2

Patrol

Patrolling is a simple extension to Path Finding. Instead of just having a single target in mind, we may have 2 or more points. We might go back and forth between them, or travel in a never ending loop.

In this chapter, you will learn about:

* How patrolling works
* How to patrol in QuickPath, React and RAIN AI packages
* Become introduced to Behavior Trees
* How the Character Controller can be applied to characters.
* How Unity’s NavMesh can be applied, even without these other AI’s

# Patrolling Overview

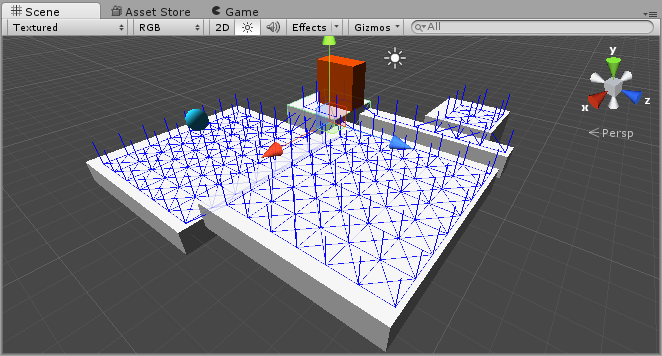
Patrolling, is a way to get a game object from point A to point B to point C and so on. Path finding is still required for getting from one waypoint to another, but here we daisy chain them in to a larger more meaningful path.

# QuickPath AI

Quick Path is back again, with built in capabilities for handling patrol. With its simple approach to the AI, it is only a few straight forward steps to get a scene finished. Here is a breakdown of those steps.

* Making the world ready for patrol
* Setting up the Patrol Script

We’ll start by expanding on our world from Chapter 1’s Quick Path demo. Stretch out a couple of the blocks to make a larger surface area. Then click on the terrain object (the parent of all the cubes forming the terrain) and in the Inspector, click the “Bake” button.



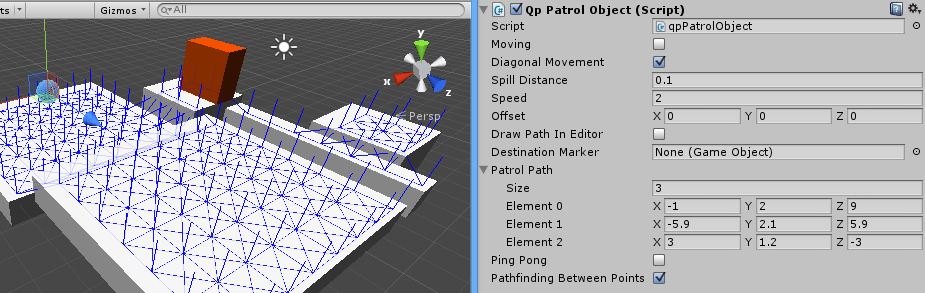
|  |
| --- |
| If the “Bake” function isn’t covering all the areas, you’ll need to check its grid dimensions. X remains the same as the world space, but Y is actually the Z axis. You might need to increase or decrease these numbers to cover everything.    Quick Path converts the Y/Z axis values internally. Be default it is set to Y as the Up/Down Axis, but you can change this with the Up Direction parameter. |

Now that we have a larger surface area to work with, we’ll get the NPC object setup to patrol. Presuming it still has the Following script, you will need to remove it, but then add the Quick Path Patrol script. First thing, is to move the speed down to 2, as its default of 10 was rather fast.

Next, set the “Spill Distance” to 0.1. Spill Distance is how close you have to be to a waypoint before it picks the next waypoint as your target. If “Ping Pong” is checked, the NPC will stop at the end of its path and back track. If it is unchecked, at the end of its path, it will target the first position and start over.

“Path Finding between Points” is an option to help it navigate around obstacles between way points. If your path is already clear, then you can keep this option off and save extra processing.

Finally, we have the “Patrol Path”, which houses the waypoints for the NPC to travel on. Increase that to 3, and then set the way points. A trick to figure out the values, is to move your NPC in the scene to the waypoint positions you want, and then copy its position to one of the way points. So select 3 points for your NPC to travel on.



Now your game is ready to run, with your NPC navigating a course you just setup.

# React AI

React AI doesn’t come equipped with a Patrol script, so I provided one to use with it. We’ll jump a little farther into the behaviour tree and also into the script code used to manage it. Here are the steps to reproduce it.

* Create a Patrol Script
* Create a Patrol AI
* Setup the NPC Patrol

To start with, I’ve provided a script for you to use. In it, I started with the last script for path finding, and then extended it to use a similar configuration to the “Patrol Path” in Quick Path’s Patrol script. Here are a couple key points about that script:

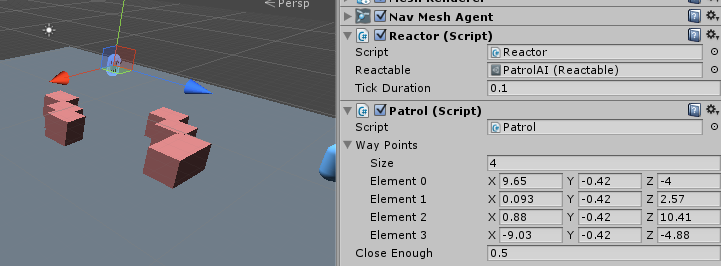
* It is based off of the FollowThePlayer script from the previous chapter.
* You can find the code in the book’s contents \Scripts\React AI\Patrol.cs
* It stores a public array of Vector3’s, so the Inspector UI can allow a designer to set the waypoints.
* Instead of the target being a player, it is set to the next waypoint in the list. Once we are “Close Enough”, it selects the next waypoint. Close Enough is the float field that allows the inspector to find it.
* If there are no waypoints left to select, it starts over at the first waypoint.

But now we need to create the user end point. Right click on a folder in the Project tab, and choose Create 🡪 Reactable. Name the Reactable “PatrolAI”. In the inspector for it, add the Patrol script a one of the behaviours. Next right click on the PatrolAI asset and select “Edit Reactable”.

In the reactible, right click on the root element and select Add 🡪 Branch 🡪 Sequence. A Sequence repeats all the steps in order. Under the Sequence, right click and select Add 🡪 Leaf 🡪 Action. Assuming you only added the Patrol script to its behaviours, it should automatically select “Patrol.Go” as its action. You can add notes to each step, to help write a better story of what the AI is doing. When it is this simple, it does not matter so much, but many AI’s will become more complex.

Next, the NPC needs to be setup to use this new patrol AI. Find the NPC in the previous chapter React AI project. You’ll need to remove the Following AI that was on the NPC prior, or create a new NPC. If you create a new NPC, do not forget to add the Nav Mesh Agent so it can navigate.

Add 2 components to the NPC’s: Reactor and Patrol. In the Reactor, you will need to set the Reactable to the PatrolAI asset we created earlier. And then in the Patrol script, add some way points. Like we did for the Quick Path patrol script, we need to set the Vector locations for each of the way points. A tip for getting exact numbers, is to just move the NPC to the way point positions you want, and then copy the position of the NPC to one of the way points.

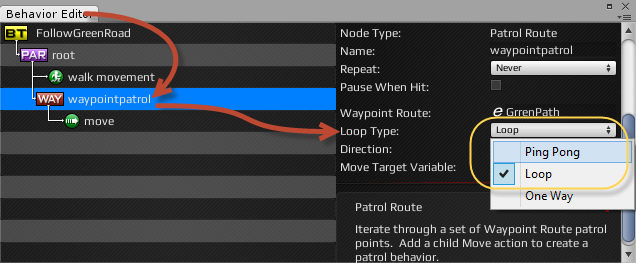


And now your game should have a character who patrols from point to point.

# RAIN AI

RAIN has this section put together pretty well. In reality, we only have one small section to change from the Path Finding demo. Particularly because the path finding demo had actually turned off the Patrol feature.

Start with the project for RAIN AI from Chapter 1. From the menus, select RAIN 🡪 Behaviour Tree Editor. From the editor, select FollowGreenRoad. Under the Sequence, is a Patrol Route node called “waypointpatrol”, select it. Finally, we have a property called “Loop Type”. Presently it is on “One Way”, which makes stops at the last waypoint. You can switch it to “Ping Pong” or “Loop”.



Ping Pong bounces you back and forth on the path, while Loop moves connects the last waypoint to the first to start over.

# Review

We were able to get patrolling operational in all three AI’s. Each AI had its own approach,

Experiences of working with all three:

For Patrol as well as Path Finding, Quick Path had very few steps. It worked exactly as expected, and again, there were no bugs on my first pass with setting up patrol. Bugs are pretty common place on a first pass, forgetting a step, something not working on this version of what ever, but QP had no issues. While it does not carry a decision or behaviour tree to manage AI, if your need is mostly path finding, it does exactly what it claims to do easily.

As for React, it was not able to do what I wanted out of the box, just like last chapter. However, it was not difficult to create a script that operates inside its Behaviour Tree AI Editor. This is a powerful system that you can use to allow designers easy access to apply your awesome scripts, but you have to be comfortable programming in order to build the pieces for it.

Finally we have RAIN, which made this about as easy as a big red button. With one setting changed, we changed the Path Finding AI from the previous chapter into a Patrolling AI. RAIN comes equipped with a huge variety of prebuilt character controlling capabilities that all work together very well. While you can program for it, you may find in many cases that you don’t have to. That, and this is freely available.