```cpp
//------------------------------------------------------------------------
// Copyright(c) 2025 Void Audio.
//------------------------------------------------------------------------
#pragma once

#include "vstgui\plugin-bindings\vst3editor.h"
#include "vstgui\lib\cbitmap.h"


#include "licenser.h"


namespace VOID {


class RendEditorDelegate : public  VSTGUI::VST3EditorDelegate, public VSTGUI::IControlListener
{

protected:
    VSTGUI::CBitmap* bitmap1 = nullptr;//createBitmapFromPath("shell1.png");
    VSTGUI::CBitmap* bitmap2 = nullptr;//createBitmapFromPath("shell2.png");
    //VSTGUI::CBitmap* bitmap3 = createBitmapFromPath("resource\\shell\\licenseShell.png");

    // VSTGUI::CBitmap* sheer     = nullptr;
    // VSTGUI::CBitmap* swell     = nullptr;
    // VSTGUI::CBitmap* mangle    = nullptr;
    // VSTGUI::CBitmap* msSwitch  = nullptr;
    // VSTGUI::CBitmap* redButton = nullptr;


    VSTGUI::VST3Editor* editor = nullptr;
    VSTGUI::ParameterChangeListener* paramChangeListener = nullptr;
    LicenseOverlayView* licenseOverlay = nullptr;


public:


    RendEditorDelegate()
    {
        bitmap1   = new VSTGUI::CBitmap("shell/shell1.png");
        bitmap2   = new VSTGUI::CBitmap("shell/shell2.png");
        // sheer     = new VSTGUI::CBitmap("Sheer_210.png");
        // swell     = new VSTGUI::CBitmap("Swell_280.png");
        // mangle    = new VSTGUI::CBitmap("Mangle!_210.png");
        // msSwitch  = new VSTGUI::CBitmap("monoswitch_60x162 .png");
        // redButton = new VSTGUI::CBitmap("bigredswitch_200x219.5 .png");
    }



    float isLicenseValid = 0.0f;
    float switchstate    = 0.0f;


    // For example, if your switch parameter is named "SwitchParam"
    Steinberg::Vst::ParamID switchParamID = 13;

    VSTGUI::CView* createCustomView (VSTGUI::UTF8StringPtr name, const VSTGUI::UIAttributes&
    attributes,
                                     const VSTGUI::IUIDescription* description, VSTGUI::VST3Editor* editor)
                                     override
    {

        // If online, return nullptr to allow default behavior
        return nullptr;
```

```
    }




void didOpen(VSTGUI::VST3Editor* editor) override
{
    //using namespace LicenseSpring;

    this->editor = editor;
    if (!editor || !editor->getFrame())
        return;

    auto* frame = editor->getFrame();
    auto* controller = editor->getController();
    if (!controller)
        return;

    // --- Register listener for switch control ---
    if (VSTGUI::CControl* switchControl = findControlByTag(frame, 13)) {
        switchControl->registerControlListener(this);
    }

    // --- Get switch param state ---
    float switchState = 0.f;
    if (auto* param = controller->getParameterObject(13)) {
        switchState = param->getNormalized();
    }

    // --- Get main view container ---
    VSTGUI::CView* rootView = frame->getView(0);
    auto* mainContainer = dynamic_cast<VSTGUI::CViewContainer*>(rootView);
    if (!mainContainer)
        return;

    // --- Inject license overlay ---
    // LicenseOverlayView will check license state internally in its constructor




    // --- Set background based on switch state ---
    if (switchState <= 0.5f) {
        mainContainer->setBackground(bitmap1);
    } else {
        mainContainer->setBackground(bitmap2);
    }

    isLicenseValid = GlobalLicenseState.isLicenseUnlocked.load();

    if (isLicenseValid <= 0.5)
    {
        CRect overlaySize = mainContainer->getViewSize();
        licenseOverlay = new LicenseOverlayView(overlaySize);
        mainContainer->addView(licenseOverlay);
    }
    mainContainer->invalid();
    frame->setZoom(0.6);


}

void willClose(VSTGUI::VST3Editor* editor) override
{
```

```
}
void valueChanged(VSTGUI::CControl* pControl) override {
        if (pControl->getTag() == 13)
        {
            VSTGUI::CViewContainer* myContainer = nullptr;
            VSTGUI::CView* view = editor->getFrame()->getView(0);
            myContainer = dynamic_cast<VSTGUI::CViewContainer*>(view);
            // Get switch value (0.0 = off, 1.0 = on)
            float switchstate = pControl->getValue();
            if (switchstate <= 0.5) {
                myContainer->setBackground(bitmap1);
                myContainer->invalid();
                editor->getFrame()->setZoom(0.6);
            } else if (switchstate > 0.5) {
                myContainer->setBackground(bitmap2);
                myContainer->invalid();
                editor->getFrame()->setZoom(0.6);
                    }

        }
}
VSTGUI::CBitmap* createBitmapFromPath(const std::string& filePath)
{
    VSTGUI::CResourceDescription desc(filePath.c_str());
    return new VSTGUI::CBitmap(desc);
}



private:

// Helper function to find control by tag using documented VSTGUI methods
VSTGUI::CControl* findControlByTag(VSTGUI::CViewContainer* container, int32_t tag)
{
    if (!container) return nullptr;

    // Use documented CViewContainer methods: getNbViews() and getView()
    uint32_t numViews = container->getNbViews();
    for (uint32_t i = 0; i < numViews; i++) {
        VSTGUI::CView* view = container->getView(i);

        // Try to cast to CControl and check tag
        if (auto control = dynamic_cast<VSTGUI::CControl*>(view)) {
            if (control->getTag() == tag) {
                return control;
            }
        }

        // Recursively search child containers
        if (auto childContainer = dynamic_cast<VSTGUI::CViewContainer*>(view)) {
            VSTGUI::CControl* result = findControlByTag(childContainer, tag);
            if (result) return result;
        }
    }
    return nullptr;


}
};
}
```