Tutorial for Animation Manipulation

- Configuring and Loading an Animation
- Controlling Animations
- Using Animation Callback Functions
- Manipulating Animation Keyframe Sequences

Configuring and Loading an Animation

Candera Animation Concept

The following key concepts are required to run animations in a Candera application:

- Candera::Animation::AnimationPlayer provides the control interfaces (use Candera::Animation::AnimationGroupPlayer for animation groups)
- Candera::Animation::AnimationTimeDispatcher dispatches the current world time to animations; animations can also query the current world time from the dispatcher
- Candera::Animation::AnimationKeyframeSequence configures proper animation values for each time point of an animation.
- Candera::Animation::InterpolationStrategy specifies for a key frame sequence, how values between several key frames shall be calculated.
- Candera::Animation::AnimationPropertySetter applies animation values to animated properties

In SceneComposer, animations can be configured in the AnimationDesign view:

- Create a new animation - New Animation in SceneComposer User Manual
- Select a node property to animate - Add Animated Property in SceneComposer User Manual
- Specify the key frame sequence including property values and interpolation strategy - Animation Values in SceneComposer User Manual

It is also possible to import an animation to SceneComposer from a content generation tool via FBX format - How to Import Resources in SceneComposer User Manual

Configuring Animation Player

In the Player or in an application, the animation can be retrieved from the generated asset by specifying its Candera name (e.g. *MyModule#Animations#Animation*).

```
AssetProvider* assetProvider = GetAssetProvider();
m_animPlayer = assetProvider->GetAnimation(m_animName);
```

> In SceneComposer the animation can be retrieved from AssetProvider only if it was previously loaded in the scene. A better way to access it directly is to define it as a widget property.

```
// Define Animation property
CdaProperty(Animation,Candera::Animation::AnimationPlayer::SharedPointer, GetAni
    CdaDescription("Animation")
CdaPropertyEnd()
// End Define Animation property
```

Next, as already explained in Initializing AssetLoader and Animations, an instance of Candera::Animation::AnimationTimeDispatcher is needed to provide current world time to animations.

For this, the new animation player must be added to the Candera::Animation::AnimationTimeDispatcher instance maintained by the application.

```
// Add player to time dispatcher

Animation::AnimationTimeDispatcher::SharedPointer timeDispatcher = GetAnimationTimeDispa
if (timeDispatcher == 0) {
    return;
}
static_cast<void>(timeDispatcher->AddPlayer(m_animPlayer));

// End Add player to time dispatcher
```

Now the animation player is ready to be used.

Controlling Animations

Controlling Animation Flow

This section describes the control flow interfaces of Candera::Animation::AnimationPlayer.

- Candera::Animation::AnimationPlayer::Start starts the animation
- Candera::Animation::AnimationPlayer::Stop stops the animation
- Candera::Animation::AnimationPlayer::Finish finishes the animation within the specified time
- Candera::Animation::AnimationPlayer::Pause pauses the animation
- Candera::Animation::AnimationPlayer::Resume resumes the animation
- Candera::Animation::AnimationPlayer::SetRepeatMode sets the repeat mode, for possible modes see Candera::Animation::AnimationPlayer::RepeatMode
- Candera::Animation::AnimationPlayer::SetDirection sets the direction, for possible directions see Candera::Animation::AnimationPlayer::PlayDirection

Animation Speed, Repeats, Time Modulation

This section specifies more interfaces to set animation player properties relevant for timing and execution flow.

- Candera::Animation::AnimationPlayer::SetSpeedFactor increases / decreases animation speed
- Candera::Animation::AnimationPlayer::SetRepeatCount modifies repeat count
- Candera::Animation::AnimationPlayer::SetSequenceDurationMs modifies animation duration in milliseconds
- Candera::Animation::AnimationPlayer::SetSequenceStartTimeMs adds a start delay in milliseconds

Using Animation Callback Functions

Animation Event Listener

If an application shall be notified about events of the triggered animation, a listener needs to be implemented. The class Candera::Animation::AnimationPlayerListener allows to listen to a versatile set of animation events like:

- OnStartAntimation,
- OnResumeAnimation,
- OnPauseAnimation,
- OnDirectionChange,
- OnStopAnimation,
- OnFinishAnimation,
- OnPastEnd.

Implement AnimationPlayerListener Functions

In order to register an AnimationPlayerListener to an AnimationPlayer simply derive from Candera::Animation::AnimationPlayerListener and override pure virtual functions with custom code.

In the example below the 3D application derives from AnimationPlayerListener and overrides the listener's functions.

```
class AnimationPlayerWidget : public CgiWidget3D, public Candera::Animation::AnimationPlayerList
```

```
        virtual void OnPastEnd(Candera::Animation::AnimationPlayerBase
* animationPlayer, Candera::Int32 completedIterationsCount);
        virtual void OnStartAnimation(Candera::Animation::AnimationPlayerBase
* animationPlayer);
        virtual void OnStopAnimation(Candera::Animation::AnimationPlayerBase* animationPlayer);
        virtual void OnFinishAnimation(Candera::Animation::AnimationPlayerBase
* animationPlayer);
        virtual void OnResumeAnimation(Candera::Animation::AnimationPlayerBase
* animationPlayer);
        virtual void OnPauseAnimation(Candera::Animation::AnimationPlayerBase
* animationPlayer);
```

Use Candera::Animation::AnimationPlayerBase::AddAnimationPlayerListener and Candera::Animation::AnimationPlayerBase::RemoveAnimationPlayerListener for registering or removing a listener.

```
        static_cast<void>(m_animPlayer->AddAnimationPlayerListener(this));
```

```
        static_cast<void>(m_animPlayer->RemoveAnimationPlayerListener(this));
```

Now the application is notified about all animation listener events it has implemented. For example when implementing the Candera::Animation::AnimationPlayerListener::OnStartAnimation, the application is notified about all started animations.

```
void AnimationPlayerWidget::OnStartAnimation(Animation::AnimationPlayerBase* /* animationPlayer
{
    // do something when an animation has started
}
```

Manipulating Animation Keyframe Sequences

This method allows to modify specific keyframes of a previously initialized keyframe sequence.

Define Keyframe Values

Size of array must be equal with the number of values that are already assigned to each keyframe (numberOfComponents).

```
    Float keyFrameManipulationValues[3] =
    {
        1.0, 1.0, 11.0
    };
```

Set Keyframe Values

Values will be attached to one keyframe.

```
    m_keyframeSequence->SetKeyframeValues(1, keyFrameManipulationValues[0]);
```

See also:

Candera::Animation::AnimationKeyframeSequence::SetKeyframeValues