自定义效果和材质

Qt 快速 3D 效果和 Qt 快速 3D 材质模块包含一组现成的效果和材质，您可以将其应用于 3D 模型。如果现成的效果和材质不能满足您的需求，您可以创建自定义效果和材质。每个效果或材质都必须有一个片段着色器，用于实现计算着色颜色所需的所有功能。材料系统还提供现成的功能，以帮助您实现材料。如果 3D 效果和材质未显示在“组件”中，则应将“**QtQuick3D.效果**”和“**QtQuick3D.材质**”模块添加到项目中，如添加和移除模块中所述。

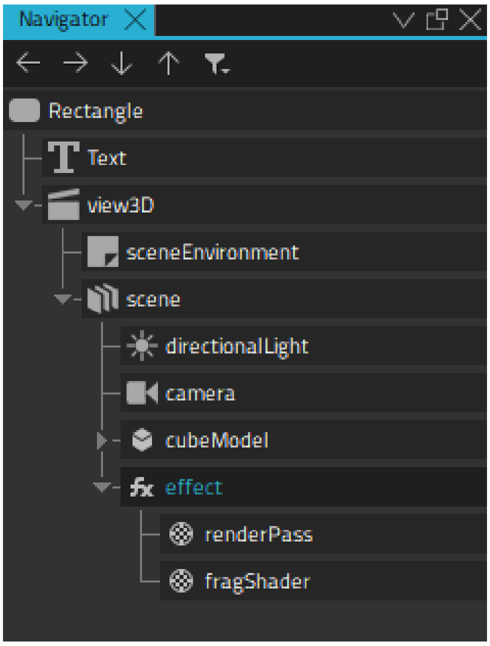
材质系统支持介电、金属和透明材质、点光源、区域光源、环境光遮蔽、阴影、双面多边形、折射率和片段截止（遮罩）。有关更多信息，请参阅 Qt 快速 3D 自定义材料参考。

使用组件> QtQuick3D > Qt **快速 3D 自定义着色器工具中的组件**来创建自定义效果和材质。您可以在 **QtQuick3D 效果**> Qt **快速** 3D **自定义着色器工具**>**组件**中找到效果组件，并在组件中找到**自定义材质组件**> QtQuick3D **材质**> Qt **快速** 3D **自定义着色器工具**。有关着色器实用程序和命令及其属性的详细信息，请参阅自定义着色器。

> **注意：** 必须使用其他工具创建实际的着色器源文件，并将其复制到项目文件夹中。然后，您可以在自定义效果或材质属性中指定源文件名。要在着色器文件中使用自定义*制服，必须将它们指定为自定义效果或材质组件的* QML 属性。Qt设计工作室根据属性值自动生成着色器的制服。

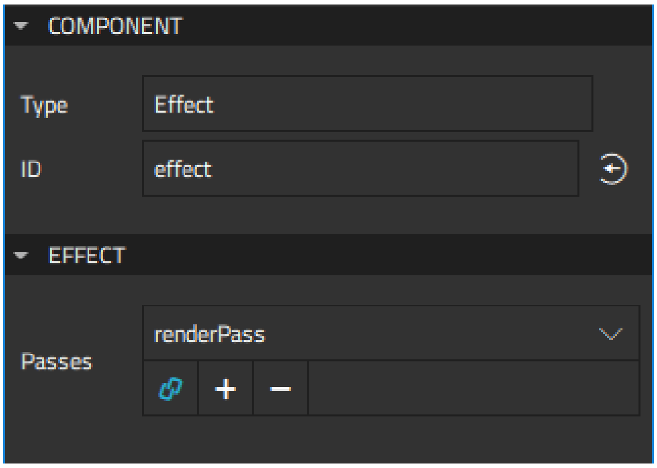
创建自定义效果

默认情况下，自定义效果组件在片段阶段包含“传递”组件和“着色器”组件。您可以将刀路、着色器和其他着色器实用程序添加到效果中。



创建片段着色器组件时，占位符表示着色器文件的路径。指定要在着色器属性中使用的着色器文件的路径

1. 将**效果**组件从组件的 **Qt 快速 3D 自定义着色器** Utils 选项卡**拖放到**“导航”中的模型组件。
2. 在“**导航**”中选择自定义效果组件，以在“属性”视图中编辑其属性的值。



COMPONENT

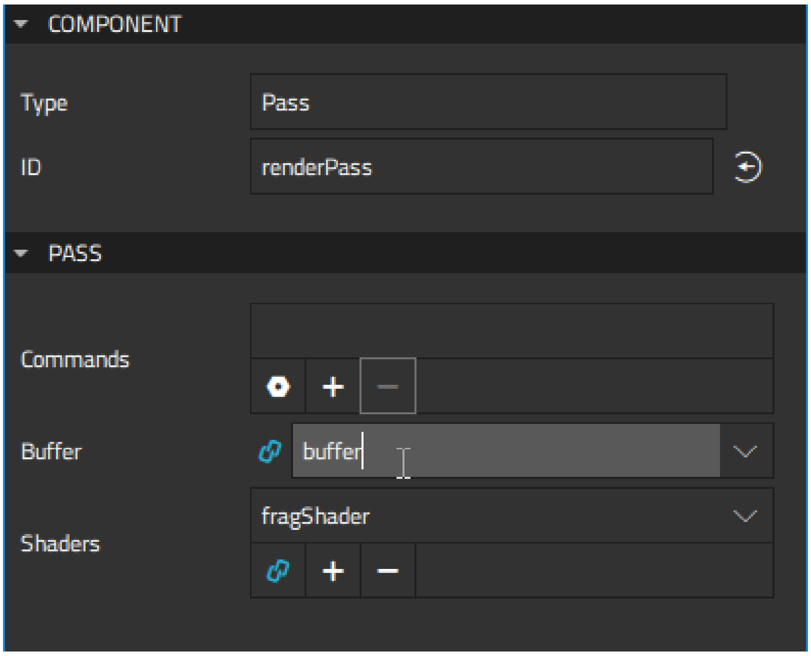
Type: Effect

ID: effect

EFFECT

Passes: renderPass

3. 在**“刀路**”字段中，选择效果的刀路组件。
4. 在“**导航”**中选择刀路组件，以在**“属性”**中为其属性指定值。



COMPONENT

Type: Pass

ID: renderPass

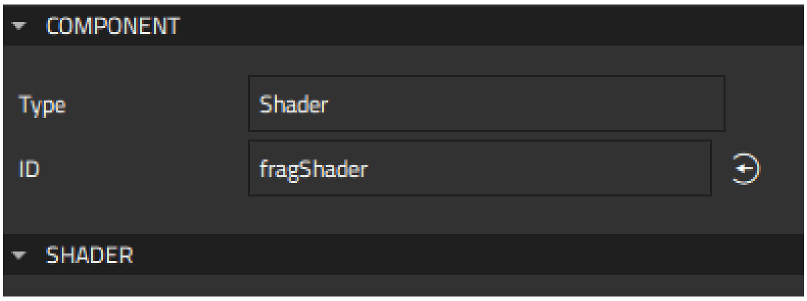
PASS

Commands

Buffer: buffer

Shaders: fragShader

5. To execute commands during the pass, drag-and-drop the following command components from **Component** to the custom material in **Navigator**: **Blending**, **Buffer Blit**, **Buffer Input**, **Cull Mode**, **Depth Input**, **Render State**, and **Set Uniform Value**. Then select the commands in the **Commands** field.
6. To allocate a buffer for the pass, drag-and-drop a **Buffer** component to the custom material. Then select the buffer in the **Buffer** field.
7. Select the shader component in **Navigator** to set the path to the shader files in the **Source** field in **Properties**.

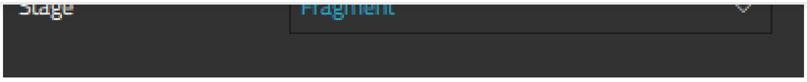


COMPONENT

Type: Shader

ID: fragShader

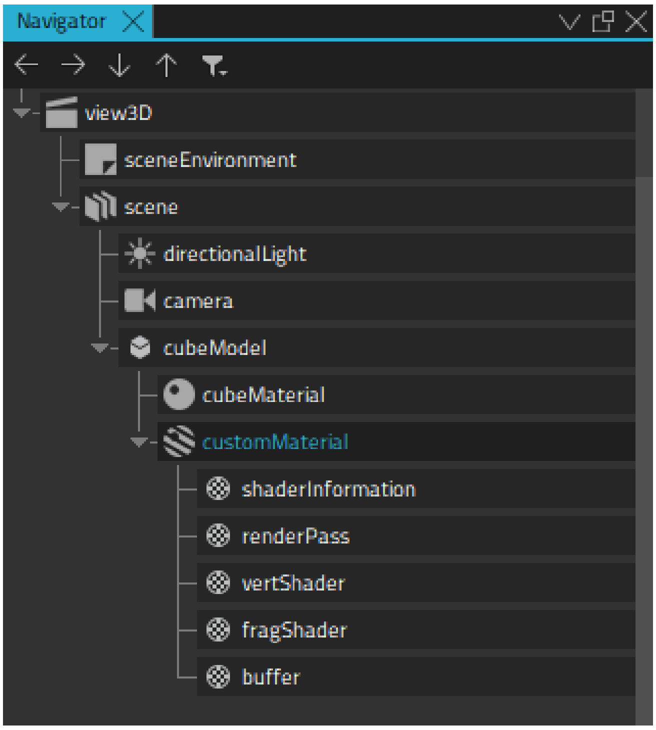
SHADER



Stage Fragment

Creating Custom Materials

By default, a Custom Material component contains two Shader components, a Shader Info component, and a Pass component. You can add shaders, passes, and other shader utilities to the material.



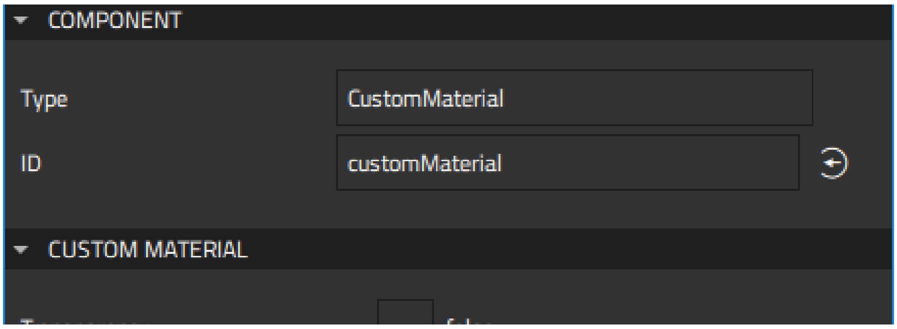
By default, fragment and vertex shaders are created with placeholders for the paths to the shader files. Specify the paths to the shader files to use in the shader properties.

The Shader Info component specifies the shader component and version, as well as the options used by the shader based on the selected shader key values, such as diffuse or specular lighting, refraction, transparency, displacement, transmissiveness, glossiness, and alpha cutout.

The shaders are used with the Pass component to create the resulting material. A pass can contain multiple rendering passes and other commands. You can use a Buffer component to allocate a buffer for storing intermediate rendering results.

To create a custom material:

1. Drag-and-drop a **Custom Material** component from the **Qt Quick 3D Custom Shader Utils** tab of **Component** to a Model component in **Navigator**.
2. Select the custom material component in **Navigator** to edit the values of its properties in the **Properties** view.

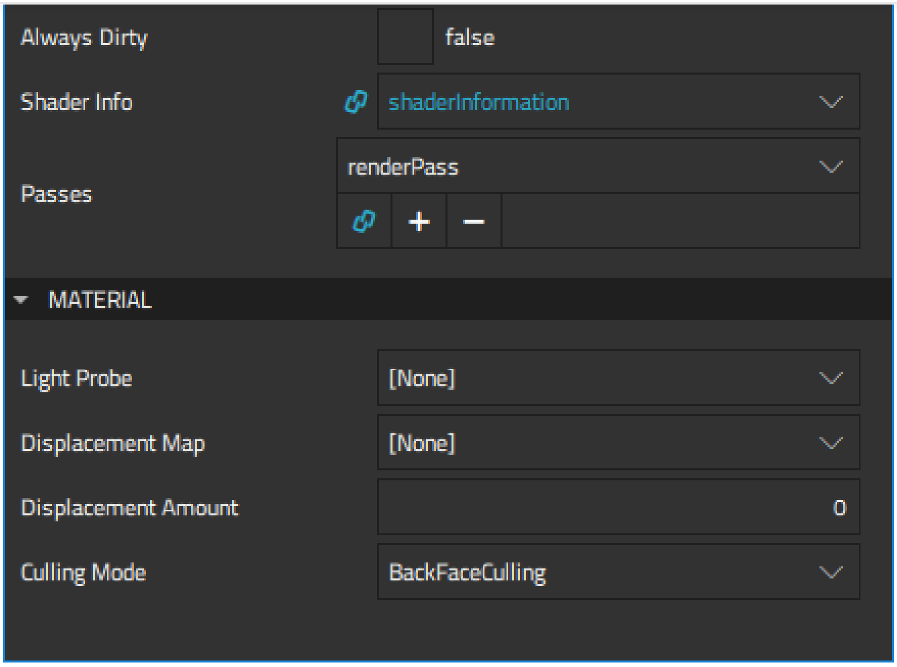


COMPONENT

Type CustomMaterial

ID customMaterial

CUSTOM MATERIAL



Always Dirty □ false

Shader Info shaderInformation

Passes renderPass

MATERIAL

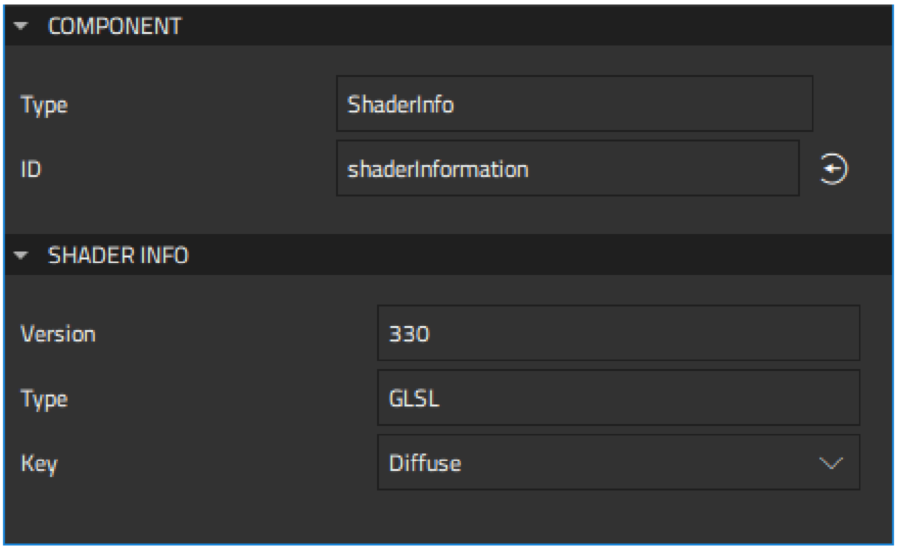
Light Probe [None]

Displacement Map [None]

Displacement Amount 0

Culling Mode BackFaceCulling

3. Select the **Transparency** check box to make the material transparent.
4. Select the **Refraction** check box to specify that the material is reflective.
5. Select the **Always dirty** check box to determine that the material needs to be refreshed every time it is used.
6. In the **Shader Info** field, select the shader info component to use.
7. In the **Passes** field, select the pass components for the effect.
8. In the **Material** group, select the light probe, displacement map and amount, and culling mode to use.
9. Select the shader info component in **Navigator** to specify values for its properties in **Properties**.



COMPONENT

Type ShaderInfo

ID shaderInformation

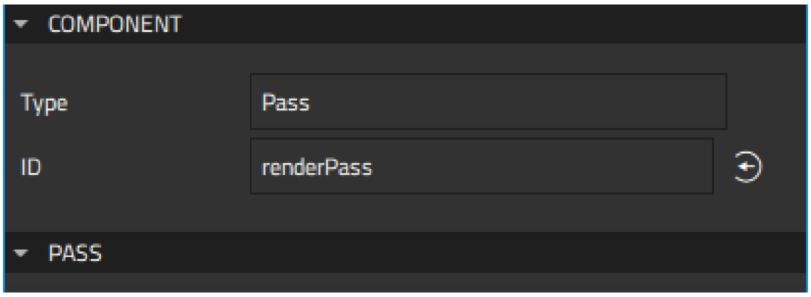
SHADER INFO

Version 330

Type GLSL

Key Diffuse

10. Select the pass component in **Navigator** to specify values for its properties in **Properties**.

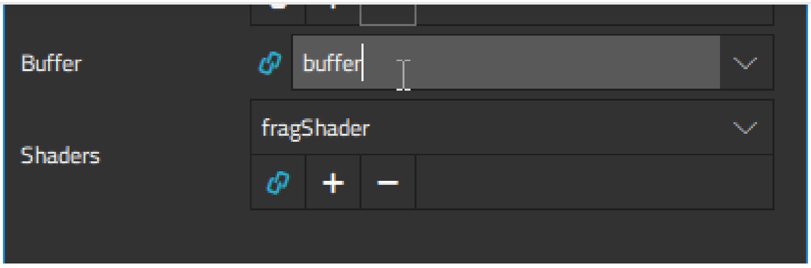


COMPONENT

Type Pass

ID renderPass

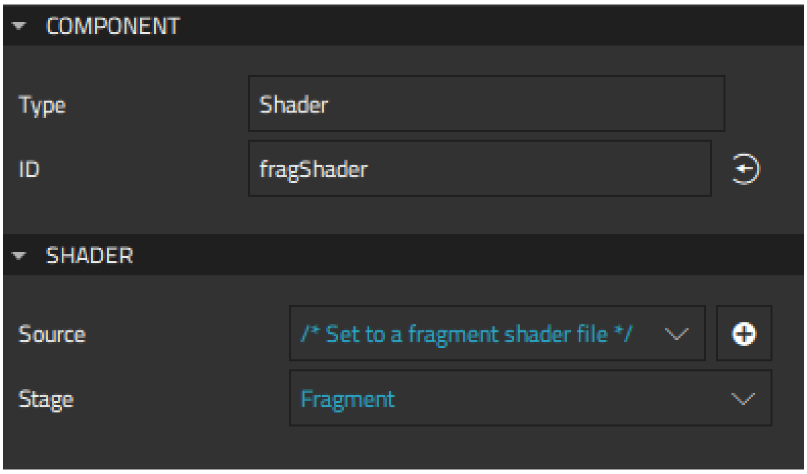
PASS



Buffer buffer

Shaders fragShader

11. To execute commands during the pass, drag-and-drop the following command components from **Component** to the pass component in **Navigator**: **Blending**, **Buffer Blit**, **Buffer Input**, **Cull Mode**, **Depth Input**, **Render State**, and **Set Uniform Value**. The command components are created at the same level as the pass component and automatically added to the **Commands** field.
12. To allocate a buffer for the pass, drag-and-drop a **Buffer** component to the custom material. Then select the buffer in the **Buffer** field.
13. To add a shader to the pass, drag-and-drop the **Shader** component from the **Component** to the pass component in **Navigator**. The shader components are created at the same level as the pass component and automatically added to the **Shaders** field.
14. Select the shader components in **Navigator** to set the paths to the shader files in the **Source** field in **Properties**.



COMPONENT

Type Shader

ID fragShader

SHADER

Source /* Set to a fragment shader file */

Stage Fragment

Creating Shader Files

The requirements set for shaders that you can use in custom effects and materials are described in Qt Quick 3D Custom Material Reference.

If you use custom uniforms in the shader files, you must specify them as QML properties for the custom effect or material component. Qt Design Studio automatically generates the uniforms based on the property values.

For example, the following code snippet shows fragment shader code that uses two uniforms: and .`uTextureInUseuInputTexture`

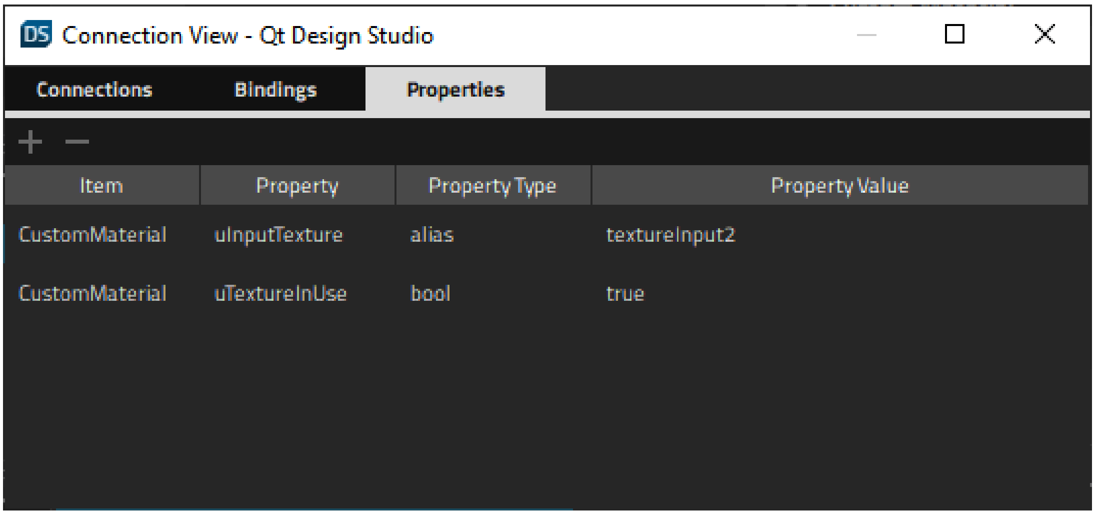
```
out vec4 fragColor;

in vec3 pos;
in vec3 texCoord0;

void main() {

    vec4 textCol;
    if (uTextureInUse)
    }
```

To use the above fragment shader in a custom effect or material component, you must remove the uniforms from the shader code and define them as properties for the component on the **Properties** tab in the Connections view.



Item	Property	Property Type	Property Value
CustomMaterial	uInputTexture	alias	textureInput2
CustomMaterial	uTextureInUse	bool	true

For more information about adding properties, see Specifying Custom Properties.

‹ Custom Shaders Lights ›

Training

Contact Us

Customer Success

Community

Contribute to Qt

Forum

Wiki

Downloads

Marketplace

Feedback Sign In