The Slate UI Framework

Part 1: Introduction

Gerke Max Preussner
max.preussner@epicgames.com
UE1, UE2 and UE3

UNREAL ENGINE
Slate Design & Principles

Overview

Features

Concepts

Tools

Architecture
- Written entirely in C++
- Platform agnostic (works on mobile and consoles, too!)
- SlateCore module provides low-level functionality
- Slate module contains library of common UI widgets
- Does not require Engine or Editor modules

Current Use Cases
- Unreal Editor
- Standalone desktop applications
- Mobile applications
- In-game UI
Slate Design & Principles

Overview

Features

Concepts

Tools

Styling
• Customize the visual appearance of your UI
• Images [PNGs and Materials], Fonts, Paddings, etc.
• Customizable user-driven layouts

Input Handling
• Keyboard, mouse, joysticks, touch
• Key bindings support

Render Agnostic
• Supports both Engine renderer and standalone renderers

Large Widget Library
• Layout primitives, text boxes, buttons, images, menus, dialogs, message boxes, navigation, notifications, dock tabs, list views, sliders, spinners, etc.
Default Settings (AutoSize):
- Default: Slot is auto-sized
- Slots are packed tightly
- Alignment within the slot does not matter

Fill Size:
Will stretch to fill any available room based on the fill coefficients.
- FillWidth(2)
- FillWidth(1)
- FillWidth(3)

Aspect Ratio:
- Somewhat lengthy text: Apricot
- A somewhat long piece of text

Alignment:
SBBox supports various alignments, padding and a fixed override for
the content's desired size. FixedSize is rarely needed if your content
appears too large, never crush it by forcing a fixed size. Instead,
figure out why the content's DesiredSize is too large! Making extra
room via FixedSize is not as bad.

Options:
- Top Left
- Top Center
- Top Right
- Top Fill
- Center Left
- Center Center
- Center Right
- Center Fill
Slate Design & Principles

Overview

Features

Concepts

Tools

Declarative Syntax
- Set of macros for declaring widget attributes
- Avoids layers of indirection

Composition
- Compose entire widget hierarchies in a few lines of code
- Uses fluent syntax for ease of use
- Preferred over widget inheritance
- Any child slot can contain any other widget type
- Makes it very easy to rearrange UIs in code
class STextButton : public SCompoundWidget {
public:

SLATE_BEGIN_ARGS(SMyButton)
{
}

// The label to display on the button.
SLATE_ATTRIBUTE(FText, Text)

// Called when the button is clicked.
SLATE_EVENT(FOnClicked, OnClicked)
SLATE_END_ARGS()

// Construct this button
void Construct(const FArguments& InArgs);
};

void STextButton::Construct(const FArguments& InArgs)
{
    ChildSlot
    [ SNew(SButton)
        .OnClicked(InArgs._OnClicked)
        [ SNew(STextBlock)
            .Font(FMyStyle::GetFontStyle("TextButtonFont"))
            .Text(InArgs._Text)
            .ToolTipText(LOCTEXT("TextButtonToolTip", "Click Me!"))
        ];
    ];
}
Slate Design & Principles

Overview

Widget Inspector
• Visually debug and analyze your UI
• Can jump directly to widget code in Visual Studio or XCode

Features

UDK Remote
• iOS app for simulating touch devices on your PC
• Remote server is a plug-in (enabled by default)
• Primarily used for game development

Concepts

Tools
Description
UDK Remote allows you to quickly and easily test your mobile-focused Unreal Engine 3 or 4 gameplay directly on your development computer!

What's New in Version 1.2
- Sends to multiple ports at once (editor and game)
- Support for iPhone 5 aspect ratio
- Support for iOS 7

Screenshots

Customer Ratings
We have not received enough ratings to display an average for this version of the application.

More by Epic Games, Inc.
Epic Citadel
View on iTunes
// color wheel
+ SOverlay::Slot()
  + SHorizontalBox::Slot()
    .FillWidth(1.0f)
    .HAlign(HAlign_Center)
    + SColorWheel
      .SelectedColor(this, &ColorPicker::GetCurrentColor)
      .Visibility(this, &ColorPicker::HandleColorPickerModeVisibility, EDColorPickerMode::Value)
      .OnValueChanged(this, &ColorPicker::HandleColor SpectrumValueChanged)
      .OnMouseCaptureBegin(this, &ColorPicker::HandleInteractiveChangeBegin)
      .OnMouseCaptureEnd(this, &ColorPicker::HandleInteractiveChangeEnd)
  + SHorizontalBox::Slot()
    .AutoWidth()
    .Padding(4.0f, 0.0f)
    // saturation slider
    MakeColorSlider(EOColorPickerChannels::Saturation)
  + SHorizontalBox::Slot()
    .AutoWidth()
    // value slider
    MakeColorSlider(EOColorPickerChannels::Value)
+ SOverlay::Slot()
Going A Little Deeper

State Updates

Widget Roles

Anatomy

Attributes

Polling instead of Invalidation
- Avoids duplicate state data
- Exception: Non-trivial data models (use caches instead)
- Performance depends on number of visible widgets

```cpp
SNew(SColorWheel)
 .SelectedColor(this, &SColorPicker::GetCurrentColor)
 .Visibility(this, &SColorPicker::HandleColorPickerModeVisibility, EColorPickerModes::Wheel)
 .OnValueChanged(this, &SColorPicker::HandleColorSpectrumValueChanged)
 .OnMouseCaptureBegin(this, &SColorPicker::HandleInteractiveChangeBegin)
 .OnMouseCaptureEnd(this, &SColorPicker::HandleInteractiveChangeEnd)
```
Going A Little Deeper

Fundamental Widget Types
• SCompoundWidget – Can have nested child widgets
• SLeafWidget – Does not contain child widgets
• SPanel – Base class for layout panels

Special Widgets
• SWidget – Root base class for all widgets (do not inherit!)
• SNullWidget – Empty default widget

User Widgets
• More efficient in terms of compile time
Going A Little Deeper

Common Interfaces
- Arguments – Widget parameters that do not change
- Attributes – Parameters that are polled
- Event handlers – Usually named ‘OnSomeEvent’

Common Internals
- ComputeDesiredSize() - Calculates widget’s desired size
- ArrangeChildren() - Arranges children within allotted area
- OnPaint() – Draws the widget
Going A Little Deeper

**State Updates**

**Widget Roles**

**Anatomy**

**Attributes**

**Common Attributes**
- Enabled state, Visibility, Hit testability
- Tooltip Widget, Tooltip Text, Cursor Style
- Horizontal & Vertical Alignment, Padding

**Attributes Can Be:**
- Constants, i.e. `IsEnabled(false)`
- Delegate bindings,
  i.e. `IsEnabled(this, &SMyWidget::HandleIsEnabled)`
Upcoming Features

Slate Optimizations
- Decreased impact on compilation time
- Split built-in widget library into multiple modules

Unreal Motion Graphics (UMG)
- Artist friendly WYSIWYG editor
- 2D transformation and animation with Sequencer
- Blueprint integration
- Better styling system
Questions?

Documentation, Tutorials and Help at:

- AnswerHub: http://answers.unrealengine.com
- Engine Documentation: http://docs.unrealengine.com
- Official Forums: http://forums.unrealengine.com
- Community Wiki: http://wiki.unrealengine.com
- YouTube Videos: http://www.youtube.com/user/UnrealDevelopmentKit
- Community IRC: #unrealengine on FreeNode

Unreal Engine 4 Roadmap

- imgtfy.com/?q=Unreal+engine+Trello+
The Slate UI Framework

Part 2: Game UI & Unreal Motion Graphics

Gerke Max Preussner
max.preussner@epicgames.com
Current In-Game UI Features

**HUD Canvas**
- FCanvas
  - Low-level C++ API for drawing directly to the screen
  - Has been part of Unreal Engine for many years
  - All functions are in FCanvas class
  - DrawText(), DrawTexture(), DrawTile(), etc.
  - Use AHUD.Canvas to access the canvas object

**VP Widgets**

**Game Menus**

**HHitProxy**
- Provides basic interaction support for FCanvas
- Create one hit proxy per interactive object
- Hit proxy ID is sent to GPU for per-pixel hit tests
Current In-Game UI Features

HUD Canvas

UGameViewportClient
  • Allows usage of Slate widgets inside game viewport
  • Use all features of Slate (except SWindow)
  • Add/RemoveViewportWidgetContent()

VP Widgets

Things to keep in mind
  • All added widgets will be layered on top of each other (SOverlay)
  • Widgets should use TWeakObjPtr for UObject references

Game Menus
void APlayerController::CreateTouchInterface()
{
    ULocalPlayer* LocalPlayer = Cast<ULocalPlayer>(Player);

    // do we want to show virtual joysticks?
    if (LocalPlayer && LocalPlayer->ViewportClient && SVirtualJoystick::ShouldDisplayTouchInterface())
    {
        // load what the game wants to show at startup
        FStringAssetReference DefaultTouchInterfaceName = GetDefault<UInputSettings>().DefaultTouchInterface;

        if (DefaultTouchInterfaceName.IsValid())
        {
            // create the joystick
            VirtualJoystick = SNew(SVirtualJoystick);

            // add it to the player's viewport
            LocalPlayer->ViewportClient->AddViewportWidgetContent(VirtualJoystick.ToSharedRef());

            // activate this interface if we have it
            UTouchInterface* DefaultTouchInterface = LoadObject<UTouchInterface>(NULL, *DefaultTouchInterfaceName.ToString());
            if (DefaultTouchInterface != NULL)
            {
                DefaultTouchInterface->Activate(VirtualJoystick);
            }
        }
    }
}
Current In-Game UI Features

HUD Canvas

VP Widgets

Game Menus

The Hard Way
- Use FCanvas to draw your own menus
- Not recommended

The Custom Way
- Use HUD Widgets to create any menu layout

The Lazy Way
- Use GameMenuBuilder for paged menus
- FGameMenuPage - Single menu page
- FGameMenuItem - An option in a menu page
- Can be customized and styled
- Mostly used for settings screens
Unreal Motion Graphics

Overview

One UI Solution To Rule Them All
- Built on top of Slate
- Adds real-time animation and transformation to widgets
- Integrated with Blueprints
- WYSIWYG Editor for artists and designers
- No programming required (unless you want to)
- Not officially released yet, but already in the code base
Unreal Motion Graphics

Overview

Adding Behavior to your UI

• Designers should not have to write code!
• Blueprints allow scripting of UI
• You can still use C++ as well, but probably won’t
Unreal Motion Graphics

Overview

Currently working on:

- Materials!
- 2D Transforms
- Style assets
- Special effects
- In-game 3D UI
- Workflow polish

Scripting

Upcoming
Arbitrary 2D Transforms
Questions?

Documentation, Tutorials and Help at:

- AnswerHub: http://answers.unrealengine.com
- Engine Documentation: http://docs.unrealengine.com
- Official Forums: http://forums.unrealengine.com
- Community Wiki: http://wiki.unrealengine.com
- YouTube Videos: http://www.youtube.com/user/UnrealDevelopmentKit
- Community IRC: #unrealengine on FreeNode

Unreal Engine 4 Roadmap

- lmgtfy.com/?q=Unreal+engine+Trello+