

# UE4移动平台UI优化

郭春彪

chunbiao.guo@epicgames.com

# 目录

## 案例介绍

- 演示工程
- 性能指标
- 优化效果

## 游戏线程优化

- Invalidation Box
- Visibility设置
- Widget Binding
- 蓝图 → C++

## 渲染线程优化

- 合并批次
- 合并贴图
- Retainer Box
- 事件驱动的 Retainer Box
- 切换材质

## 编程技巧

- C++ 开发
- Widget Manager
- 释放贴图内存
- 3D RTT

# 目录

## 案例介绍

- 演示工程
- 性能指标
- 优化效果

## 游戏线程优化

- Invalidation Box
- Visibility设置
- Widget Binding
- 蓝图 → C++

## 渲染线程优化

- 合并批次
- 合并贴图
- Retainer Box
- 事件驱动的 Retainer Box
- 切换材质

## 编程技巧

- C++ 开发
- Widget Manager
- 释放贴图内存
- 3D RTT

# 演示工程





# 测试环境

- 小米4C

CPU型号	高通 骁龙808
CPU频率	1.8GHz
GPU型号	高通 Adreno418
RAM容量	2GB
ROM容量	16GB

- Mobile HDR

# 性能指标





不要使用Stat.Slate

可以使用stat dumpave -num=120 -ms=0.5查看LOG



# 性能指标

- Slate Tick - STAT\_SlateTickTime
  - 游戏线程：Vertex Buffer
- Slate Render - STAT\_SlateRenderingRTTime
  - 渲染线程：UI渲染到Back Buffer
- Widget Render - FWidgetRenderer\_DrawWindow
  - 渲染线程：UI RTT / Retainer Box

	FPS	50.144
	Slate Tick	16.804 ms
	Slate Render	0.881 ms
	Widget Render	0.452 ms



# 优化效果

## 初始性能数据





# 优化效果

开启Invalidation Box



# 优化效果

	FPS	36.881
	Slate Tick	11.908 ms
	Slate Render	8.299 ms
	Widget Render	0 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
Close		

	FPS	38.098
	Slate Tick	1.338 ms
	Slate Render	7.706 ms
	Widget Render	0 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
Close		

	FPS	48.955
	Slate Tick	1.392 ms
	Slate Render	1.494 ms
	Widget Render	3.164 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
Close		

	FPS	58.149
	Slate Tick	0.796 ms
	Slate Render	1.576 ms
	Widget Render	0 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
Close		

# 目录

## 案例介绍

- 演示工程
- 性能指标
- 优化效果

## 游戏线程优化

- Invalidation Box
- Visibility设置
- Widget Binding
- 蓝图 → C++

## 渲染线程优化

- 合并批次
- 合并贴图
- Retainer Box
- 事件驱动的 Retainer Box
- 切换材质

## 编程技巧

- C++ 开发
- Widget Manager
- 释放贴图内存
- 3D RTT



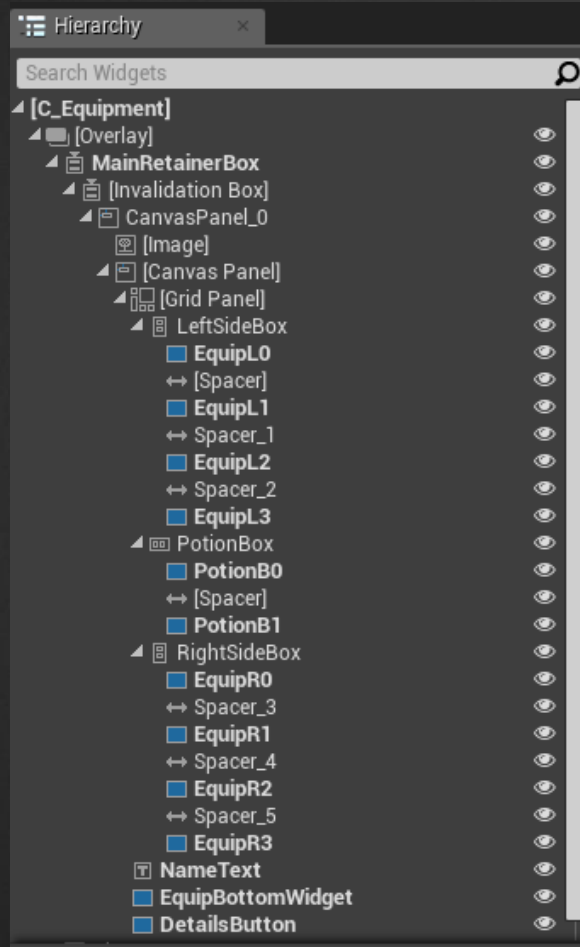
# Invalidation Box

- 示例



- 每帧操作：
  - Grid Panel遍历所有的Child Widgets
  - Image1, Text1, Image2分别计算Draw Elements
  - Grid Panel将Image1和Image2的Draw Elements合并
  - Grid Panel返回2个Draw Elements进行渲染

# Invalidation Box



# Invalidation Box: 原理

- 缓存Draw Elements (Vertex Buffer)
- 用Invalidation Box封装Grid Panel



- 当某个Child Widget的Transform或Visibility等信息变化时, Invalidation Box会重新计算缓存数据



# Invalidation Box: Volatile

- 标志成Volatile的Widget每帧都会重新计算
- 一些属性的Widget Binding会使得Widget变成Volatile
- Check Box放在Invalidation Box下会不起作用，需要设置成Volatile，建议自定义User Widget，用Button实现对应功能

# Invalidation Box: Volatile



- 使用Slate.InvalidationDebugging找出Volatile
- 使用Slate.AlwaysInvalidate测试是否会突然卡顿

# Invalidation Box: 使用



- Invalidation Box自身会被标志成Volatile
- 重复使用的子控件建议不要Invalidation Box, 会有额外计算
- Invalidation Box放在Retainer Box的下层





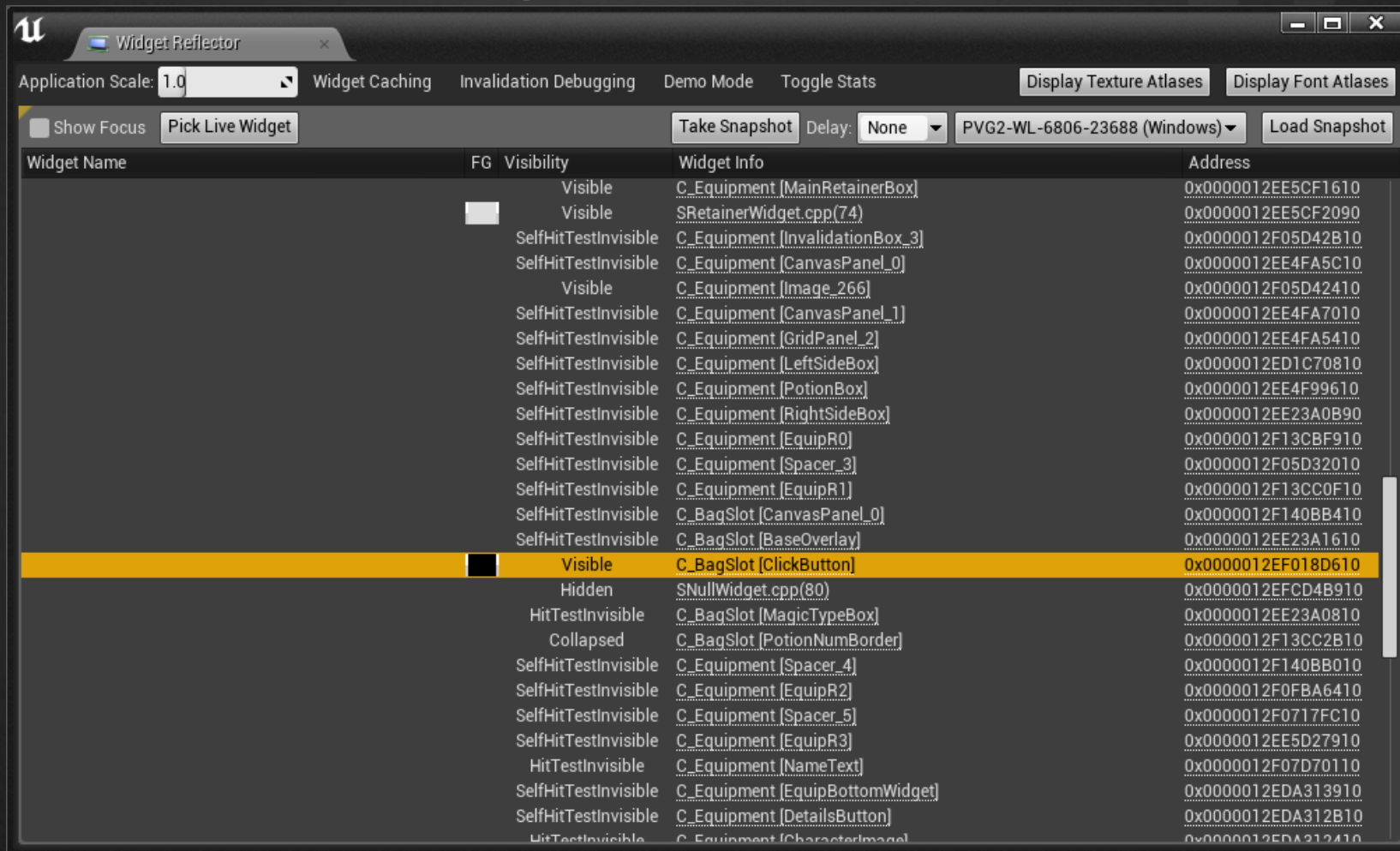
# 可见性 - Widget Visibility

- 是否显示 & 点击测试
- Visible 可见、可点击
- HitTestInvisible 可见、当前控件不可点击、所有子控件不可点击
- SelfHitTestInvisible 可见、当前控件不可点击、不影响子控件
- Hidden 不可见、占用布局空间
- Collapsed 不可见、不占用布局空间

# Widget Visibility

- 大量的Visible会导致点击响应太慢
- Button设置成Visible，其它Widgets可以设置成Self Hit Test Invisible或Hit Test Invisible
- Collapsed不占用布局空间, 略优于Hidden（但不明显）
- Show/Collapse要优于AddToViewport/RemoveFromViewport

# Widget Reflector



Widget Name	FG	Visibility	Widget Info	Address
		Visible	C_Equipment [MainRetainerBox]	0x0000012EE5CF1610
		Visible	SRetainerWidget.cpp(74)	0x0000012EE5CF2090
		SelfHitTestInvisible	C_Equipment [InvalidationBox_3]	0x0000012F05D42B10
		SelfHitTestInvisible	C_Equipment [CanvasPanel_0]	0x0000012EE4FA5C10
		Visible	C_Equipment [Image_266]	0x0000012F05D42410
		SelfHitTestInvisible	C_Equipment [CanvasPanel_1]	0x0000012EE4FA7010
		SelfHitTestInvisible	C_Equipment [GridPanel_2]	0x0000012EE4FA5410
		SelfHitTestInvisible	C_Equipment [LeftSideBox]	0x0000012ED1C70810
		SelfHitTestInvisible	C_Equipment [PotionBox]	0x0000012EE4F99610
		SelfHitTestInvisible	C_Equipment [RightSideBox]	0x0000012EE23A0B90
		SelfHitTestInvisible	C_Equipment [EquipR0]	0x0000012F13CBF910
		SelfHitTestInvisible	C_Equipment [Spacer_3]	0x0000012F05D32010
		SelfHitTestInvisible	C_Equipment [EquipR1]	0x0000012F13CC0F10
		SelfHitTestInvisible	C_BagSlot [CanvasPanel_0]	0x0000012F140BB410
		SelfHitTestInvisible	C_BagSlot [BaseOverlay]	0x0000012EE23A1610
		Visible	C_BagSlot [ClickButton]	0x0000012EF018D610
		Hidden	SNullWidget.cpp(80)	0x0000012EFCD4B910
		HitTestInvisible	C_BagSlot [MagicTypeBox]	0x0000012EE23A0810
		Collapsed	C_BagSlot [PotionNumBorder]	0x0000012F13CC2B10
		SelfHitTestInvisible	C_Equipment [Spacer_4]	0x0000012F140BB010
		SelfHitTestInvisible	C_Equipment [EquipR2]	0x0000012F0FBA6410
		SelfHitTestInvisible	C_Equipment [Spacer_5]	0x0000012F0717FC10
		SelfHitTestInvisible	C_Equipment [EquipR3]	0x0000012EE5D27910
		HitTestInvisible	C_Equipment [NameText]	0x0000012F07D70110
		SelfHitTestInvisible	C_Equipment [EquipBottomWidget]	0x0000012EDA313910
		SelfHitTestInvisible	C_Equipment [DetailsButton]	0x0000012EDA312B10
		HitTestInvisible	C_Equipment [CharacterImage]	0x0000012EDA212410



# Widget Binding

- 在某些属性上Widget Binding会导致对应Widget被放入Volatile List
- 这些属性发生变化，表示对应的控件需要重新计算Vertex Buffer

```
bool STextBlock::ComputeVolatility() const
{
    SCOPE_CYCLE_COUNTER(Stat_SlateTextBlockCV);
    return SLeafWidget::ComputeVolatility()
        || BoundText.IsBound()
        || Font.IsBound()
        || ColorAndOpacity.IsBound()
        || ShadowOffset.IsBound()
        || ShadowColorAndOpacity.IsBound()
        || HighlightColor.IsBound()
        || HighlightShape.IsBound()
        || HighlightText.IsBound()
        || WrapTextAt.IsBound()
        || AutoWrapText.IsBound()
        || WrappingPolicy.IsBound()
        || Margin.IsBound()
        || Justification.IsBound()
        || LineHeightPercentage.IsBound()
        || MinDesiredWidth.IsBound();
}
```

```
bool SProgressBar::ComputeVolatility() const
{
    return SLeafWidget::ComputeVolatility() || Percent.IsBound();
}
```

# Widget Binding

- Widget Binding会每帧Tick执行
- 可以通过C++ Event设置Widget属性

# 蓝图 → C++

- 移动平台上，不建议在Tick或频繁执行的函数中使用复杂的蓝图
- 在C++中声明变量，引擎会自动绑定编辑器中的Widget

```
public:
    UPROPERTY(meta = (BindWidget))
    UTextBlock* OpNameText;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* EquipButton;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* DropButton;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* CloseButton;
```





# 目录

## 案例介绍

- 演示工程
- 性能指标
- 优化效果

## 游戏线程优化

- Invalidation Box
- Visibility设置
- Widget Binding
- 蓝图 → C++

## 渲染线程优化

- 合并批次
- 合并贴图
- Retainer Box
- 事件驱动的 Retainer Box
- 切换材质

## 编程技巧

- C++ 开发
- Widget Manager
- 释放贴图内存
- 3D RTT

# 合并批次



- 不合并批次:
- Canvas Panel
- 合并批次 :
- Grid Panel
- Uniform Grid Panel
- Vertical Box
- Horizontal Box

# 合并批次



3个Draw Call



1个Draw Call



# 合并批次

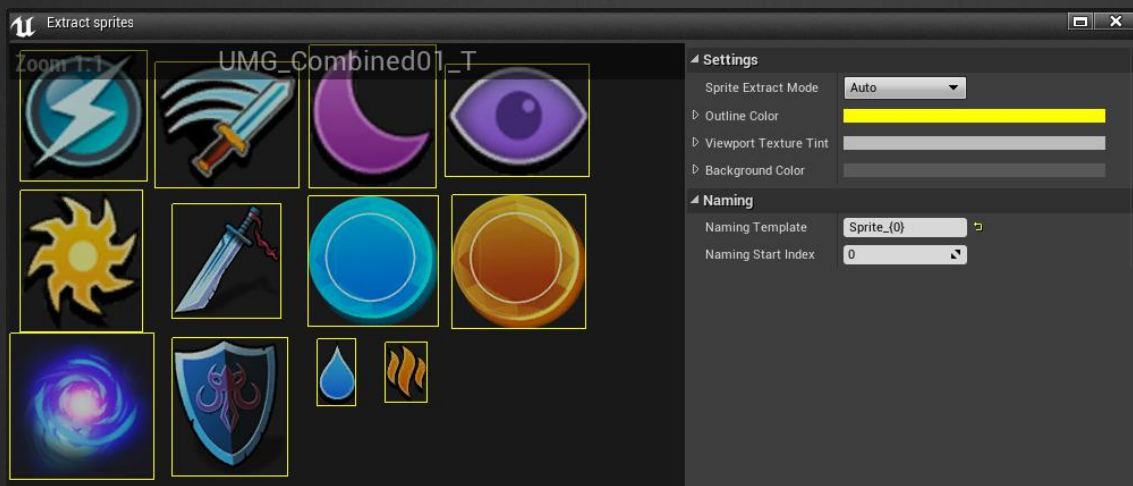
- 通过Stat Slate查看批次：Num Batches

Counters	Average	Max
Num Painted Widgets	29.00	29.00
Num Vertices	12.00	12.00
Num Elements (Prebatch)	4.00	4.00
Num Batches	4.00	4.00
Num Layers	4.00	4.00
Elements (Box)	3.00	3.00
Elements (Text)	0.00	0.00

- 尽量使用可以批次的UI容器，但不用刻意追求合并批次

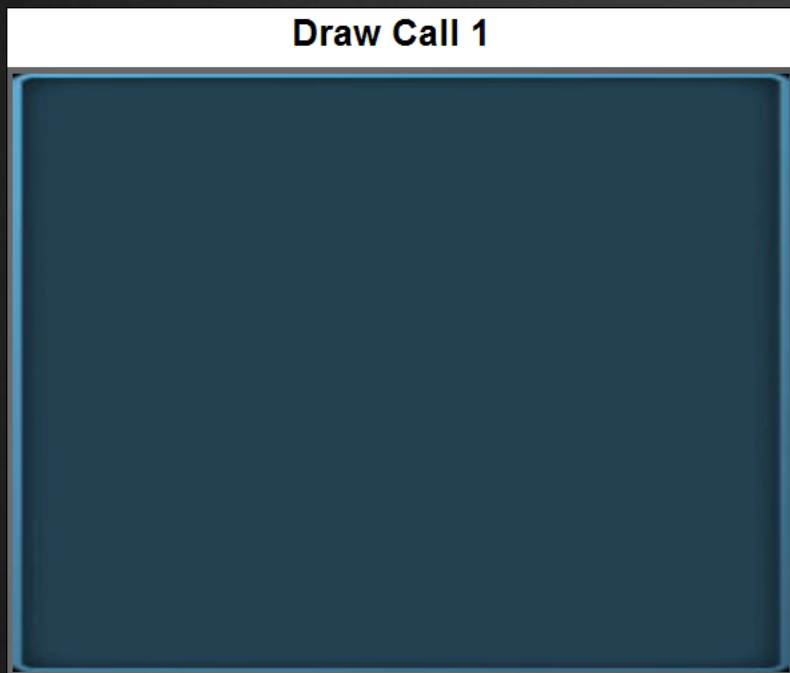
# 合并贴图

- 通过Sprite实现合并贴图功能



- 减少约50个Draw Call，小米4c上性能无明显变化

# 像素填充率



- 背包界面的前5个Draw Call
- 后4个Draw Call的渲染面积接近第一个背景图
- 约5倍的Pixel Shader的执行次数
- UI渲染的瓶颈



# Retainer Box

- 将UI渲染到Render Target，再将Render Target 渲染到屏幕
- 引擎处理了点击响应区域的映射

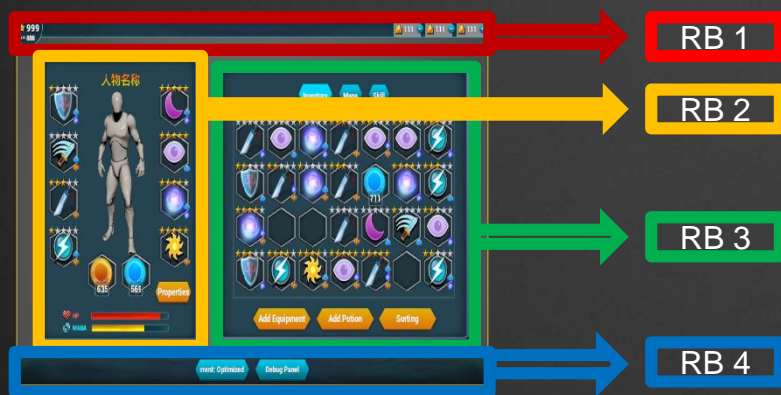


# Retainer Box



# Retainer Box















- Widget Render : 将UI渲染到Render Target
- Slate Render: 使用缓存的Render Target渲染Back Buffer
- 每隔3帧一个循环进行Retainer Box的更新
- 将1帧的UI渲染工作量分配到3帧去处理




Frame 1	RB 1	RB 4
Frame 2	RB 2	
Frame 3	RB 3	
Frame 4	RB 1	RB 4
Frame 5	RB 2	
Frame 6	RB 3	



# 性能对比

	FPS	38.098
	Slate Tick	1.338 ms
	Slate Render	7.706 ms
	Widget Render	0 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
		

	FPS	48.955
	Slate Tick	1.392 ms
	Slate Render	1.494 ms
	Widget Render	3.164 ms
	Invalidation Box	
	Retainer Box	
	Turn Off Effects	
	Event Retainer	
	Texture Atlas	
	C++ Binding	
	Recycle Memory	
	Stat Slate	
	Dump Frames	
		

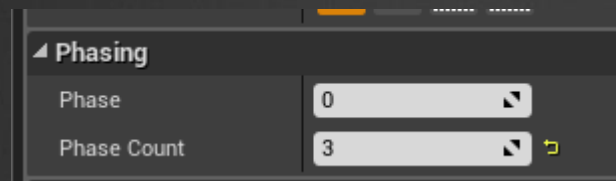
- 关闭Retainer Box: 7.7ms + 0ms
- 开启Retainer Box: 1.5ms + 3.2ms
- FPS提升 : 38 --> 48

# Retainer Box：注意事项

- 会占用额外的显存，因此建议仅在主界面上使用
- Retainer Box区域尽量小，提高渲染效率、降低显存使用
- Retainer Box会为每个User Widget实例创建一个Render Target, 因此重复使用的User Widget不要使用Retainer Box
- 游戏线程的Tick也会相应的隔几帧执行一次
- 持续表示的效果（如3D 角色、材质特效）可以从Retainer Box中分离出来，但需要注意像素填充率；也可以从特效设计的方面解决
- Invalidation Box放置在Retainer Box上方没有意义；推荐一个Retainer Box下跟一个Invalidation Box的方式
- Retainer Box可以上材质效果

# Retainer Box:注意事项

- 每隔3帧更新一次Retainer Box A，在第0帧更新



- 每隔5帧更新一次Retainer Box B，在第2帧更新



- 每隔15帧这两个Retainer Box就会同时更新，导致帧数下降比较多
- Phase Count的设置要全局考虑，避免重叠而导致帧数不稳定



# 事件驱动的Retainer Box

- 事件触发才进行Render Target的更新
- 更换装备或UI效果变化

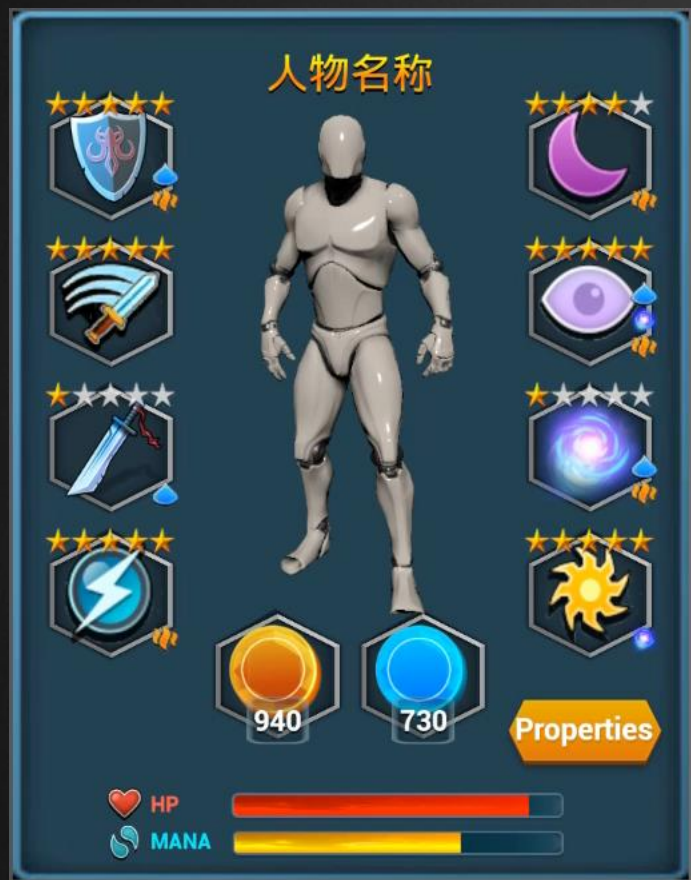
# 事件驱动的Retainer Box

FPS	48.955
Slate Tick	1.392 ms
Slate Render	1.494 ms
Widget Render	3.164 ms
✓ Invalidation Box	
✓ Retainer Box	
✗ Turn Off Effects	
✗ Event Retainer	
✗ Texture Atlas	
✗ C++ Binding	
✗ Recycle Memory	
✗ Stat Slate	
✗ Dump Frames	
Close	

FPS	58.149
Slate Tick	0.796 ms
Slate Render	1.576 ms
Widget Render	0 ms
✓ Invalidation Box	
✓ Retainer Box	
✗ Turn Off Effects	
✓ Event Retainer	
✗ Texture Atlas	
✗ C++ Binding	
✗ Recycle Memory	
✗ Stat Slate	
✗ Dump Frames	
Close	

- 在不需要频繁交互的UI上使用
- 不适用于带特效的UI，在低端机上可以考虑关闭UI特效后开启这个功能
- 对于一些控件需要自定义实现效果
- 实际应用中Widget Render: 0 ~ 3.164

# 切换材质



兼顾高端机效果和低端机性能

DYNAMIC\_MULTICAST



# 目录

## 案例介绍

- 演示工程
- 性能指标
- 优化效果

## 游戏线程优化

- Invalidation Box
- Visibility设置
- Widget Binding
- 蓝图 → C++

## 渲染线程优化

- Retainer Box
- 事件驱动的 Retainer Box
- 合并批次
- 合并贴图
- 切换材质

## 编程技巧

- C++ 开发
- Widget Manager
- 释放贴图内存
- 3D RTT



# C++开发

- 运行效率比蓝图高
- 更灵活，很多C++接口并未开放成蓝图接口
- 除了UI动画，其它代码都能用C++实现
- UPROPERTY ( meta = BindWidget )

```
public:
    UPROPERTY(meta = (BindWidget))
    UTextBlock* OpNameText;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* EquipButton;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* DropButton;

    UPROPERTY(meta = (BindWidget))
    UCBUTTONBLUEUSERWIDGET* CloseButton;
```



# Widget Manager

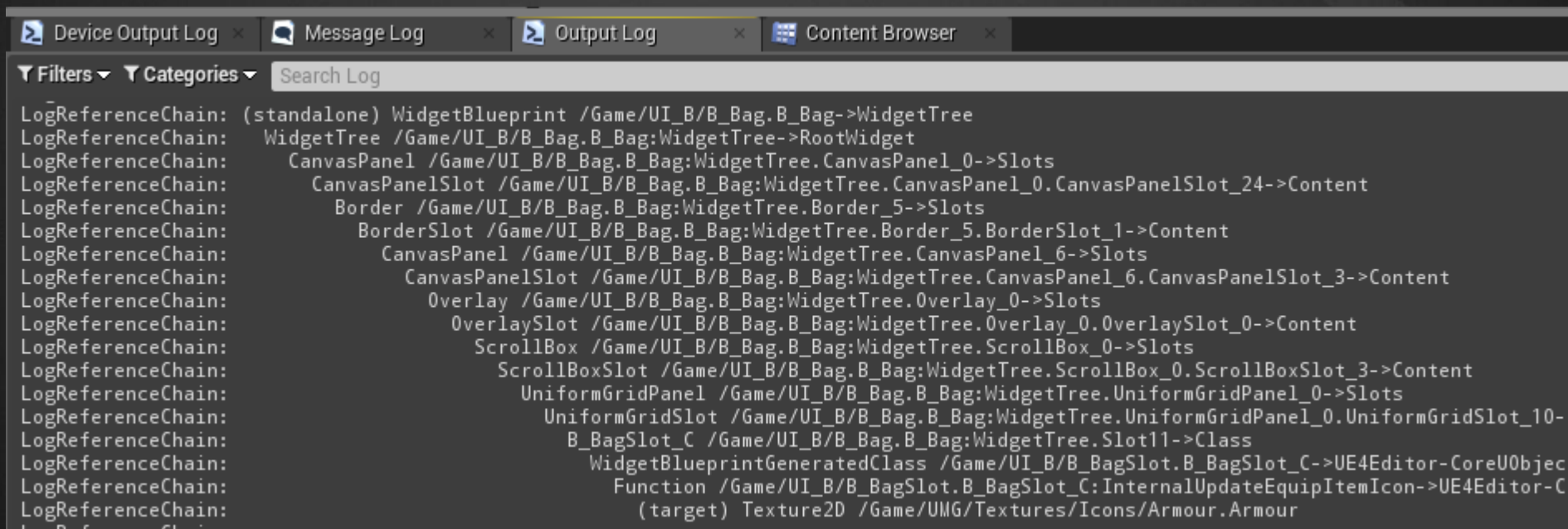
- 使用Widget Manager全局管理所有User Widget
- Brush、Font等资源也可以在Widget Manager中统一管理

# 释放贴图内存

- 某些UI的贴图较大，可以在关闭UI后，释放对应贴图
- 将UI贴图控件自定义成弱引用
- 对于绑定C++的Widget，需要首先RemoveFromViewport

# 释放贴图内存

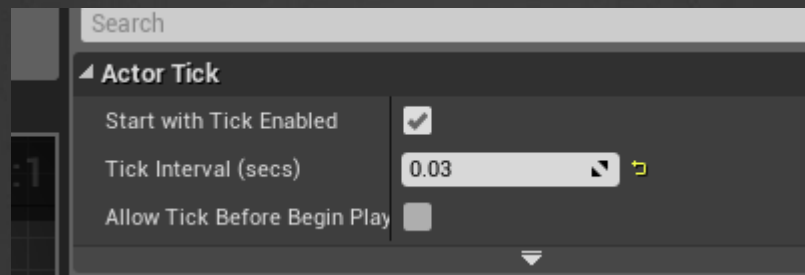
- GC时回收内存
- Obj Refs Name=Armour





# 3D RTT

- 不需要每帧Tick，和动画频率大致同步即可

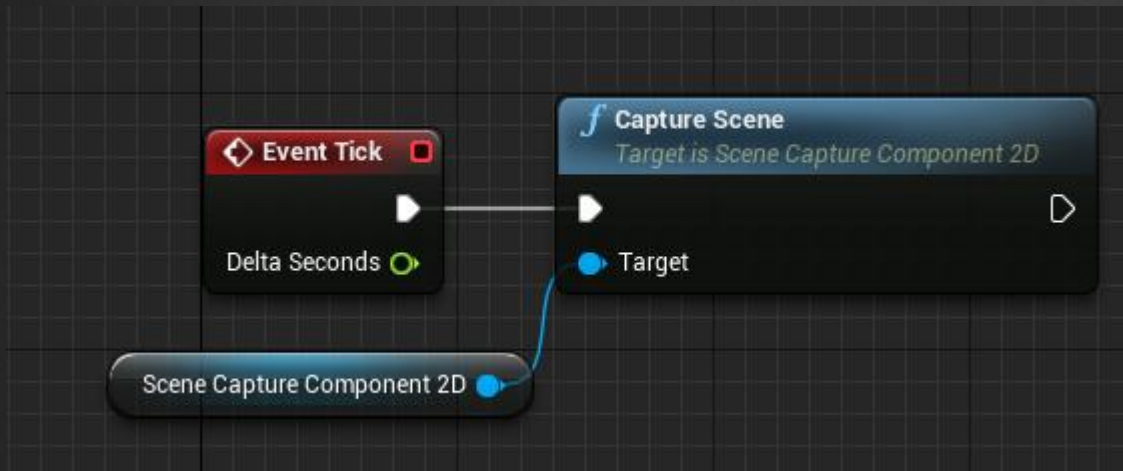


- SceneCaptureComponent2D关闭每帧渲染



# 3D RTT

- 在蓝图中手动调用



- Render Target的尺寸不要太大，会影响显存和渲染效率

# Q & A

