Playables API

Zochu Liu

jliu5@berklee.edu

The Playables API provides a way to create tools, effects or other gameplay mechanisms by organizing and evaluating data sources in a tree-like structure known as the PlayableGraph. The PlayableGraph allows you to mix, blend, and modify multiple data sources, and play them through a single output.

The Playables API supports animation, audio and scripts. The Playables API also provides the capacity to interact with the animation system and audio system through scripting.

Although the Playables API is currently limited to animation, audio, and scripts, it is a generic API that will eventually be used by video and other systems.

Playable vs Animation

The animation system already has a graph editing tool, it's a state machine system that is restricted to playing animation. The Playables API is designed to be more flexible and to support other systems. The Playables API also allows for the creation of graphs not possible with the state machine. These graphs represent a flow of data, indicating what each node produces and consumes. In addition, a single graph is not limited to a single system. A single graph may contain nodes for animation, audio, and scripts. More info about State Machine

Advantages of using the Playables API

- The Playables API allows for dynamic animation blending. This means that objects in the scenes could provide their own animations. For example, animations for weapons, chests, and traps could be

dynamically added to the PlayableGraph and used for a certain duration.

- The Playables API allows you to easily play a single animation without the overhead involved in creating and managing an AnimatorController asset.

- The Playables API allows users to dynamically create blending graphs and control the blending weights directly frame by frame.

- A PlayableGraph can be created at runtime, adding playable node as needed, based on conditions. Instead of having a huge “one-size-fit-all” graph where nodes are enabled and disabled, the PlayableGraph can be tailored to fit the requirements of the current situation.

The PlayableGraph

In Unity, a Playable Graph is a powerful tool that allows you to create complex timelines for your game or application. It's essentially a way to define and orchestrate different pieces of content, such as animations, audio, and events, into a sequence that can be controlled and manipulated at runtime.

At its core, a Playable Graph consists of a collection of playable objects that represent different types of content. For example, you might have a playable object that represents an animation clip, another that represents a sound effect, and another that represents a game event. These playable objects can then be connected together in a sequence, with each one triggering the next one in the sequence when it's finished.

What makes the Playable Graph so powerful is that it's highly customizable and flexible. You can create your own custom playable objects to represent any type of content that you want, and you can use the graph to define complex branching sequences, add conditions and triggers, and more. This makes it possible to create sophisticated, interactive experiences that are responsive to player input and other factors.

From Unity Documentation:

The PlayableGraph defines a set of playable outputs that are bound to a GameObject or component. The PlayableGraph also defines a set of playables and their relationships.

The PlayableGraph is responsible for the life cycle of its playables and their outputs. Use the PlayableGraph to create, connect, and destroy playables.

A playable is a C# struct that implements the IPlayable interface. It is used to define its relationship with other playables. Likewise, a playable output is a C# struct that implements IPlayableOutput and is used to define the output of a PlayableGraph.

Scripting With PlayableGraph

Here are the basic steps for scripting a playable graph in Unity:

1. Create a new playable graph using the **PlayableGraph.Create()** method. This will create an empty graph that you can use to define your timeline.

2. Create a series of playables that represent the content that you want to include in your timeline. This might include animations, audio clips, particle effects, and other types of content. Example: **AnimationClipPlayable.Create(playableGraphName, animationPlayableName);**

3. Create an array of **PlayableOutput** objects that represent the outputs of your playables. These outputs define how the content should be rendered or played back, such as to a specific render target or audio channel.

4. Use the **PlayableGraph.Connect()** method to connect your playables together in a sequence, defining the order in which they should be played back.

5. Use the **PlayableGraph.BindOutput()** method to bind each of your **PlayableOutput** objects to the appropriate render or playback targets.

6. Use the **PlayableGraph.Play()** method to start playback of the graph.

To script a playable in Unity, you can use the **PlayableBehaviour** class. This class provides a way to define custom behavior for a playable, such as how it should initialize, update, and exit.

Here are the basic steps for scripting a playable:

1. Create a new class that inherits from **PlayableBehaviour**.

2. Override the **PrepareFrame()** method to set up any necessary data for the playable. This method is called once per frame before the playable is updated.
3. Override the **ProcessFrame()** method to perform any necessary updates to the playable. This method is called once per frame after the **PrepareFrame()** method.

4. Optionally, override the **OnBehaviourPlay()** and **OnBehaviourPause()** methods to perform any custom behavior when the playable starts or stops playing.

5. Use the **ScriptPlayable<T>** class to create a new playable instance and associate it with your **PlayableBehaviour** class.

6. Add the new playable instance to a playable graph, as described in the previous section.

Download PlayableGraph Visualizer

https://github.com/Unity-Technologies/graph-visualizer

Learning Source:

https://docs.unity3d.com/Manual/Playables-Graph.html

https://dev.rbcafe.com/unity/unity-5.3.3/en/Manual/Playables.html

https://www.reddit.com/r/Unity3D/comments/cmho0e/where_has_playablegraph_been_all_my_life/

https://blog.unity.com/technology/unity-2017-1-feature-spotlight-playable-api