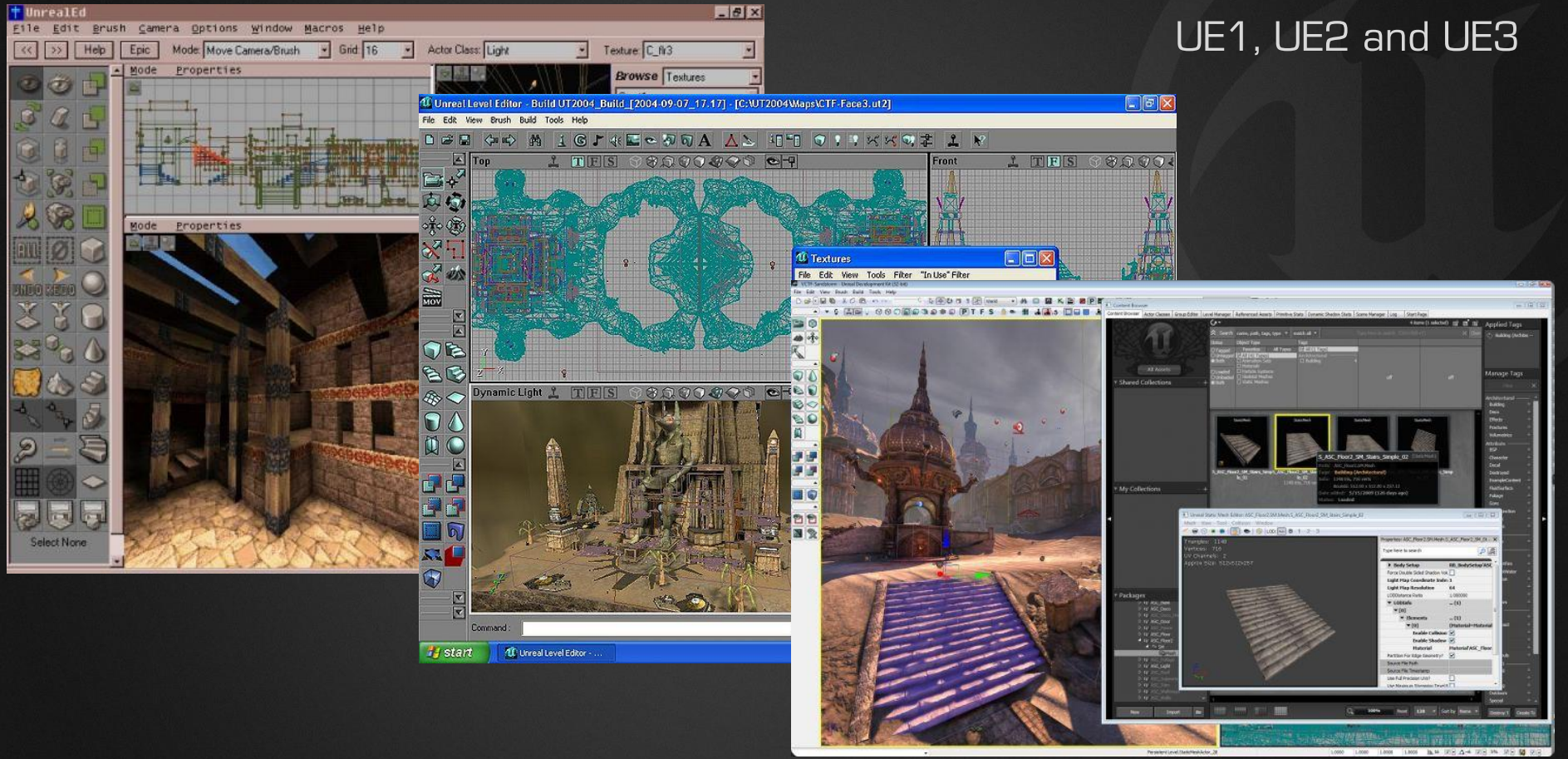
The Slate UI Framework

Architecture & Tools

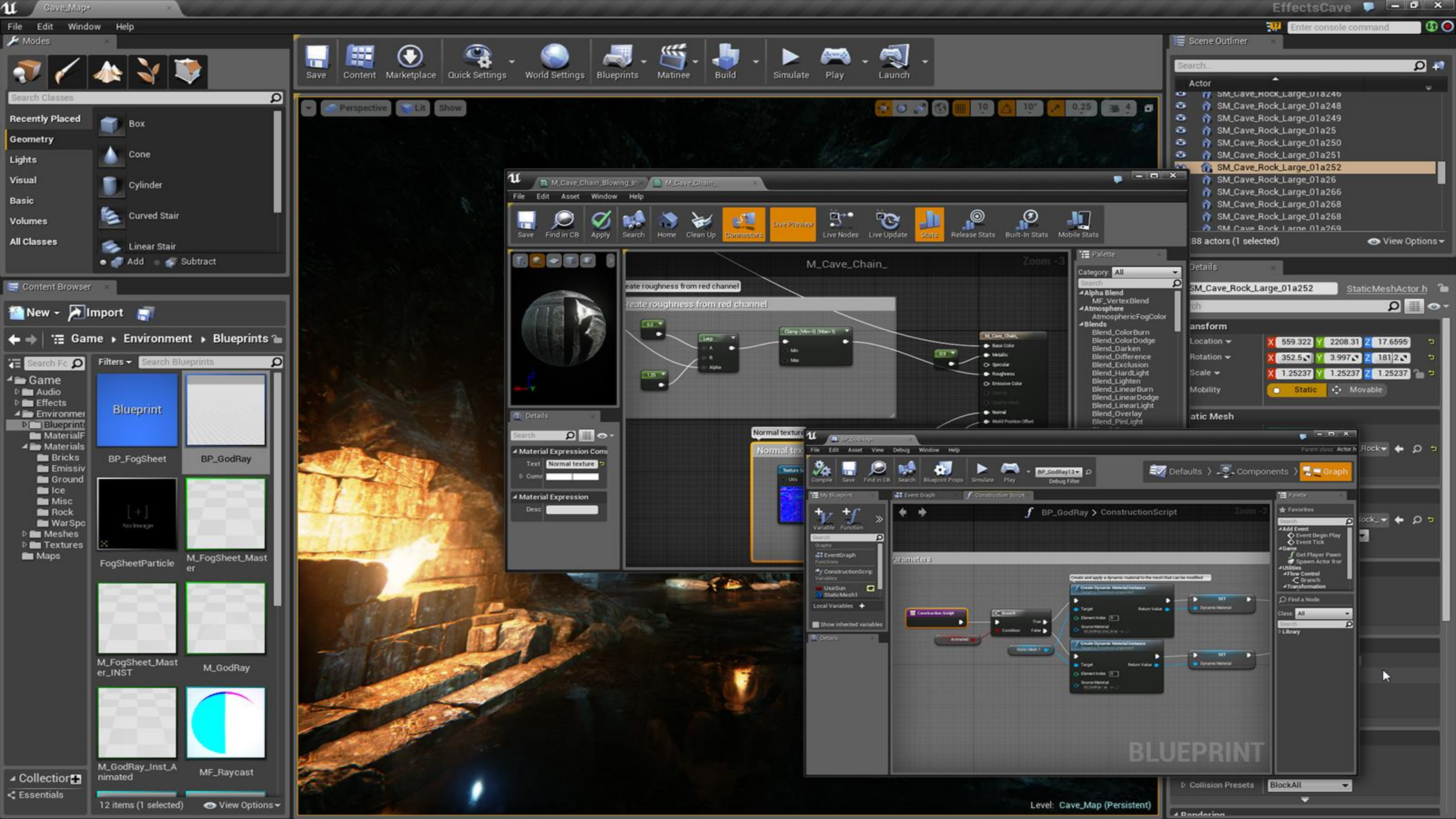
Gerke Max Preussner

max.preussner@epicgames.com

UE1, UE2 and UE3



UNREAL ENGINE


Cave_Map*
EffectsCave
File
Edit
Window
Help
Enter console command
Modes
Save
Content
Marketplace
Quick Settings
World Settings
Blueprints
Matinee
Build
Simulate
Play
Launch
Scene Outliner
Search Classes
Recently Placed
Geometry
Lights
Visual
Basic
Volumes
All Classes
Box
Cone
Cylinder
Curved Stair
Linear Stair
Add
Subtract
Perspective
Lit
Show
Actor
SM_Cave_Rock_Large_01a248
SM_Cave_Rock_Large_01a249
SM_Cave_Rock_Large_01a25
SM_Cave_Rock_Large_01a250
SM_Cave_Rock_Large_01a251
SM_Cave_Rock_Large_01a252
SM_Cave_Rock_Large_01a26
SM_Cave_Rock_Large_01a266
SM_Cave_Rock_Large_01a268
View Options
Details
SM_Cave_Rock_Large_01a252
Location
Rotation
Scale
Mobility
Static
Movable
559.322
2208.31
17.6595
1.25237
Content Browser
New
Import
Game
Environment
Blueprints
Filters
Search Blueprints
Game
Audio
Effects
Materials
Bricks
Ground
Ice
Misc
Rock
Meshes
Textures
Maps
Blueprint
BP_FogSheet
BP_GodRay
FogSheetParticle
M_FogSheet_Master
M_FogSheet_Master_INST
M_GodRay
M_GodRay_Inst_Animated
MF_Raycast
Collections
Essentials
12 items (1 selected)
View Options
File
Edit
Asset
Window
Help
Save
Find in CB
Apply
Search
Home
Clean Up
Connectors
Live Preview
Live Nodes
Live Update
Stats
Release Stats
Built-In Stats
Mobile Stats
M_Cave_Chain_
Create roughness from red channel
Palette
Category
All
Alpha Blend
Atmosphere
Blends
Blend_ColorBurn
Blend_ColorDodge
Blend_Darken
Blend_Difference
Blend_Exclusion
Blend_HardLight
Blend_Lighten
Blend_LinearBurn
Blend_LinearDodge
Blend_LinearLight
Blend_Overlay
Blend_PinLight
Material Expression
Normal texture
Compile
Simulate
Play
Defaults
Components
Graph
BP_GodRay
ConstructionScript
Parameters
Collision Presets
BlockAll
BLUEPRINT
Level: Cave_Map (Persistent)

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.

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

```
// Example custom button (some details omitted)

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 );
};
```

```
// Button implementation (some details omitted)

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

Features

Concepts

Tools

Widget Gallery

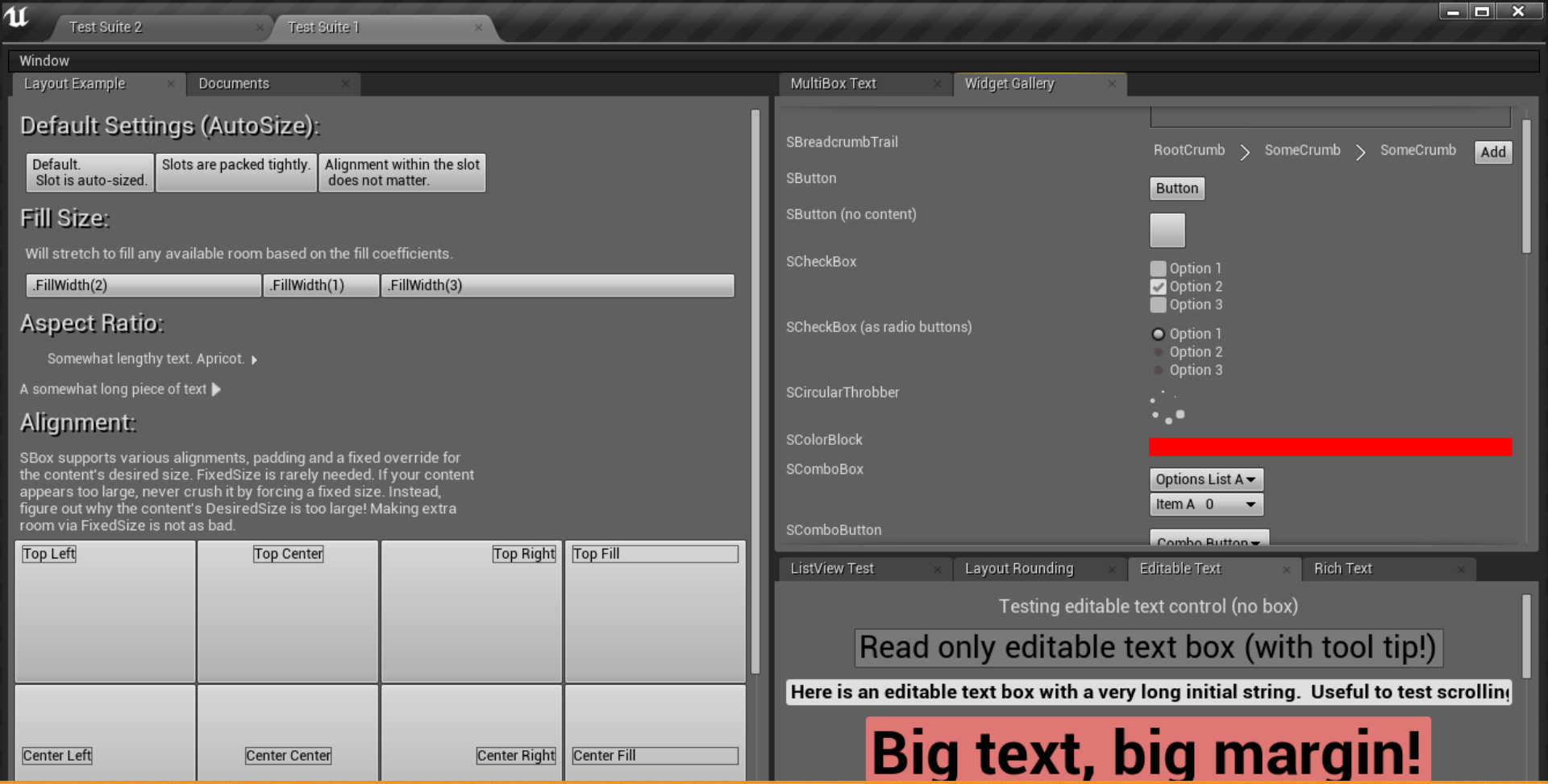
- Demonstrates all available Slate widgets and layouts

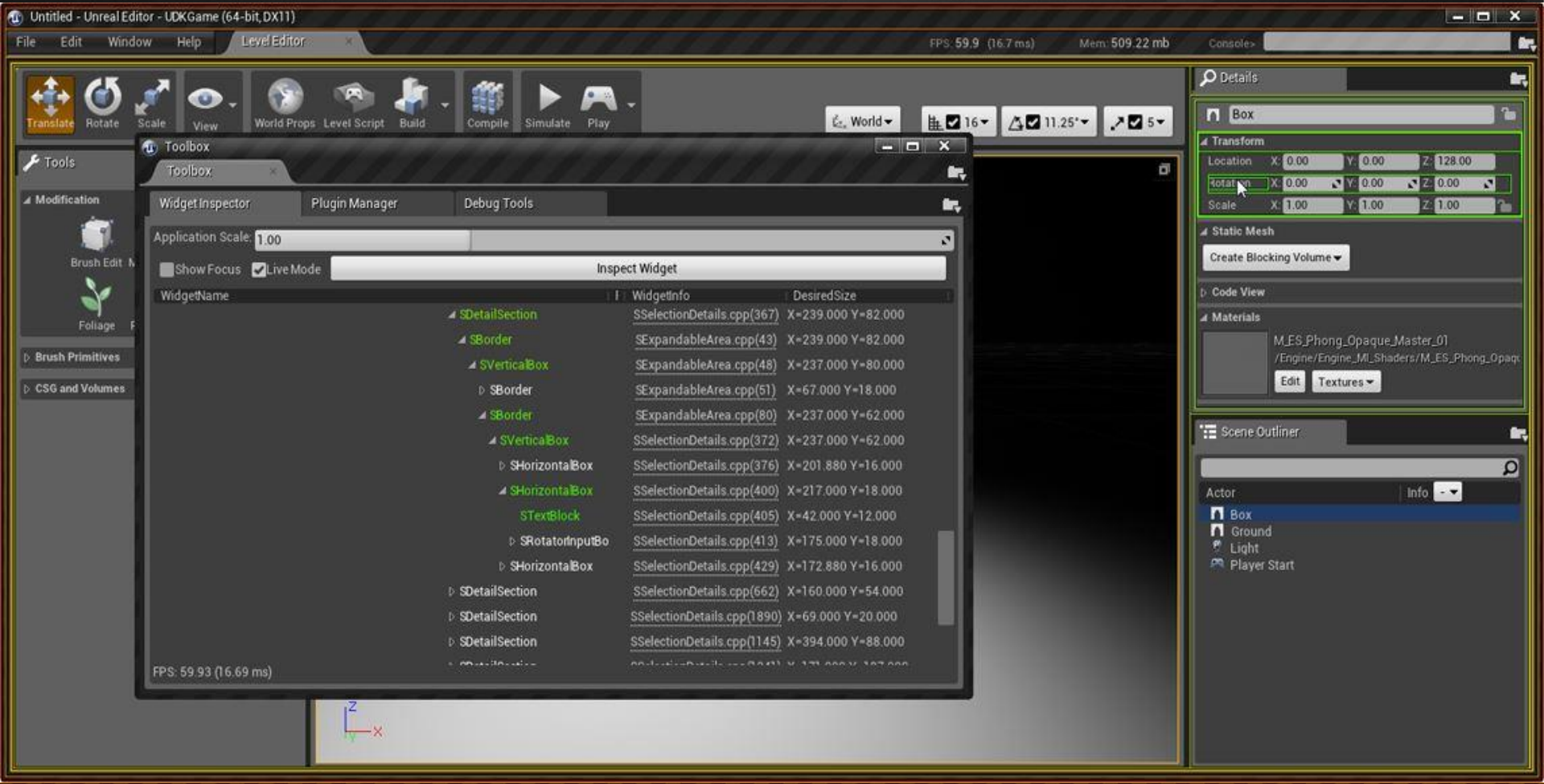
Widget Inspector

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

UDK Remote

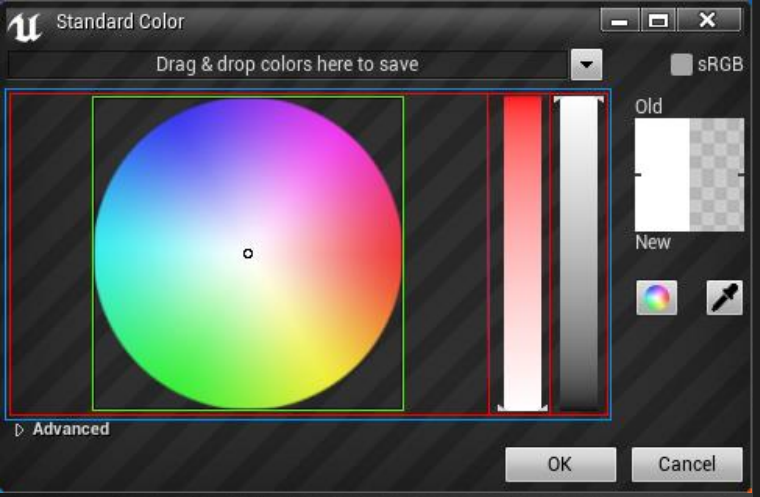
- iOS app for simulating touch devices on your PC
- Remote server is a plug-in (enabled by default)
- Primarily used for game development

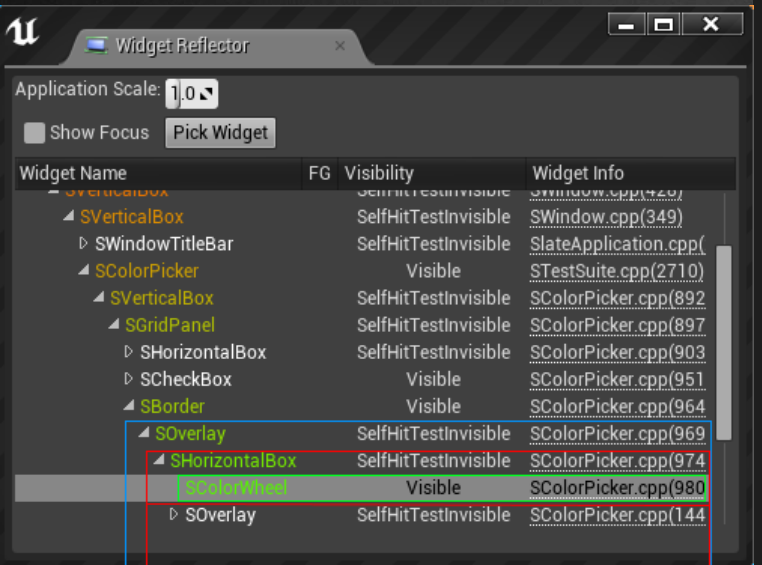

Test Suite 2
Test Suite 1
Window
Layout Example
Documents
Default Settings (AutoSize):
Default.
Slot is auto-sized.
Slots are packed tightly.
Alignment within the slot
does not matter.
Fill Size:
Will stretch to fill any available room based on the fill coefficients.
.FillWidth(2)
.FillWidth(1)
.FillWidth(3)
Aspect Ratio:
Somewhat lengthy text. Apricot.
A somewhat long piece of text
Alignment:
SBox 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.
Top Left
Top Center
Top Right
Top Fill
Center Left
Center Center
Center Right
Center Fill
MultiBox Text
Widget Gallery
SBreadcrumbTrail
RootCrumb
SomeCrumb
SomeCrumb
Add
SButton
Button
SButton (no content)
SCheckBox
Option 1
Option 2
Option 3
SCheckBox (as radio buttons)
Option 1
Option 2
Option 3
SCircularThrobber
SColorBlock
SComboBox
Options List A
Item A 0
SComboButton
ListView Test
Layout Rounding
Editable Text
Rich Text
Testing editable text control (no box)
Read only editable text box (with tool tip!)
Here is an editable text box with a very long initial string. Useful to test scrollin
Big text, big margin!
UNREAL
ENGINE
Demo



UNREAL ENGINE

Demo






```
[
SNew(SOverlay)

// color wheel
+ SOverlay::Slot()
    [
        SNew(SHorizontalBox)

        + SHorizontalBox::Slot()
            .FillWidth(1.0f)
            .HAlign(HAlign_Center)
            [
                SNew(SColorWheel)
                    .SelectedColor(this, &SColorPicker::GetCurrentColor)
                    .Visibility(this, &SColorPicker::HandleColorPickerModeVisibility, ECo
                    .OnValueChanged(this, &SColorPicker::HandleColorSpectrumValueChanged)
                    .OnMouseCaptureBegin(this, &SColorPicker::HandleInteractiveChangeBegi
                    .OnMouseCaptureEnd(this, &SColorPicker::HandleInteractiveChangeEnd)
            ]

        + SHorizontalBox::Slot()
            .AutoWidth()
            .Padding(4.0f, 0.0f)
            [
                // saturation slider
                MakeColorSlider(EColorPickerChannels::Saturation)
            ]

        + SHorizontalBox::Slot()
            .AutoWidth()
            [
                // value slider
                MakeColorSlider(EColorPickerChannels::Value)
            ]
    ]

// color spectrum
```

UDK Remote

View More by This Developer

By Epic Games, Inc.

Open iTunes to buy and download apps.

View in iTunes

This app is designed for both iPhone and iPad

Free

Category: Utilities
Updated: Oct 09, 2013
Version: 1.2
Size: 5.6 MB
Language: English
Seller: Epic Games, Inc.

Rated 4+

Compatibility: Requires iOS 5.0 or later. Compatible with iPhone, iPad, and iPod touch. This app is optimized for iPhone 5.

Customer Ratings

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

All Versions:
★★★★ 41 Ratings

More by Epic Games, Inc.

Epic Citadel
View In iTunes

Description

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

UDK Remote Support ...More

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

...More

Screenshots

iPhone | iPad

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

```
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

State Updates

Widget Roles

Anatomy

Attributes

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

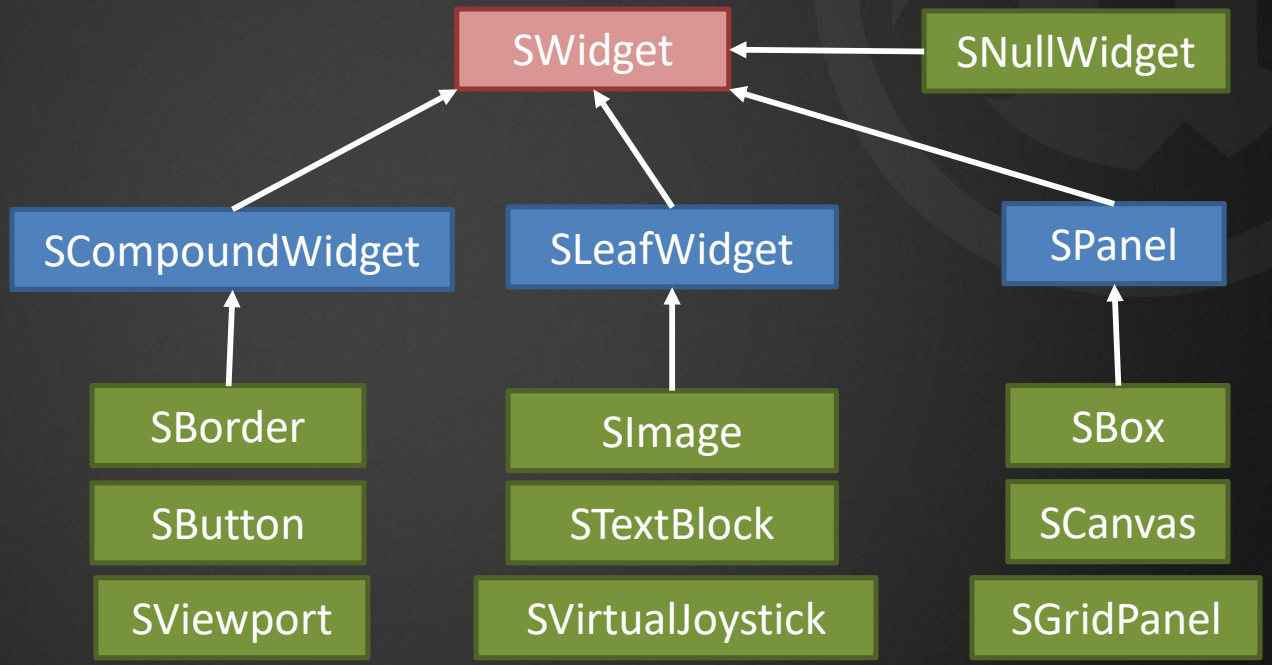
Going A Little Deeper

State Updates

Widget Roles

Anatomy

Attributes



Going A Little Deeper

State Updates

Widget Roles

Anatomy

Attributes

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)
- Lambdas are now supported, too!

In-Game UI

HUD Canvas

VP Widgets

Game Menus

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

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

In-Game UI

HUD Canvas

VP Widgets

Game Menus

UGameViewportClient

- Allows usage of Slate widgets inside game view port
- Use all features of Slate (except SWindow)
- Add/RemoveViewportWidgetContent()

Things to keep in mind

- All added widgets will be layered on top of each other (SOverlay)
- Widgets should use TWeakObjPtr for UObject references

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

    // do we want to show virtual joysticks?
    if (LocalPlayer && LocalPlayer->ViewportClient && SVirtualJoystick::ShouldDisplayTouchInterface())
    {
        // in case we already had one, remove it
        if (VirtualJoystick.IsValid())
        {
            Cast<ULocalPlayer>(Player)->ViewportClient->RemoveViewportWidgetContent(VirtualJoystick.ToSharedRef());
        }

        // 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)
            {
                ActivateTouchInterface(DefaultTouchInterface);
            }
        }
    }
}
```


ENGINE

In-Game UI

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

What's New in 4.6?

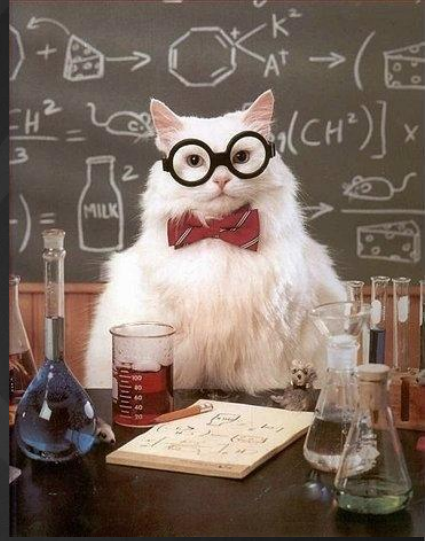
Slate Improvements

- Decreased impact on compilation time
- Split built-in widget library into multiple modules
- Named slots

Unreal Motion Graphics (UMG)

- Artist friendly WYSIWYG editor
- 2D transformation and animation with Sequencer
- Blueprint integration & scripting
- Better styling system (work in progress)

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+