Swarm Move With Splines - Manual

Alexandru Dumbrava

December 8, 2024

Contents

Chapter 1

Introduction

Swarm Move With Splines allows you to make a swarm or large group of game objects move along a spline. Spline Animate is the Unity provided solution for this but it does not allow for interaction with the physics engine and it forces game objects to be positioned exactly along the spline.

What if you want something a little more flexible? Maybe you want 100 chickens or birds or fish or soldiers following a predetermined spline path and you want them all to collide with each other. This is the problem I encountered before developing this asset and it is the problem this asset solves!

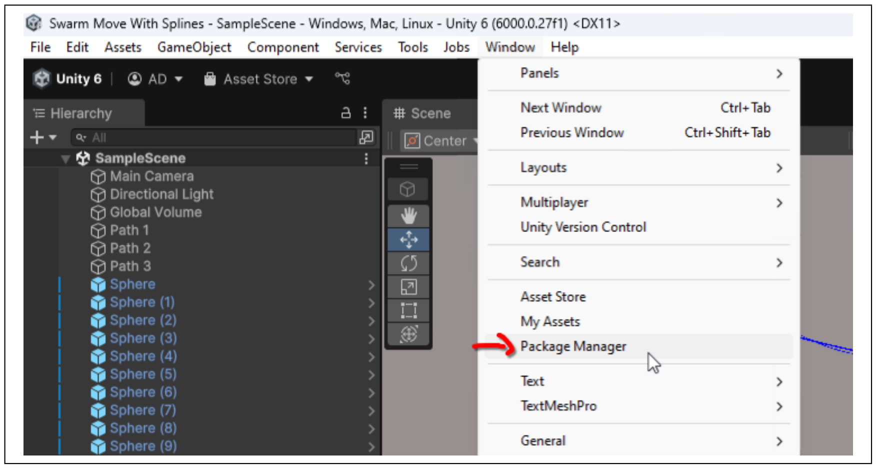
Swarm Move With Splines allows for single traversal, looping traversal and boomerang traversal all while allowing the physics engine to handle collisions and all the regular physics goodies through the usual rigidbody and collider components. It also provides a built-in PID controller that allows you to precisely tune the behaviour of your game objects.

If you've been frustrated at Spline Animate, I hope this asset package helps solve your frustrations!

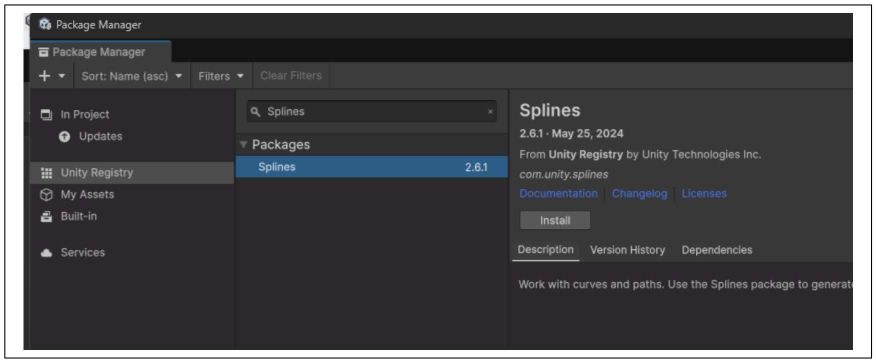
Chapter 2

Dependencies

There is only one dependency for this package. That is the Spline package from Unity. To add the Spline package, navigate to Window and then select Package Manager.



After the Package Manager is open, click on Unity registry and then search for Splines. Click the install button to add it to your project.



Work with curves and paths. Use the Splines package to generat

Chapter 3

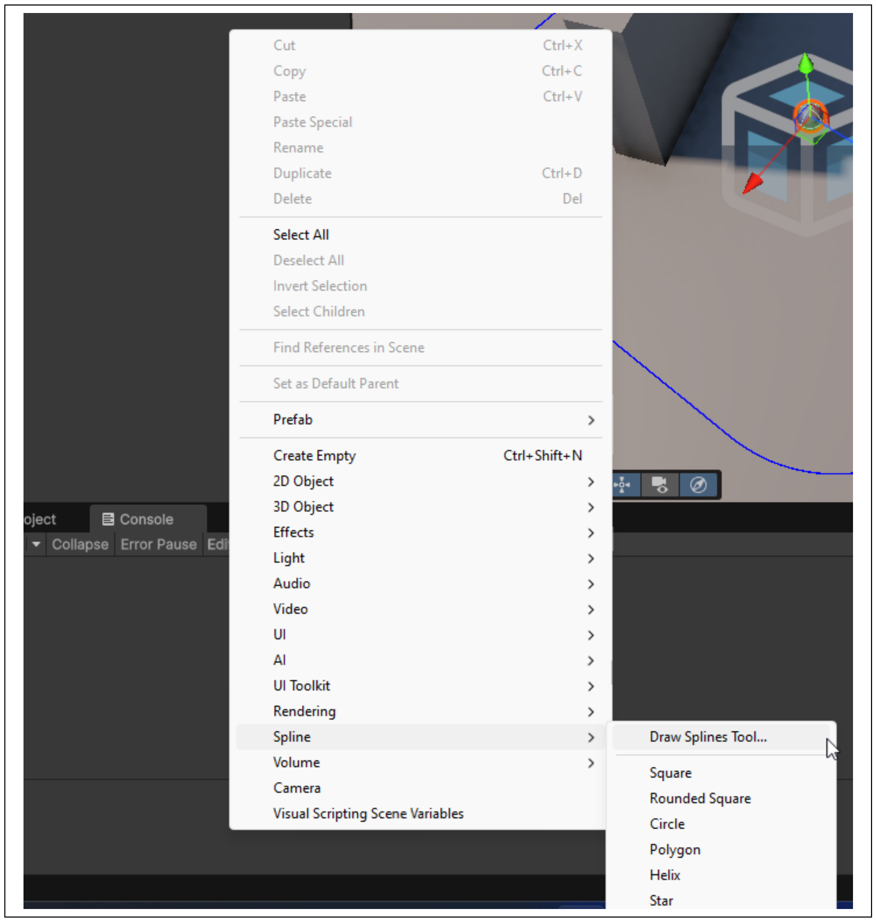
Setup

The main component in this package is the Swarm Move With Splines script. You can find this script file at the root level of the package. This script allows full customization over the spline following behaviour of any game objects it is attached to.

Add this script to any game object you want to follow a custom spline path. If you have a swarm of objects either add it to each object in the swarm or add it to a prefab and then instantiate the swarm at runtime using your own C# code.

NOTE: Make sure your game object has a rigidbody attached to it. The script adds force using the rigidbody to ensure it moves properly.

You're halfway done! The other part of the setup is creating your Spline container. Right click in the hierarchy and go to Spline. Then select the spline you want to create. There are a few standard splines or you can create your own custom spline using the Draw Splines Tool...



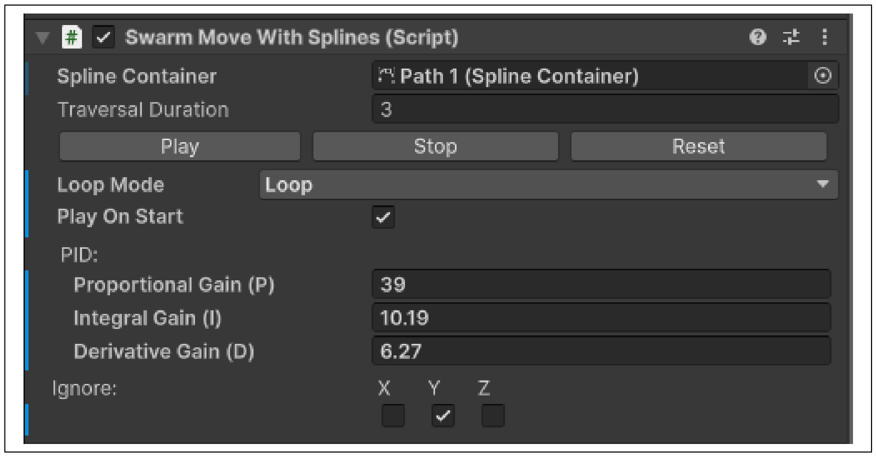
These tools are part of Unity's package so refer to their documentation if you require additional assistance with spline creation. Once you've created the spline container, you will need to add it to the Spline Container field in the script component. If the script is on an instantiated object (an object in the hierarchy), you can just drag and drop the Spline Container in. If you instead have a prefab that you instantiate through your own C# code, you will need to pass in the path programmatically yourself. There is a Spawn.cs script in the example subfolder that shows you how to handle prefabs.

And that's it for setup! The rest of the script contains fields for customization and tuning. We'll go over that soon.

Chapter 4

Script Fields

The script has the following inspector editor



Swarm Move With Splines (Script)

Spline Container: Path 1 (Spline Container)
Traversal Duration: 3

Loop Mode: Loop
Play On Start: ✓
PID:
Proportional Gain (P): 39
Integral Gain (I): 10.19
Derivative Gain (D): 6.27
Ignore: X ☐ Y ✓ Z ☐

Spline Container - this field holds a reference to your spline. Set it either through code when instantiating or set it on an object that is present in the hierarchy.

Traversal Duration - this field sets how long it takes to move along the spline in seconds. This is a float field so you can use fractions of a second. Just keep in mind that the physics simulation only runs a set amount of times per second. For boomerang loop mode, multiply this value by two to get the full duration for going from start to finish and back to start again.

Play / Stop / Reset - these buttons are here for testing purposes. Unfortunately, they do not work with multiple objects selected.

Loop Mode - There are three loop modes:

- None - Traverse the spline from start to finish once.
- Loop - Traverse the spline from start to finish and repeat once you finish the traversal. This is most useful for closed splines.
- Boomerang - Traverse the spline from start to finish and then go back to start doubling the traversal duration. Once back at the start, repeat. This is useful for swarms that need to go back and forth repetitively.

Play On Start - check this to start following the spline as soon as the game object is instantiated.

PID - PID controller that has your three P, I, and D variables. The values for these variables are covered in detail in the next chapter (Tuning).

Ignore - Gives you the option to ignore one specific global access when following the spline. For example, if your spline is drawn above ground and you don't want your game objects to fly, you can ignore Y and they will stay on the ground.

Chapter 5

Tuning

The main power of this script comes from the built in PID controller. If you've never encountered this term it's a technology that allows systems to reach a target signal. The usual example given in literature is a thermostat. If the temperature gets too low, the system needs to kick in and heat up the house. If it gets too high, it needs to shut off and allow the house too cool. Another example is the cruise control on your car.

PID controllers have three variables you can play with. P, I, and D. The first and simplest is P for Proportional Gain. This is a direct measurement of how far off the system is from the desired signal. In our case it's the difference between the game object's current position and the position it should be at along the spline. When finding your ideal values start with P first as that tends to have the greatest impact.

As you get your P close to where it needs to be for your use case, you will usually notice there is an oscillation. This means that the game object swings from one side of the spline to the other. To reduce the oscillation increase your D value. D stands for Derivative Gain and its influence increases along with increases in the error's rate of change. All you really need to know is that this reduces the oscillation caused by P.

In the middle we have I. This stands for Integral Gain. The longer an error exists in the system, the higher influence I will have. This comes in handy when you have other forces (such as gravity) acting on your game objects. If you have a rocketship trying to reach a target and it's constantly being pulled down by gravity, a PID controller without I will never reach its target. It will find an equilibrium just below the target. To fix this make sure your I is greater than 0.

Unfortunately, PID tuning is more of an art than a science. Each use case is different and the values that satisfy you will be different. In building the sample scene I started out with P then D and finally some I. P had the greatest influence on getting the swarm to follow the spline. I would recommend keeping your values above zero, however, there are times when a negative value is useful. Just keep playing around with it until you're happy. And remember to copy the values before exiting Play Test mode!