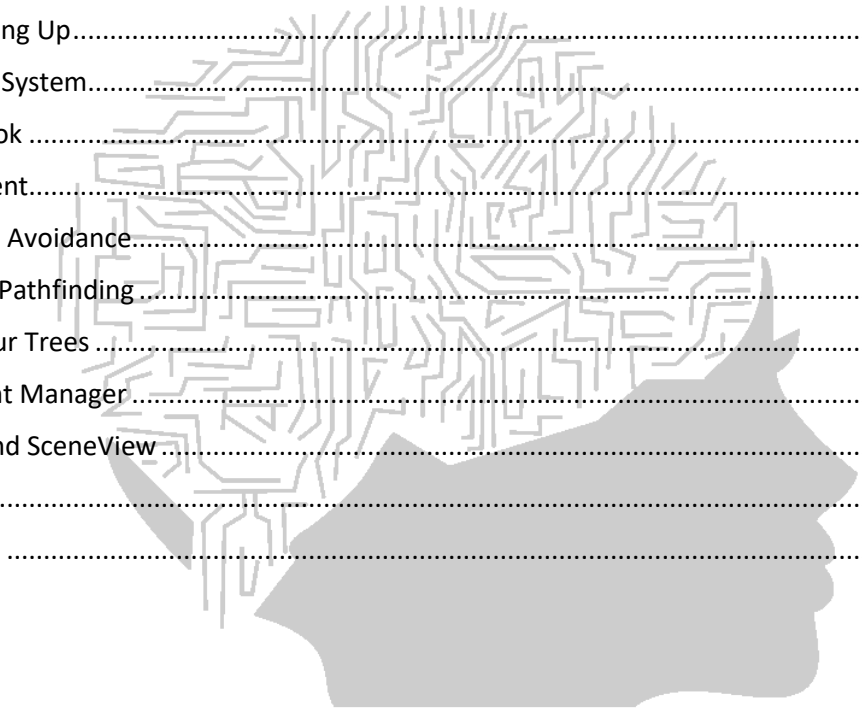
CANDICE AI

for

Games

User Manual

Contents



CANDICE

AI

Introduction

Thank you for downloading this plugin for Unity3D.

Candice AI for Games is a development tool that controls all aspects of your video game AI. You can create intelligent AI that are suitable for role playing games, racing games, tactics style games and more.

The asset also supports AI for both 2D and 3D.

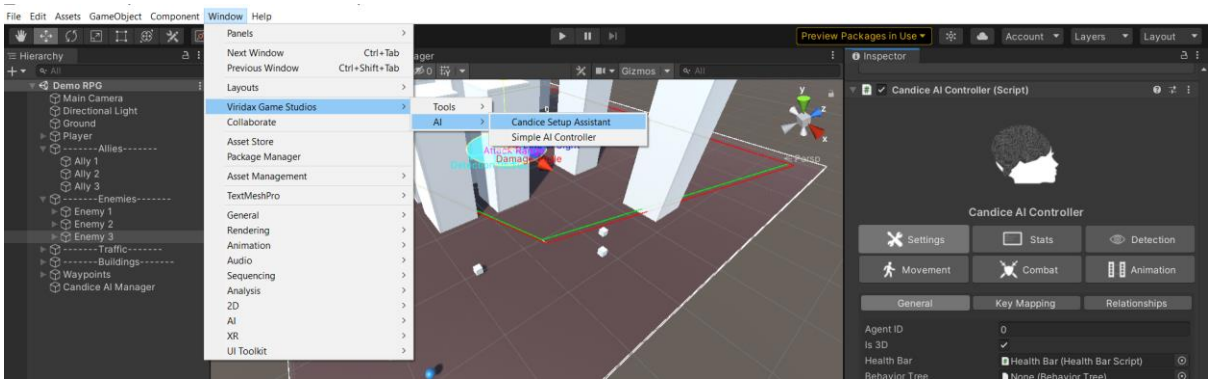
Getting Started

In this system, there are three main parts.

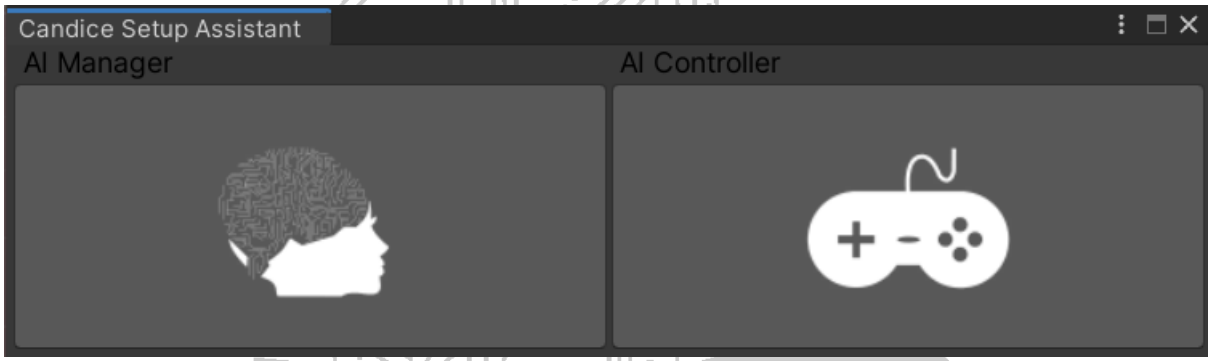
- Candice AI Manager
    - This houses all the higher level logic, such as calculating shortest paths using Pathfinding algorithms.
    - It also handles the registration of AI agents and issues each agent a unique Agent ID. It maintains a list of all registered agents that can be accessed from other classes.
    - An instance of Candice AI Manager is required for **all agents** to function normally.
    - Only one instance of Candice AI Manager is allowed per scene.
- Candice AI Controller
    - This script controls the actual AI agents. It is equipped with **Object Detection**, **Obstacle Avoidance**, **Behavior Trees** and more.
- Simple AI Controller
    - This script contains basic AI logic such as moving from one point to another, destroying on collision, orbiting etc. It is best used for bullets or missile type systems. It can even be used to simulate planets orbiting a sun. Normally, this script would be attached to the projectiles fired by the AI Agent.

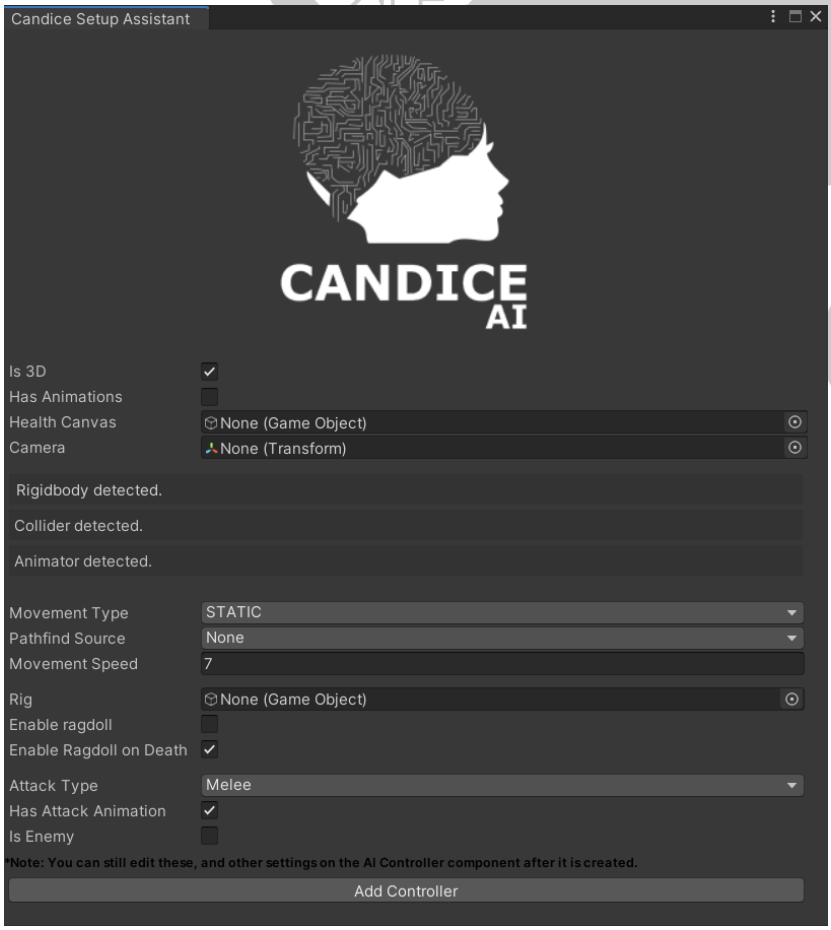
Candice Setup Assistant

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

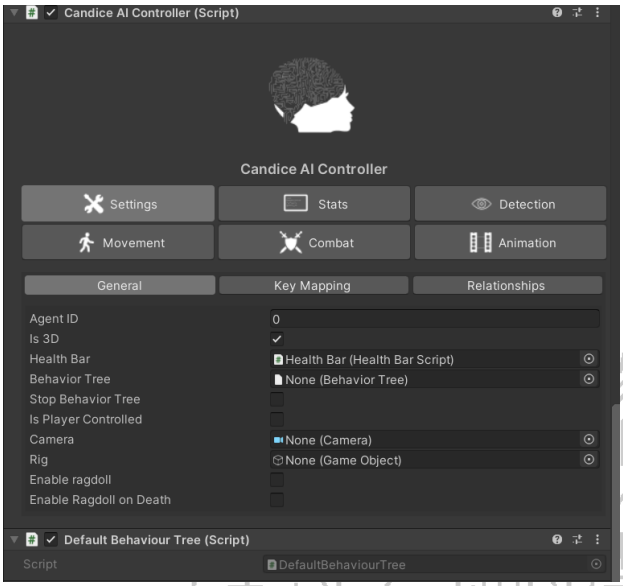


When adding the AI Controller, a Setup Assistant will popup, allowing you to either quickly generate an instance of Candice AI Manager or create a Candice AI Controller.

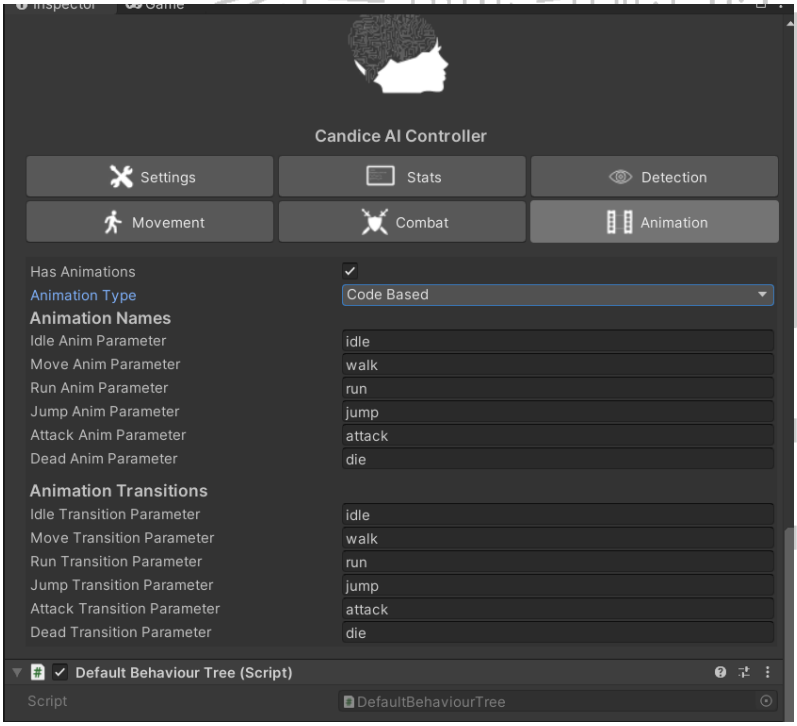




Animations



The AI Controller can work with animations. However, if you do not have animations, uncheck Has Animations in the **Animation tab**.



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

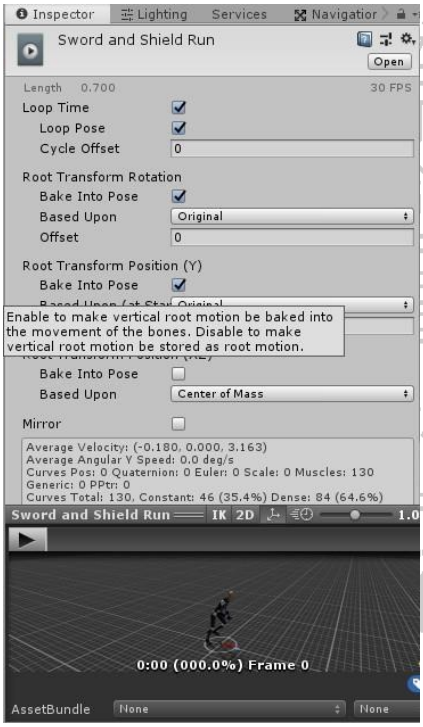
There are two main Animation Types

- Code Based
    - The Controller will call the animations by name. No transitions are needed.
- Transition Based

- The Controller will access the different animation states by triggering the different transitions.

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

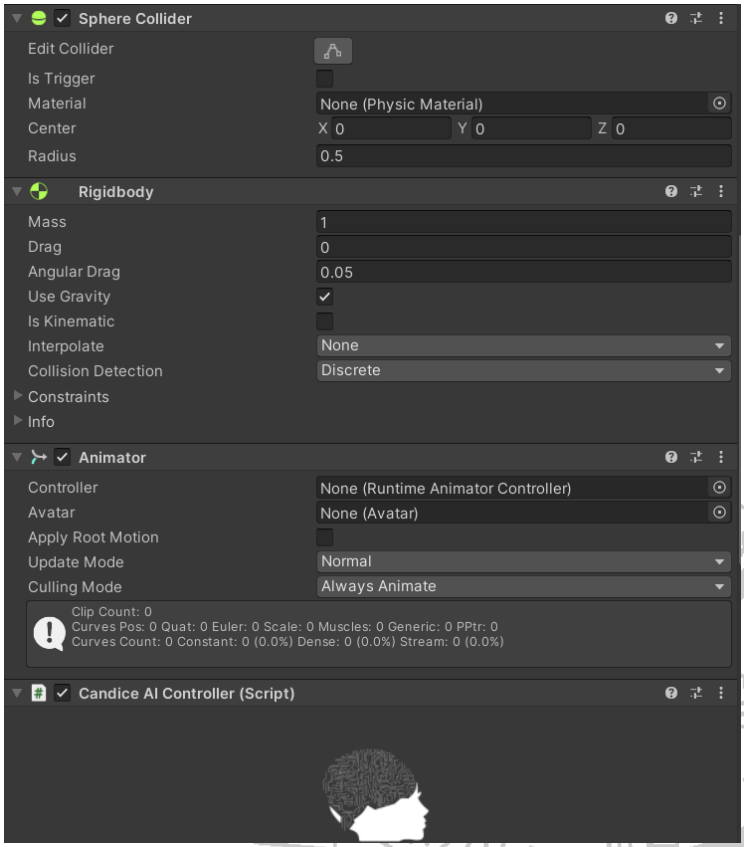
Make sure walking and running animations have "Loop Time" and "Loop Pose" enabled in order for the animations to loop seamlessly. Also, depending on what animations you use, you might have to check some other settings and also ensure that the 3D model is using the correct rig (usually it is Humanoid).



When you add the Controller independently of the Setup Assistant, you must 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, and optionally Y (depending on your type of game).
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.

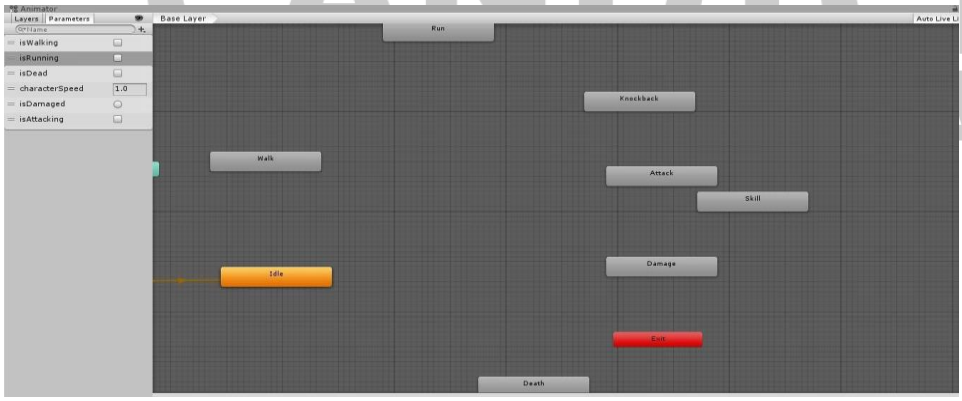
Note: The Setup Assistant does these things automatically.



Once all of these have been set up, we can now set up the animations.

Default Transition Based Animation 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:

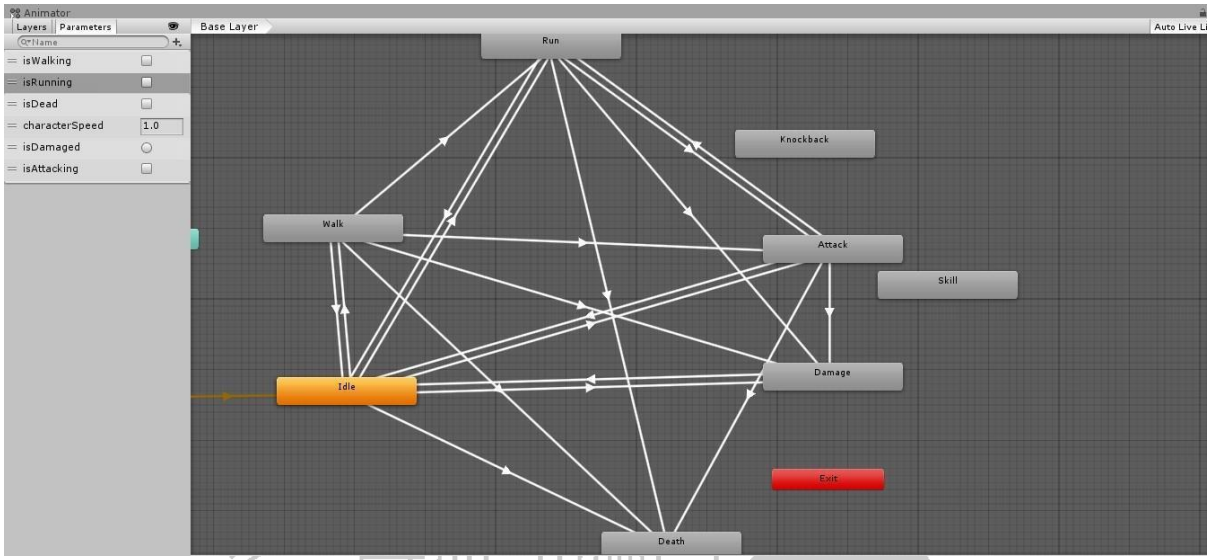


1. You can now add transitions between the different animations. You will need the following conditions:

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.

2. Your animator controller should look something like this by now:



For the Code Based Implementation, you just add your animations to the controller (with no transitions) and make sure to set the Animation Name Parameters in the Animation Tab to match the names of the animations in the controller.

***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. However, a default Behavior Tree is provided that should cover most basic AI needs.**

Features

Character Class

The Candice 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 Stat class

- This class controls each individual attribute. It can be instantiated to represent any kind of attribute, such as Strength, Hitpoints, Intelligence etc.
- It also supports Stat Modifiers that can be applied from different sources, such as items, armour, weapons and spells.

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. Finite state machines are the default implementation if the agent is controlled by the player.**

In order to create your own actions, 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

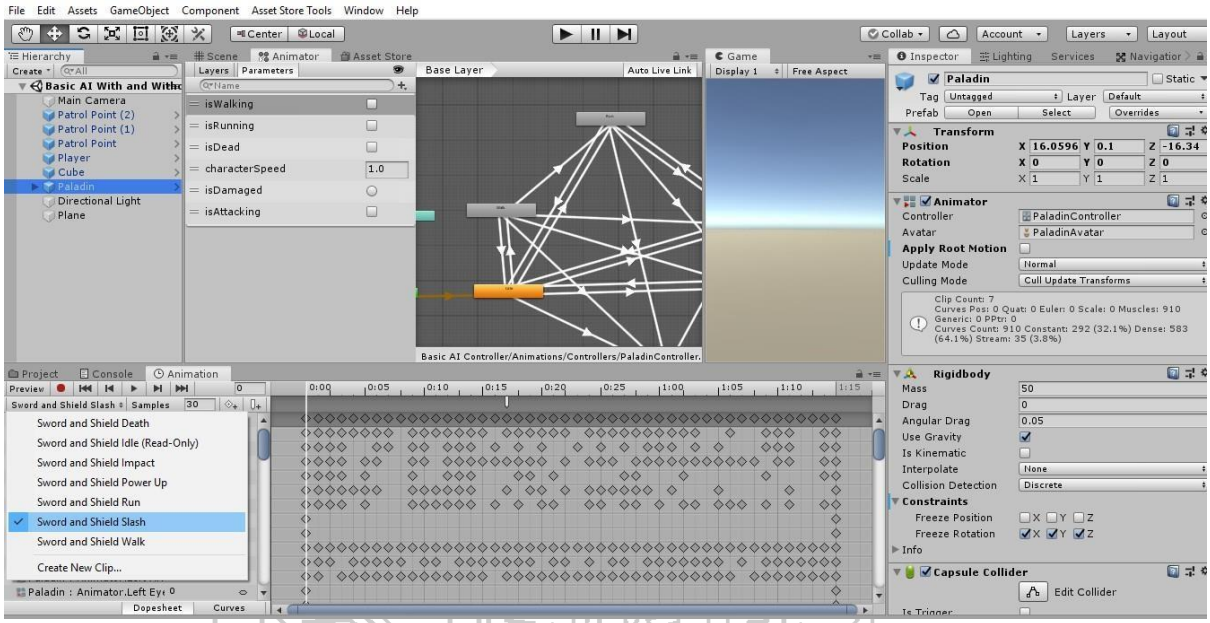
For the character to patrol between several points, add the required Patrol Points to the controller in the inspector:

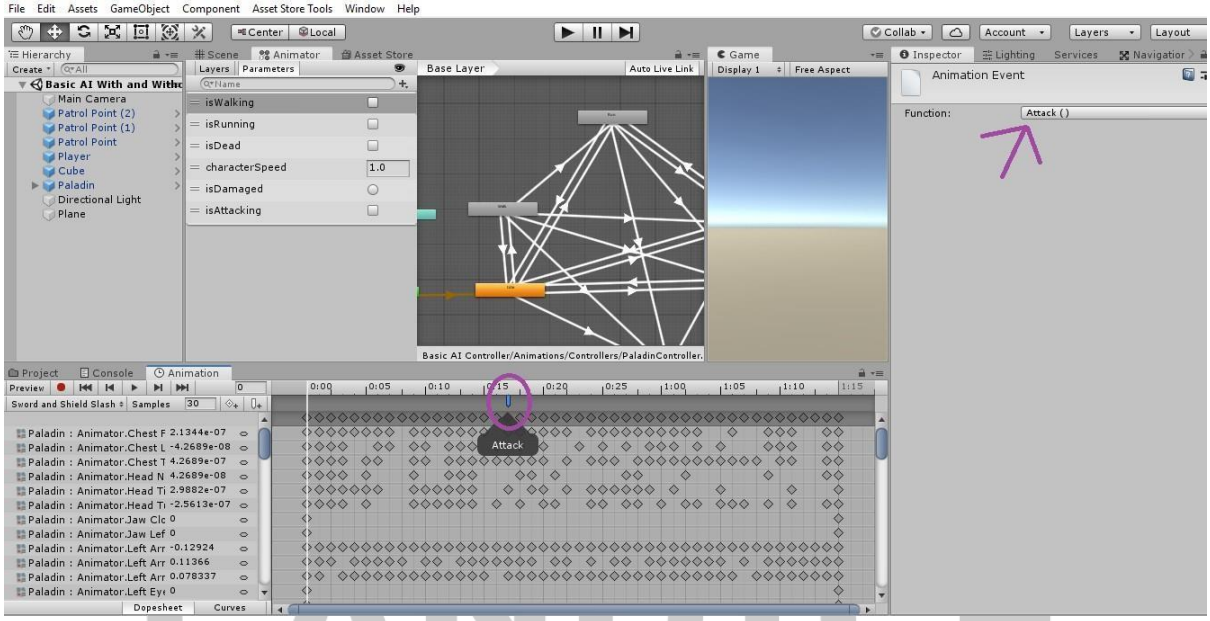
The patrol points can be anything from a simple empty GameObject to more complex things. Make sure each Patrol Point is tagged as 'PatrolPoint'. The system also supports Waypoints as a form of patrolling.

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.





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) { }

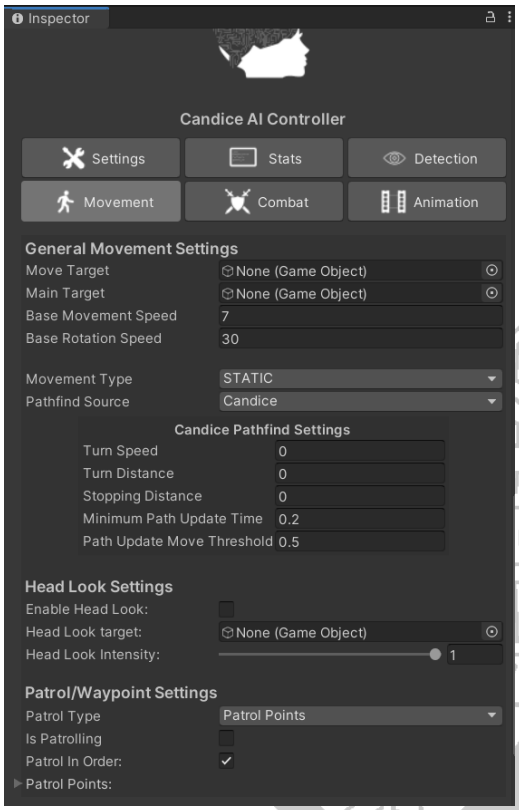
Head Look

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

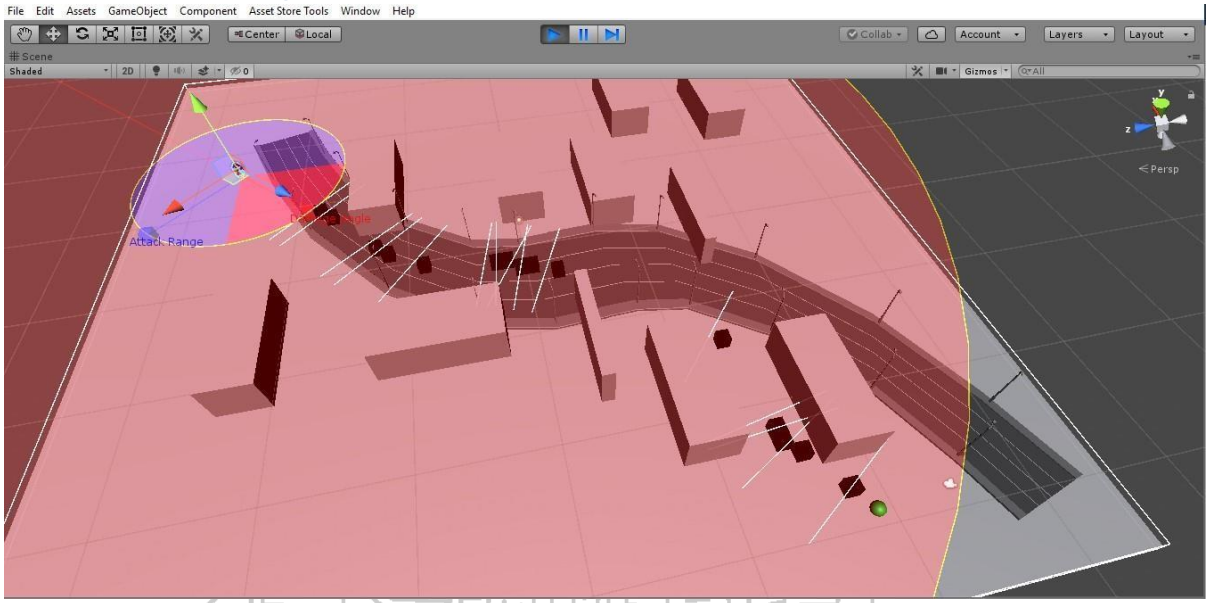
The AI controller supports three types of movement:

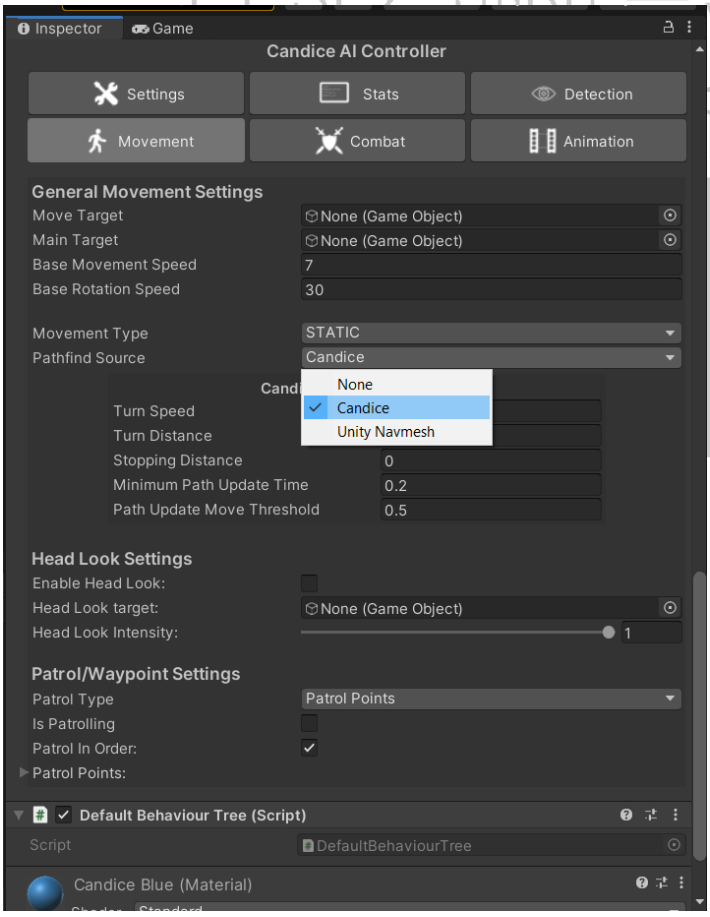
- STATIC
  - With this, the agent will blindly follow its target in a straight line.
- DYNAMIC
  - With this, the agent will utilise the **obstacle avoidance** module to evade immediate obstacles while advancing to the target.
- TILE-BASED
  - This is a special movement type that allows for tactics-style movement, found in games like chess.

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.

Candice Pathfinding





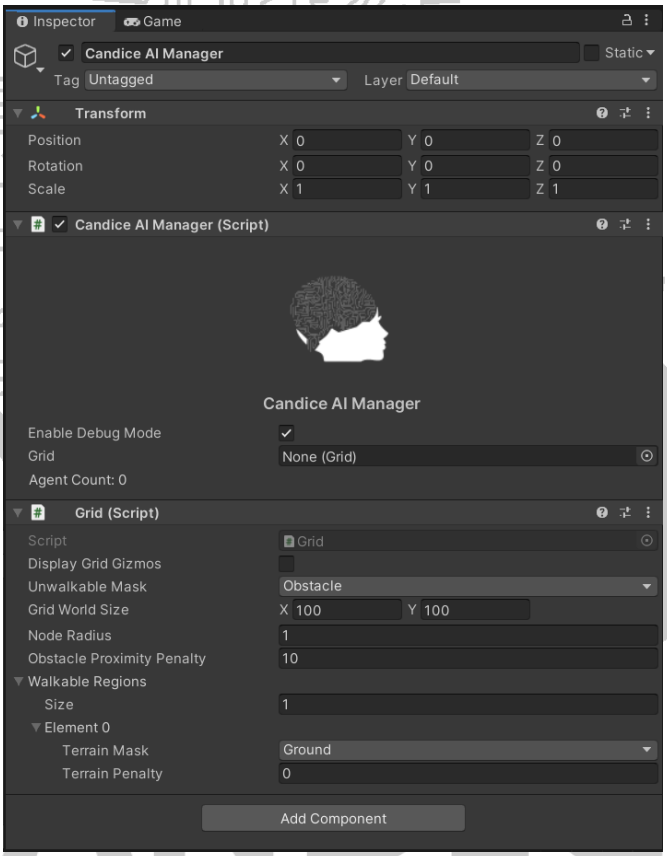
This algorithm allows the agent to find the shortest path to the target. The Controller also supports Unity Navmesh Pathfinding.

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

1. Ensure you have a **Candice AI Manage**r 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 agent 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 programmatically call **StartFindingPath()** and **StopFindingPath()** from the Candice AI Manager respectively and it will automatically calculate the best path and move the agent. However, theres no need to call these methods since the AI Controller does so automatically.

Behaviour Trees

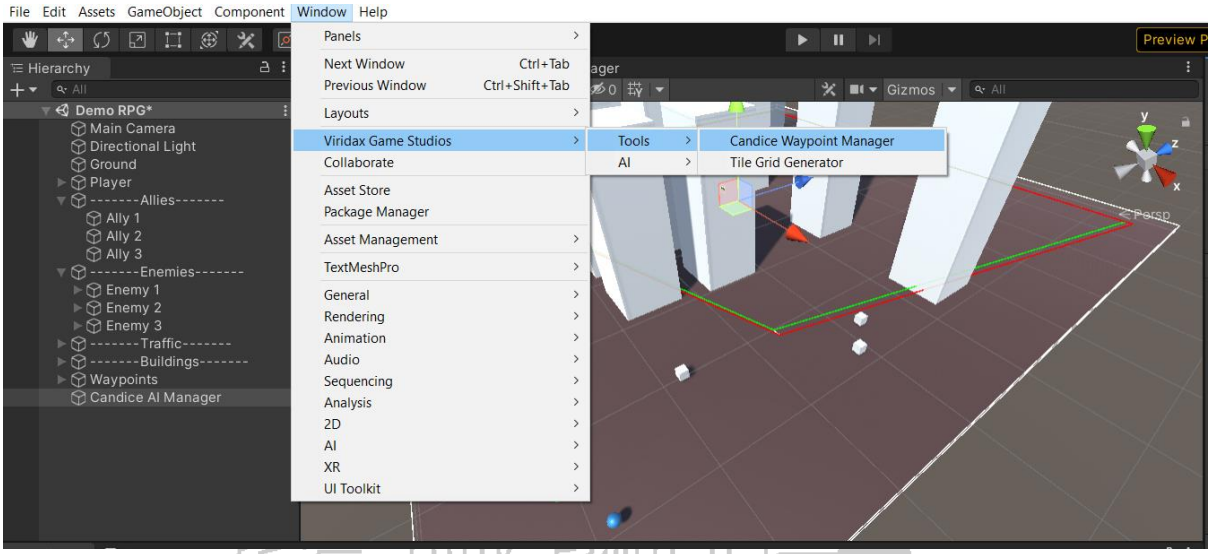
The AI Controller also supports behaviour trees. They are the default implementation if the agent is AI controlled.

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.
However, a default behavior tree script is provided that covers most basic functionality.

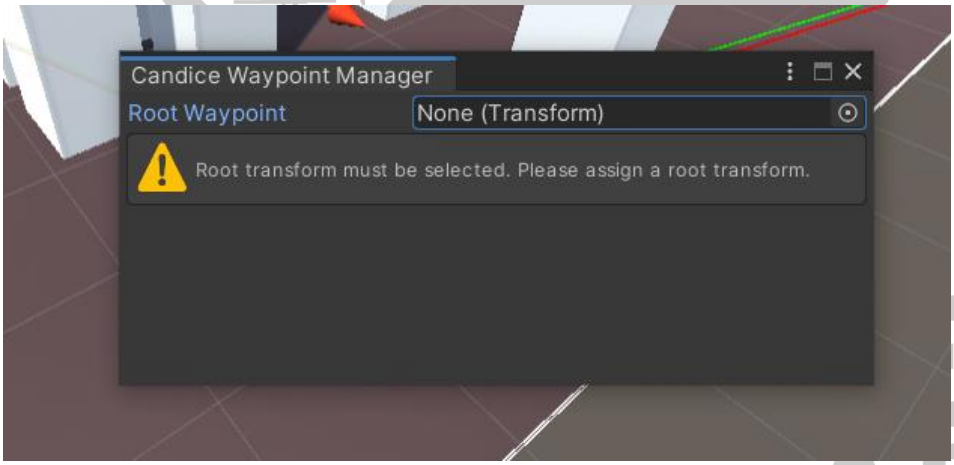
Waypoint Manager

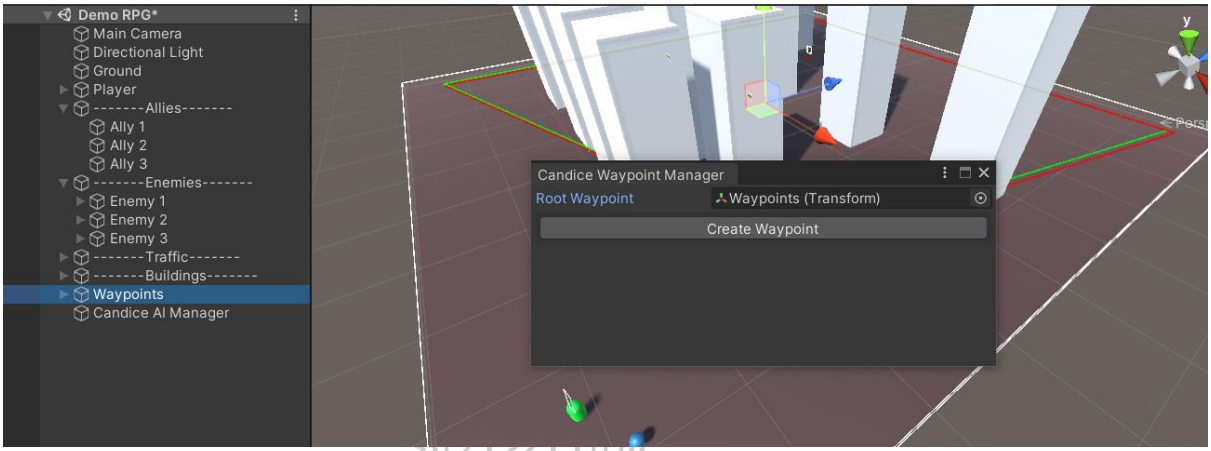
The Waypoint Manager allows you to create Waypoints that the agent can follow. Each waypoints also supports branching into other waypoints, which allows for more advanced traffic behavior.

To access the Waypoint Manager, click **Window** -> **Tools** -> **Candice Waypoint Manager**

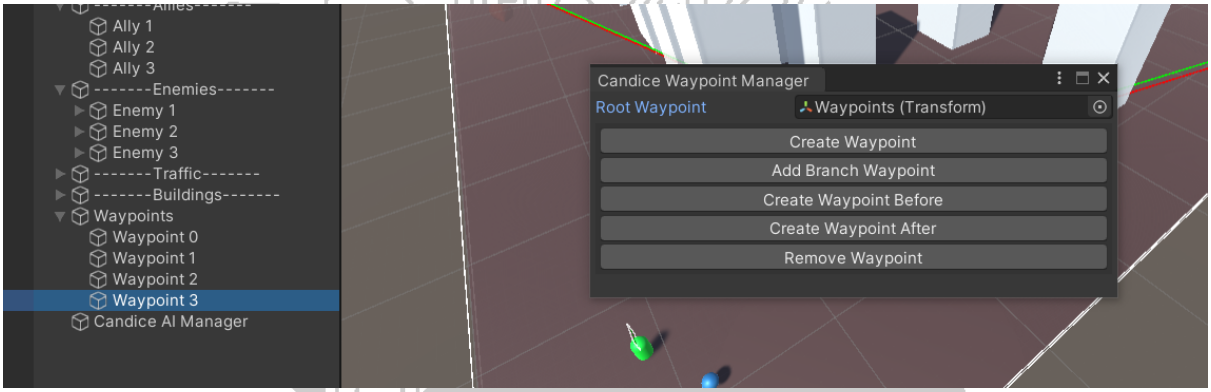


Once the manager appears, you must assign a root transform that will contain the waypoints. An empty game object is recommended. You can name this game object anything.





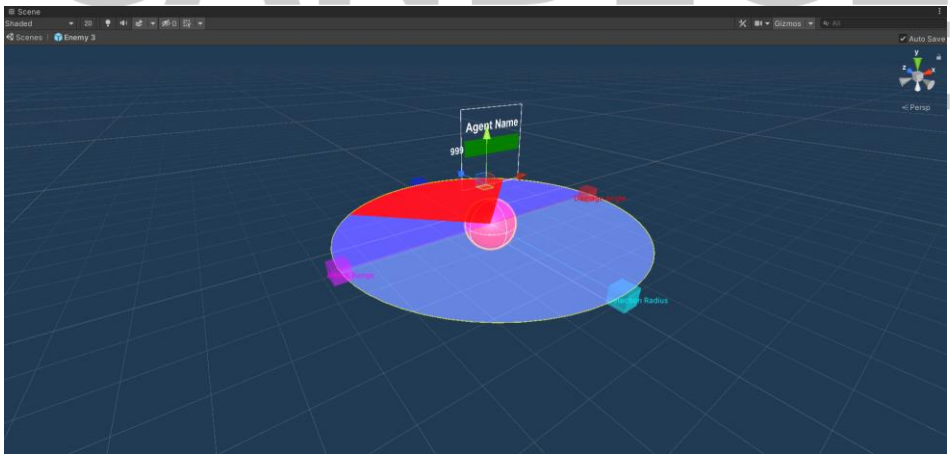
Next you Can click Create Waypoint to make your first Waypoint. After there, you can create waypoints before/after the current selected Waypoint.



From here, you can assign any waypoint to your agents and they will follow the Waypoints.

Editor and SceneView

Candice AI 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.

email: support@viridaxgamestudios.co.za

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

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

References

Item	Author	Website
Behavior Trees		https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php