HOUR 16, LECTURE 2

Tick and Timelines:
Modifying Values over Time

INTRODUCTION

You will often need to modify values at runtime in your games. In many cases, you can simply modify the value and you're done. An example might be a light that is flicked on and off.

Other times, you'll want the value to change gradually over time. For example, you would probably want a door to open smoothly, not to suddenly go from closed to fully open. This is often referred to as interpolation.

In this lecture, you will learn how to use Timelines and Tick events to animate values over time.

LECTURE GOALS AND OUTCOMES

Goals

The goals of this lecture are to

- Learn about Tick and Delta Time
- Learn how frame rate influences animations
- Learn how Timelines work

Outcomes

By the end of this lecture you will be able to

- Use the Tick event in your Actor Blueprints
- Use Timelines in your Actor Blueprint classes
- Use Delta Time to make smooth value interpolations

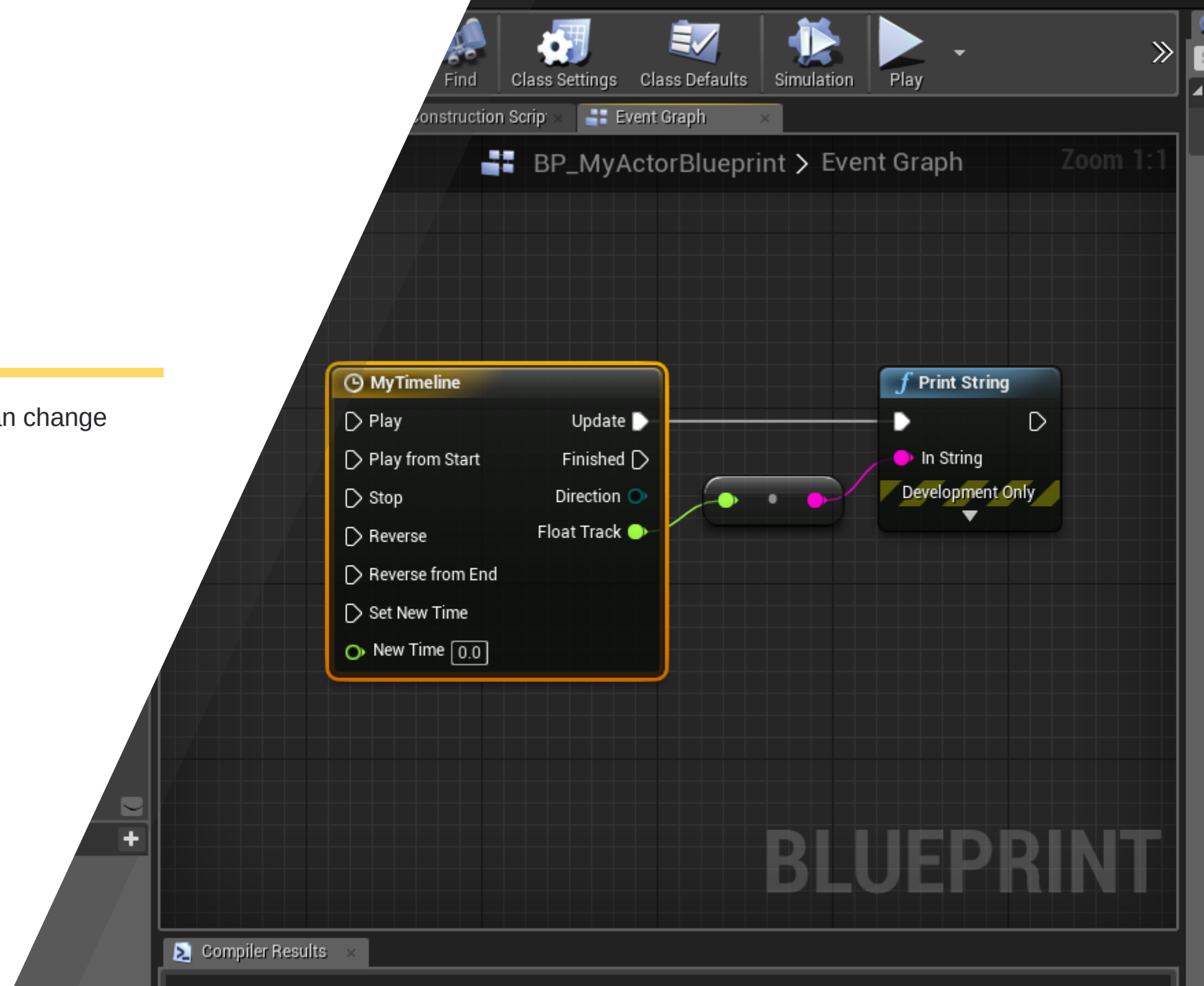
TIMELINES

Blueprint Nodes for Animating Values over Time

TIMELINES

Timelines allow you to create values that can change over a specified period of time.

You can control Timelines using common Play/Pause/Rewind controls.



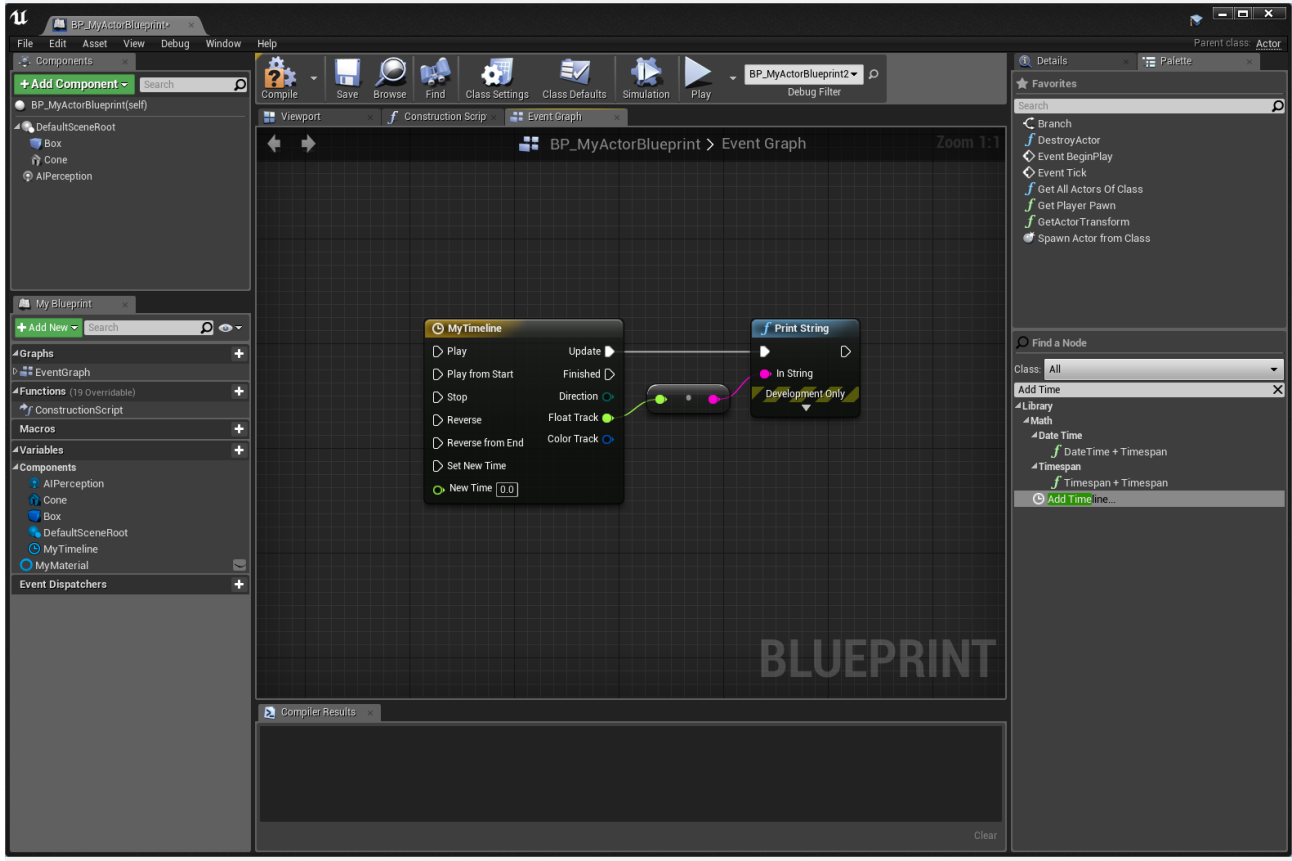
CREATING TIMELINES

Timelines are created like other Blueprint nodes.

You can use the Palette panel or the right-click Blueprint Context Menu to select Add Timeline.

You can name your Timeline.

Timelines can be accessed as variables and are available as such under the My Blueprint tab.



TIMELINE NODE

By default, you will see the following on a new Timeline node:

- **Input exec pins**: These pins appear along the left and are used to control the playback of the Timeline.
- **Update exec pin**: This pin fires on each frame when the Timeline is being played.
- **Finished exec pin**: This pin fires once the Timeline reaches the end or the Stop function is called.
- **Direction exec pin**: This pin will return an enum value of Forward or Backward.

TIMELINE EDITOR

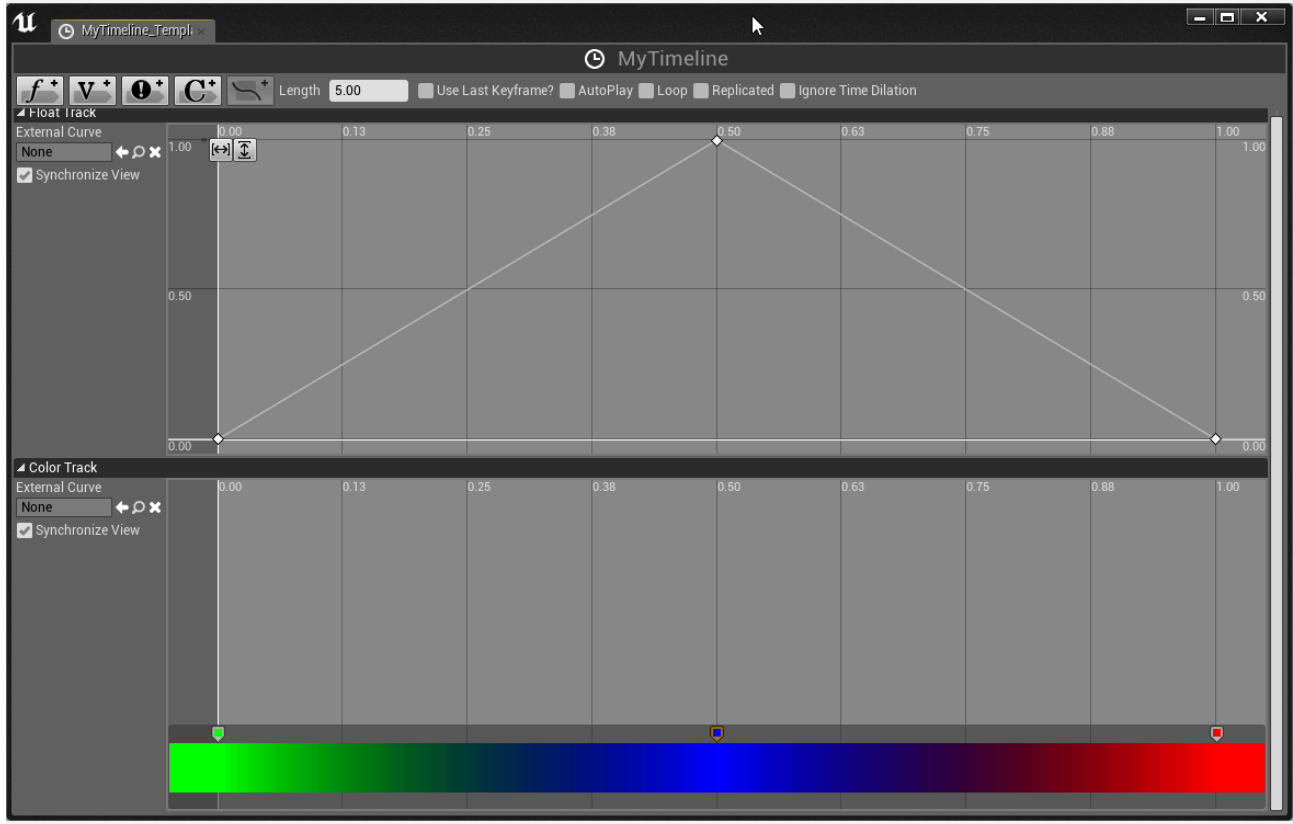
Timelines are edited directly within the Blueprint Editor interface by double-clicking on the Timeline node in the Graph Editor.

You can add multiple tracks, each of a specific type to be interpolated. Each track is represented by a single Curve Editor panel.

In the image on the right, you can see a Float Track and a Linear Color Track being edited.

You can have the following track types:

- Float
- Vector
- Event
- Linear Color
- External Curve (a curve created and edited within the Content Browser)



TIMELINE EDITOR

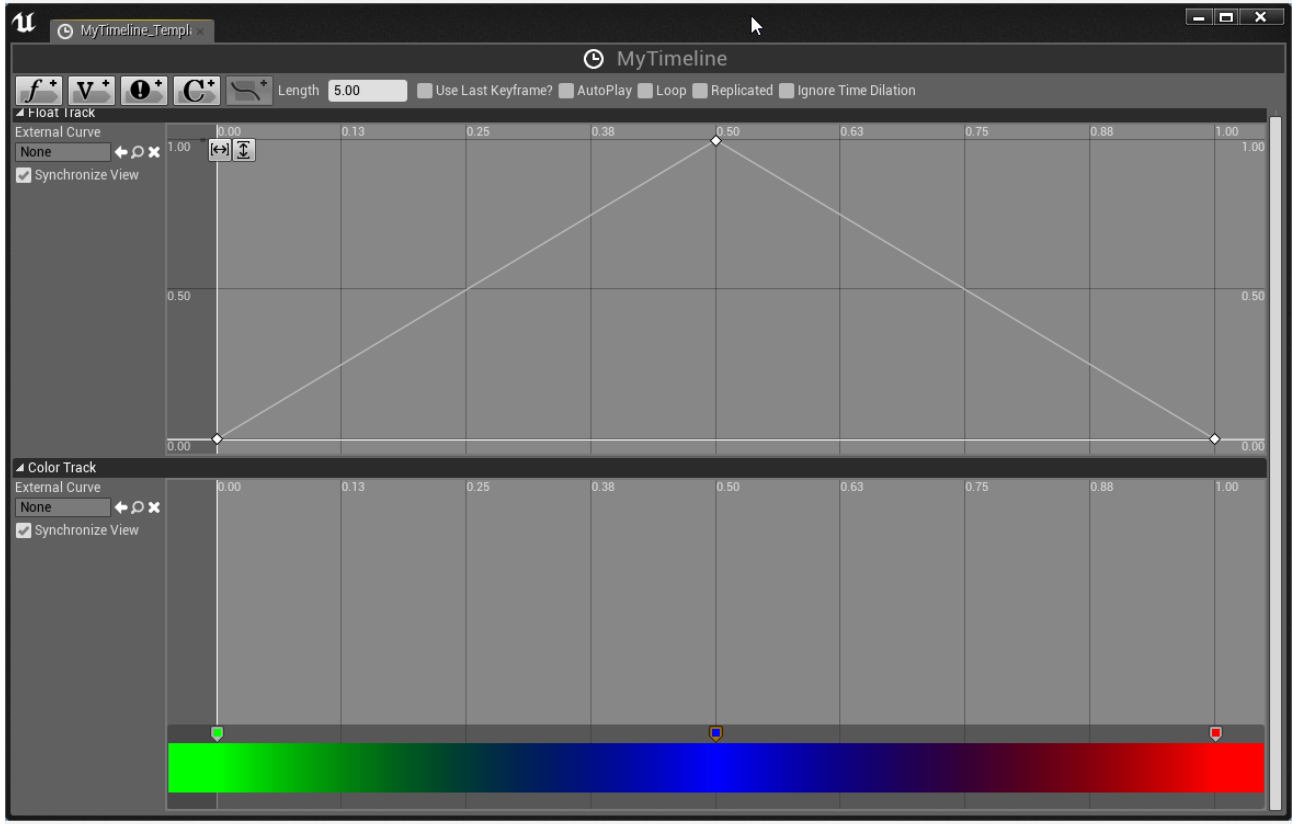
You can either explicitly set the length of the Timeline or have it auto-set with Use Last Keyframe.

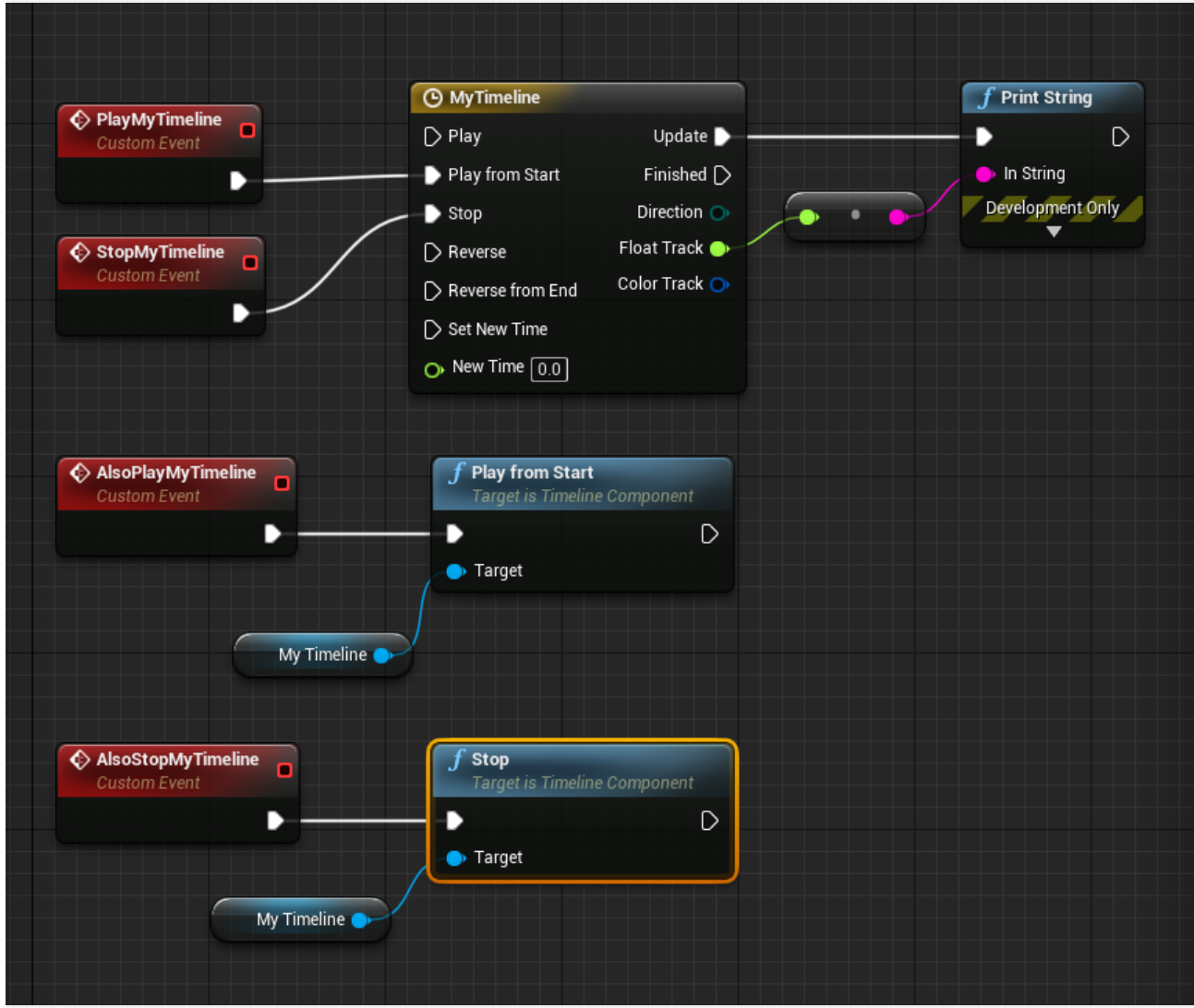
AutoPlay will make the Timeline play as soon as the Actor that contains it is spawned.

Loop will make the Timeline loop back to time 0 when it reaches either the end of the Timeline or the last keyframe.

You can choose to have the values replicated for multiplayer.

Time dilation is a method used in UE4 to slow down or speed up the passage of time in the game. You may want to ignore time dilation and ensure that 1 second in your Timeline always equals 1 second of real-world time.





CONTROLLING PLAYBACK

You can use the input execution pins on the Timeline node.

You can also control the Timeline using a reference to the Timeline, calling functions like most Blueprint variables.

***Note**: The Update output execution pin will not fire unless the Timeline is playing.*

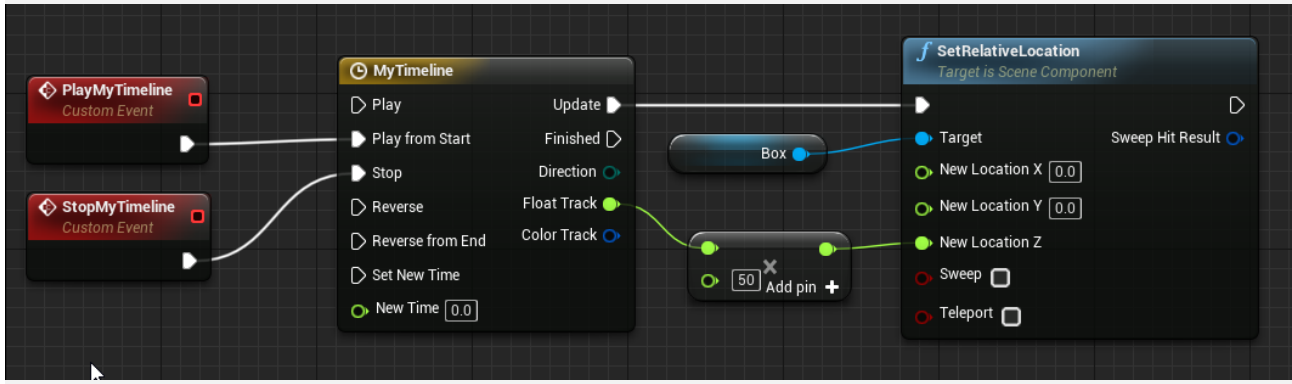
DRIVING PARAMETERS

As you add Tracks in the Timeline Editor, the Timeline node in the Event Graph will be modified to include output pins for each of the tracks you add.

- In the image on the right, you can see that the Float Track and the Linear Color Track that were added via the Timeline Editor appear as output pins on the Timeline node.

Using the Update exec pin along with the track output pins lets you drive parameters.

- Here, the position of the Box Static Mesh component is being modified based on the value of the Float Track.



TICK EVENT

Animating Values over Time Using the Tick Event

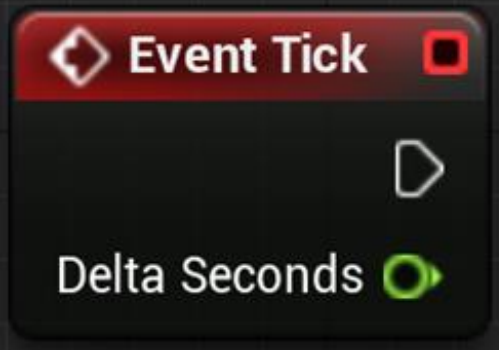
TICK EVENT

All Actor classes can use the Tick event.

Tick is called on each frame.

Tick is where you add code that happens over time or needs to happen on each frame.

The Tick event returns Delta Seconds, a float value.

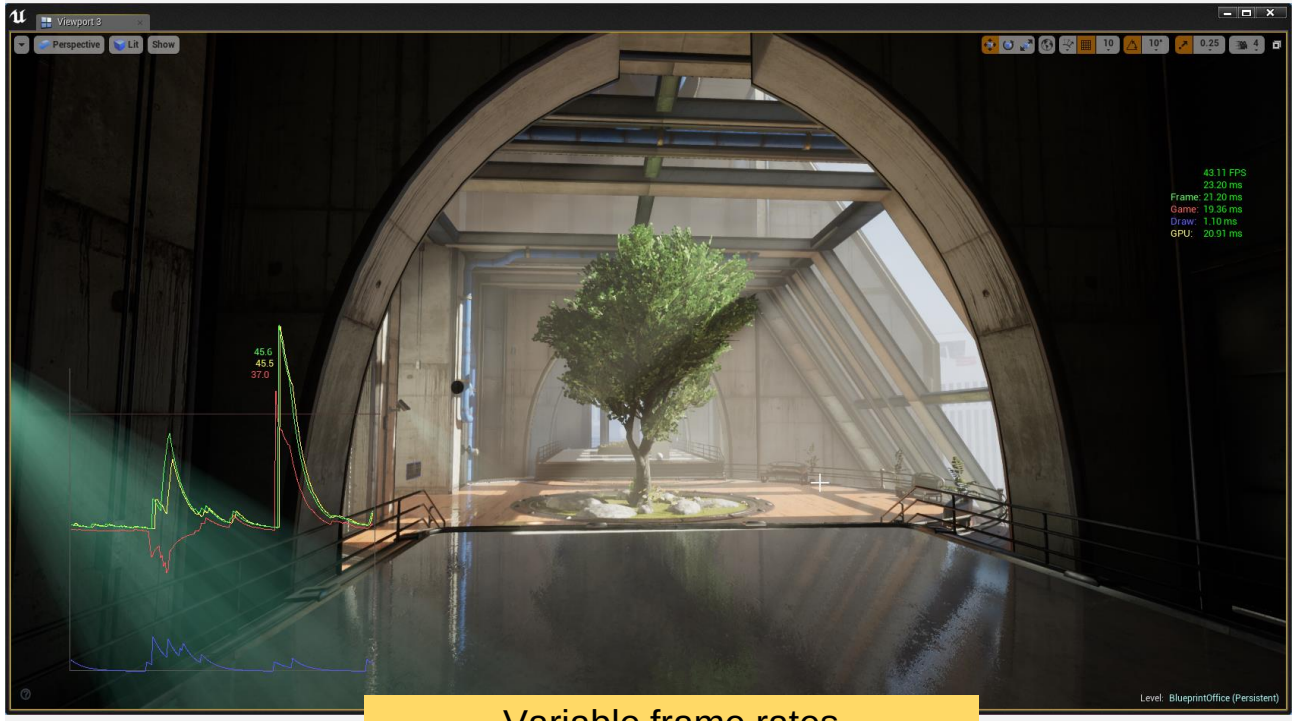


FRAME RATE

Delta Seconds returns how much time (in seconds) passed since the last time Tick was called.

Delta Seconds changes depending on your game's frame rate.

Your game might run at very different frame rates on different computers, and it might run at a different frame rate moment to moment depending on what's on-screen. You need to compensate for this in your movement code using Delta Seconds.



Variable frame rates

DELTA SECONDS

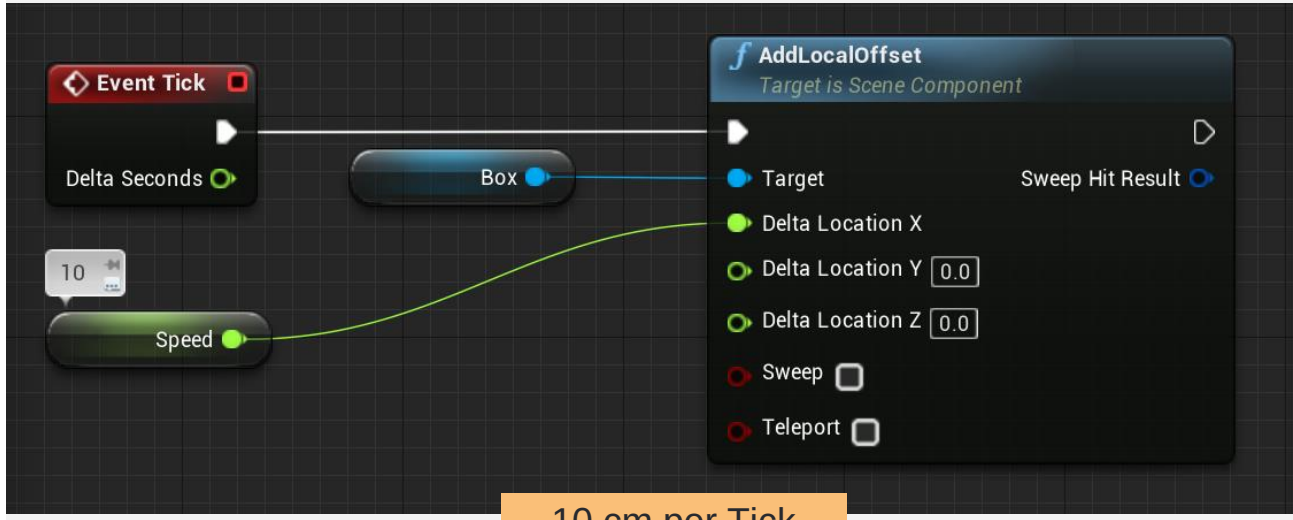
Without Delta Seconds, a box moving at 10 cm per Tick would move different distances in a single second:

- 60 fps: 10 cm * 60 = 600 cm per second
- 30 fps: 300 cm * 30 = 300 cm per second
- 12 fps: 120 cm * 12 = 120 cm per second

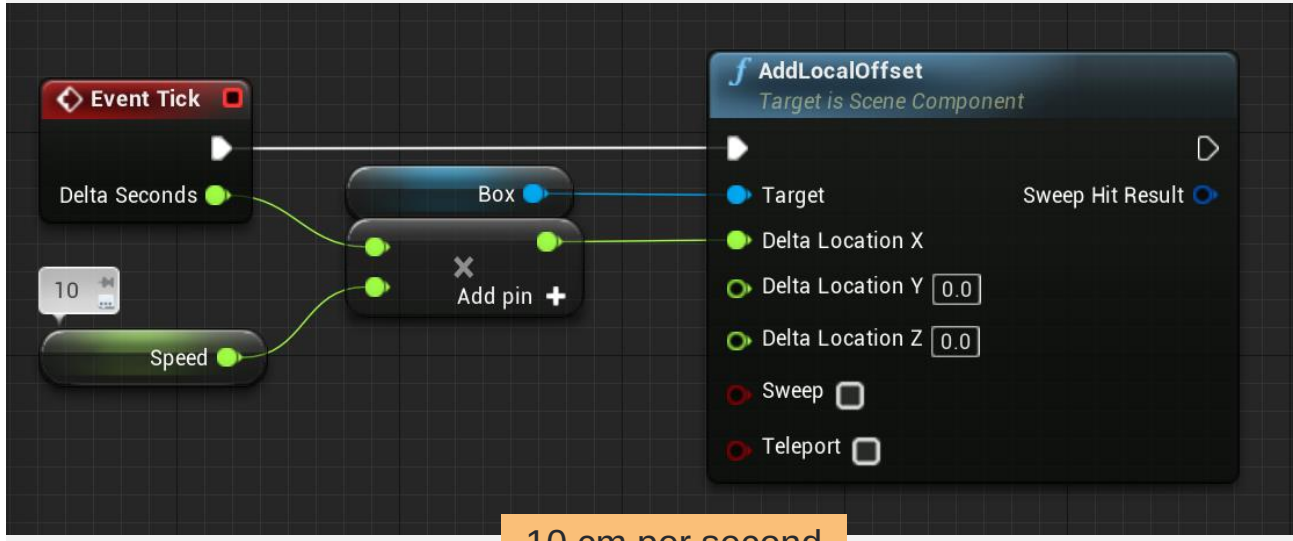
By multiplying Speed by Delta Seconds, the box will move at a consistent speed, independent of the frame rate:

- 60 fps: (10 cm * .016) * 60 = 10 cm per second
- 30 fps: (10 cm * .033) * 30 = 10 cm per second
- 12 fps: (10 cm * .083) * 12 = 10 cm per second

This makes the box's velocity consistent, moving at 10 cm per second no matter the frame rate.



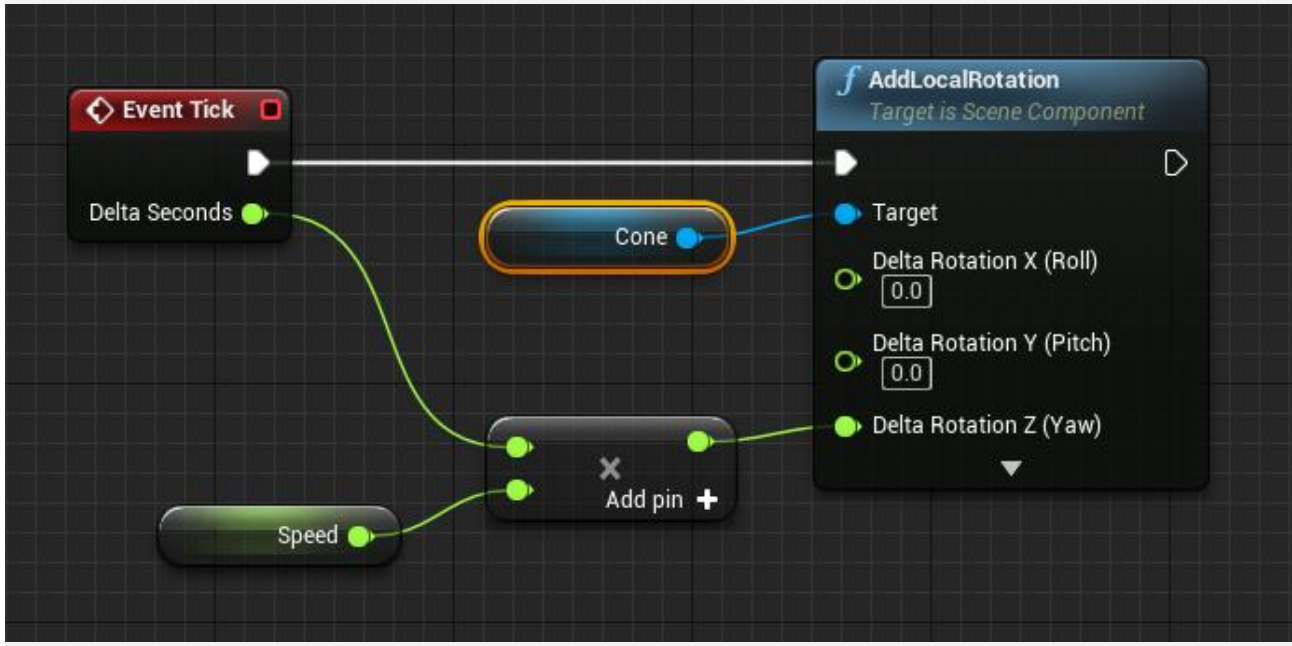
10 cm per Tick



10 cm per second

TICK ROTATION

Adding Local Rotation on each Tick event is a great way to get a constantly rotating effect.

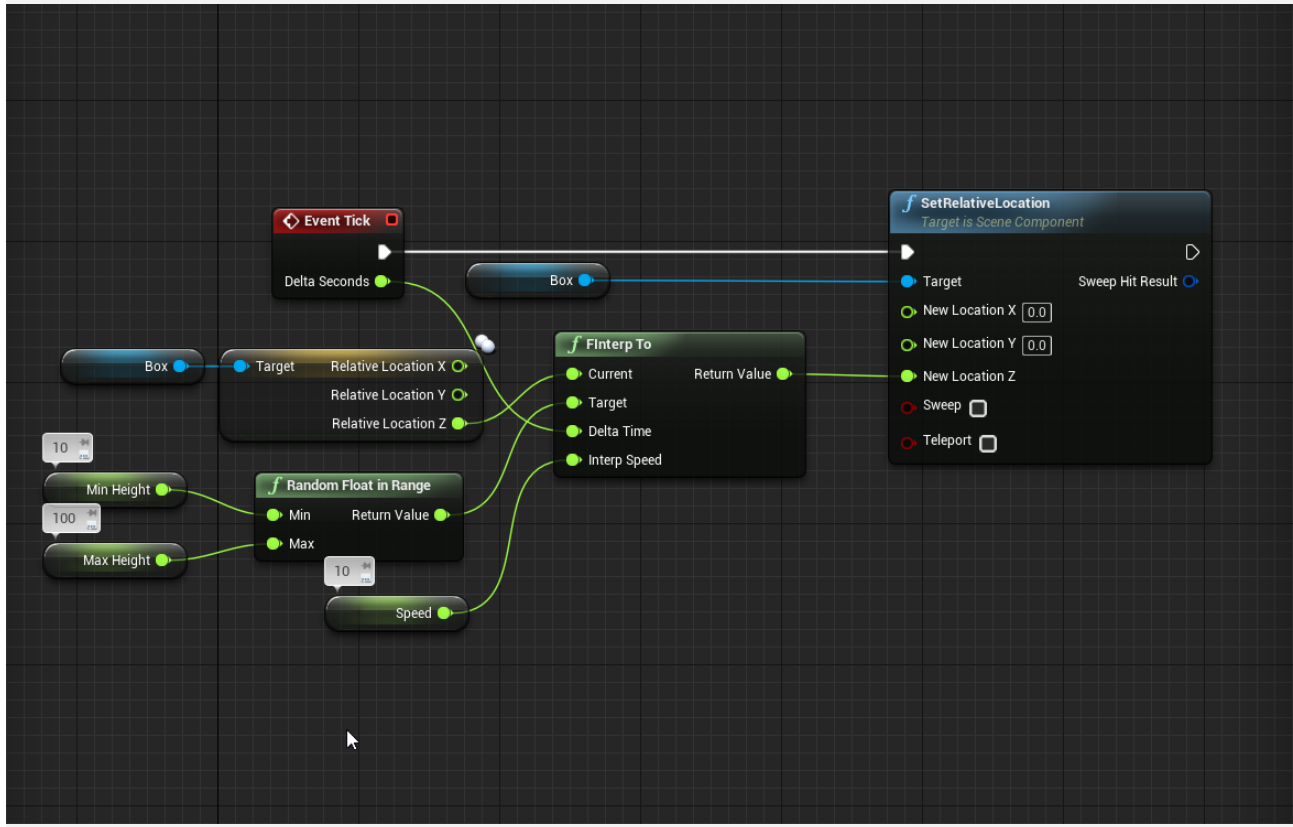


TICK INTERPOLATE

There are several interpolation nodes available.

You can add smooth motion, including easing, using interp functions.

In this example, the box will be randomly and smoothly moved up and down on the *z* axis between the Min Height and Max Height values.



CAUTION

Tick runs on every frame. This means that it can have a significant impact on performance. Be careful of what you add to the Tick event.