动画技术简介

Qt设计工作室支持以下类型的动画技术，适用于不同的目的：

- 2D 和 3D 的常见运动设计技术
- 屏幕到屏幕或状态到状态动画
- 数据驱动的 UI 逻辑动画

常见运动设计技术

下表总结了 2D 和 3D 的常见运动设计技术及其典型用例。

技术	用例
时间轴动画	通过指定关键帧处的中间值进行线性插值，而不是立即更改为目标值。
缓动附加到关键帧的曲线	关键帧之间的非线性插值，使组件在动画结束时显示为加快速度、减慢速度或反弹。
附加到关键帧的动画曲线	需要多个关键帧的复杂 3D 动画，因此有必要同时可视化关键帧的值和插值。

时间轴动画基于*关键帧*。在Qt设计工作室中，关键帧确定组件在特定时间的属性值。通过动画处理属性，其值可以在中间值之间移动，而不是立即更改为目标值。

例如，可以将矩形的 y 位置属性在动画开始时设置为 0，在动画结束时设置为 100。运行动画时，矩形从 y 轴上的位置 0 移动到 100。在动画的中间，y 属性的值为 50，因为默认情况下关键帧是线性插值的。

缓动曲线

有时您不希望线性移动，而是希望矩形在动画开始时移动得更快，在动画结束时移动得更慢。若要实现此效果，可以在起始帧和结束帧之间插入大量关键帧。为避免这种工作，您可以为关键帧之间的非线性插值指定*缓动曲线*。缓动曲线可以使组件在动画结束时显示为加快速度、减速或反弹。

动画曲线

虽然缓动曲线适用于大多数简单的 UI 动画，但更复杂的 3D 动画需要多个关键帧，因此有必要同时可视化关键帧的值和插值。"曲线"视图一次可视化属性的整个动画，并显示关键帧的有效值以及关键帧之间的插值。它还可以同时显示不同属性的动画，以便您可以并排查看x位置的动画和y位置的动画。

屏幕到屏幕或状态到状态动画

下表总结了用于在屏幕和 UI 状态之间导航的技术。

技术	用例
应用程序流	一个交互式原型，可以单击以模拟应用程序的用户体验。
状态之间的转换	使用基于关键帧的过渡时间线在 UI 的不同状态之间进行过渡。可将缓动曲线应用于关键帧。

应用程序流

您可以以*原理图*的形式设计应用程序，该逻辑示意图通过符号显示应用程序 UI 的所有重要组件及其互连。这将生成一个交互式原型，可以单击该原型来模拟应用程序的用户体验。代码是在后台创建的，可用作应用程序生产版本的基础。

有关更多信息，请参见设计应用程序流。

Transitions Between States

UIs are designed to present different UI configurations in different scenarios, or to modify their appearances in response to user interaction. Often, several changes are made concurrently so that the UI can be seen to be internally changing from one *state* to another.

This applies generally to UIs regardless of their complexity. A photo viewer may initially present images in a grid, and when an image is clicked, change to a detailed state where the individual image is expanded and the interface is changed to present new options for image editing. At the other end of the scale, when a button is pressed, it may change to a *pressed* state in which its color and position are modified so that it appears to be pressed down.

Any component can change between different states to apply sets of changes that modify the properties of relevant components. Each state can present a different configuration that can, for example:

- Show some UI components and hide others.
- Present different available actions to the user.

- Change a property value for a particular component.
- Show a different view.

State changes introduce abrupt motion that you can make visually appealing by using *transitions*. Transitions are animation types that interpolate property changes caused by state changes.

In Transitions, you can set the start frame, end frame, and duration for the transition of each property. You can also set an easing curve for each animation and the maximum duration of the whole transition.

Data-Driven UI Logic Animations

The following table summarizes techniques used for animating the UI logic by using real or mock data from a backend.

Technique	Use Case
Data-driven timeline animation	Using real or mock data from a backend to control motion.
Programmatic property animation	Interpolating property values programmatically to create smooth transitions.

Data-Driven Timeline Animation

You can connect property values to data backends to drive timeline animation. You can fetch data from various sources, such as data models, JavaScript files, and backend services. You can also connect your UI to Simulink to load live data from a Simulink simulation.

You can connect these data sources to the current frame of a timeline, creating animation when the backend changes the current frame property.

For example, you could connect the speed value from a backend to a tachometer dial in a cluster. As the speed value is increased or decreased from the backend, it moves the needle animation from one end of the timeline to the other.

For more information, see Simulating Complex Experiences.

Programmatic Animation

You can control property animation programmatically. Property animations are created by binding **Animation** components to property values of component instances to gradually change the property values over time. The property animations apply smooth movement by interpolating values between property value changes. They provide timing controls and enable different interpolations through easing curves.

Developers can control the execution of property animations by using the , , , , , and functions.`start()stop()resume()pause()restart()complete()`

You can create instances of preset animation components available in **Components** > **Default Components** > **Animation** to create animations depending on the type of the property and the behavior that you want.

For more information about **Animation** components and their properties, see Animations.

Component	Use Case
Property Animation	Applying animation when the value of a property changes. Color and number animations are property animation types for specific purposes.

Color Animation	Applying animation when a color value changes.
Number Animation	Applying animation when a numerical value changes.
Parallel Animation	Running animations in parallel.
Sequential Animation	Running animations sequentially.
Pause Animation	Creating a step in a sequential animation where nothing happens for a specified duration.
Script Action	Executing JavaScript during an animation.

‹ Motion Design Creating Timeline Animations ›

Qt DOCUMENTATION

Contribute to Qt

Forum

Wiki

Downloads

Marketplace

Feedback Sign In