudios/

References

Item	Author	Website
Paladin 3D Model	Mixamo	https://www.mixamo.com
Sword and Shield Animations	Mixamo	https://www.mixamo.com
Erika 3D Model	Mixamo	https://www.mixamo.com
Behavior Trees		https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php
Candice Behavior Designer		https://assetstore.unity.com/packages/slug/168218