bilibili

让游戏更美好

优化，无处不在

小鱼头

游戏事业部-技术中台-TA组

TIME_2021-07

资产管理

让每一份资产，物超所值

减少包体、内存、加载、编译...

渲染管线

让游戏运行时，心向所往

控制好CPU，安排好GPU

Shader库

让效果品质，如你所愿

不仅“美”，还要“好”

工业化

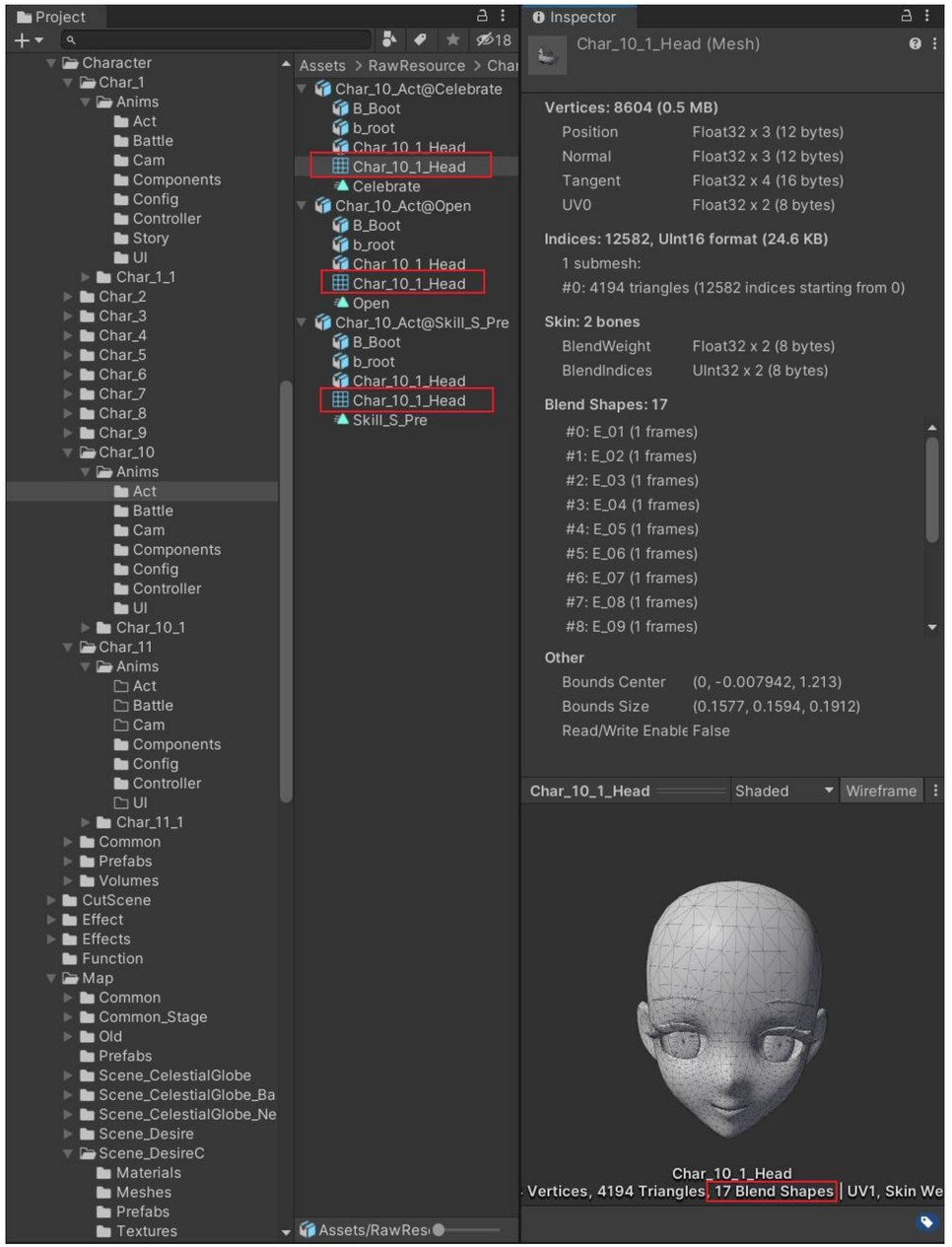
让开发团队，得心应手

提高效率、降低成本与风险

目录 CONTENTS

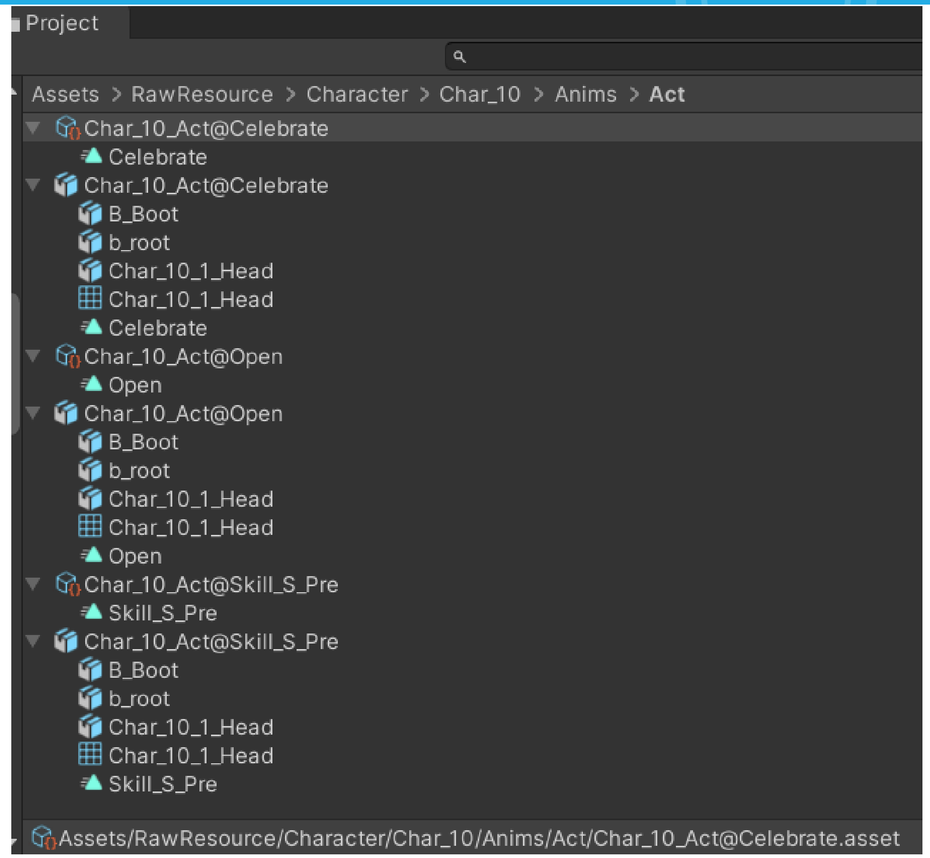
现有动画文件的问题

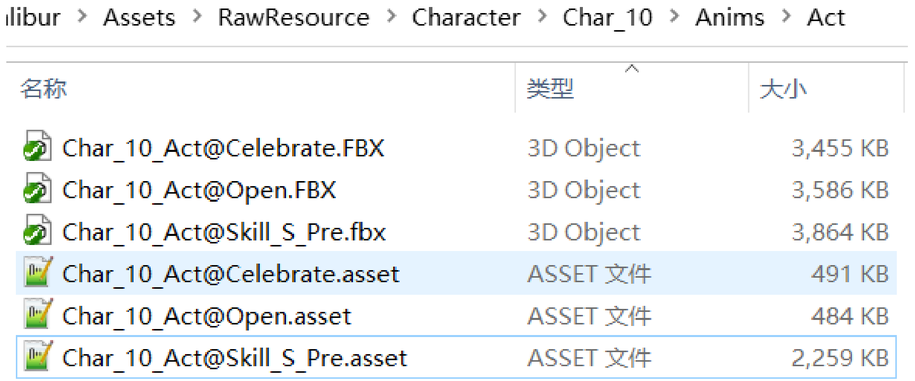
- FBX文件，不可修改
- 对带有Blend Shape内容造成数据冗余
- 二次编辑后，anim文件的Editor数据膨胀
- 不可单独控制anim文本文件转为二进制，如果用工程设置，会对全局影响
- 文本文件会造成Editor开发环境动画加载过慢，影响开发效率
- 精度浪费、部分属性固定，可裁切优化



自定义动画文件特性

- 二进制存储
- 子文件组织结构，暴露引擎可识别的动画对象，用于引用、制作Timeline等
- 不影响工程其他文件的序列化方式
- 提高了Editor开发环境动画加载
- 数据无冗余
  - ✓ 除去ShapeBlend相关Mesh数据
  - ✓ 除去Editor相关数据
  - ✓ 除去部分属性固定的数据
  - ✓ 优化精度





自定义资产的核心代码

Attribute

```
namespace Excalibur
{
  /// <summary>
  /// AnimationClip的扩展类，为了保存成二进制文件,AnimationClip文件作为它的子资源
  /// </summary>
  [PreferBinarySerialization]
  public class AnimationClipEx
        : ScriptableObject
  {
        …
  }
}
```

```
Prefer ScriptableObject derived type to use binary serialization regardless of project's asset serialization mode.
public sealed class
PreferBinarySerialization : Attribute
```

序列化

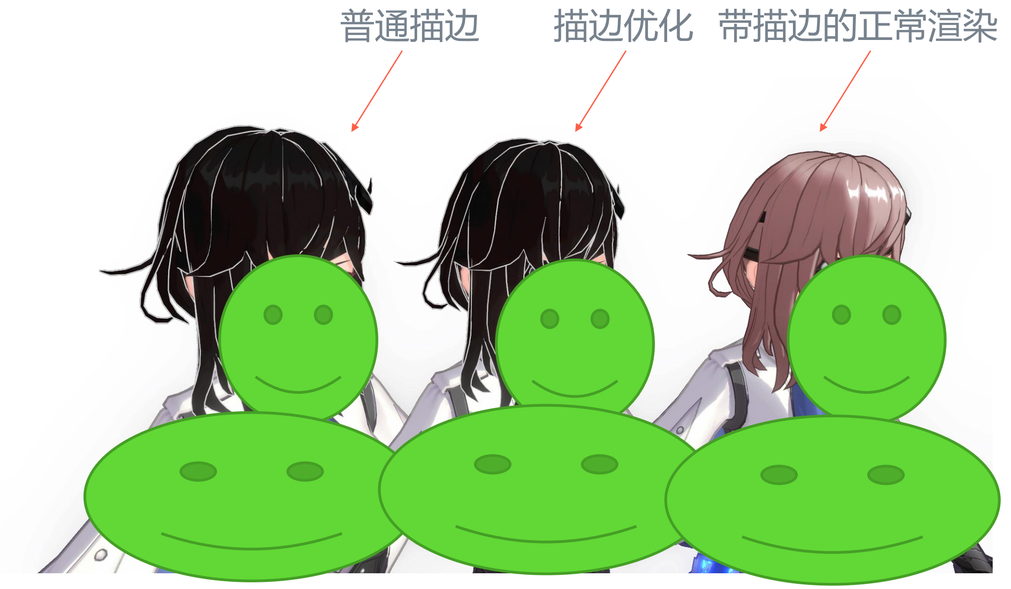
```
List<AnimationClip> addClips = new List<AnimationClip>();
foreach (var obj in objs)
{
   if (obj is AnimationClip fbxClip)
   {
      if (obj.name.Contains(INVALID_ANIM_NAME))
         continue;
      AnimationClip clip = new AnimationClip();
      EditorUtility.CopySerialized(fbxClip, clip);
      if(opt) clip = optmizeAnimationScaleCurve(clip);
      clip = optmizeAnimationFloat(clip)
      clip.hideFlags = HideFlags.NotEditable;
      addClips.Add(clip);
   }
}
AnimationClipEx ex = CreateInstance<AnimationClipEx>();
AssetDatabase.CreateAsset(ex, filePath);
foreach (var addClip in addClips)
   AssetDatabase.AddObjectToAsset(addClip, ex);
AssetDatabase.SetMainObject(ex, filePath);
```

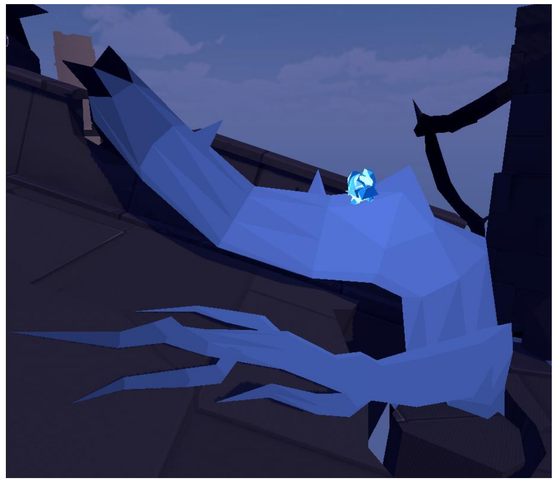
解决异常

```
//为了防止因运行游戏导致Asset资源被修改，产生Git版本控制易用性变差，因此主动还原Animator执行操作，增加genericBindings数据
var go = new GameObject("animator_temp");
var animator =
go.AddComponent<Animator>();
var controller = new AnimatorController();
controller.AddLayer("base");
foreach (var obj in objs)
{
   if (obj is AnimationClip fbxClip)
   {
      //省略 创建 clip 过程
      controller.AddMotion(clip);
   }
}
//省略 创建 ex 过程
animator.runtimeAnimatorController =
controller;
DestroyImmediate(go);
```

为什么要自定义网格文件

- FBX文件，不可修改（但有解决方案）
- 不可单独控制mesh文本文件转为二进制，如果用工程设置，会对全局影响
- 文本文件会造成Editor开发环境动画加载过慢，影响开发效率
- 不需要完整数据，可裁切优化
- 生成低精度Mesh
- **为了效果**
  - ✓ 更好的描边效果
  - ✓ 实现藤蔓色块效果
  - ✓ 实现网格线效果

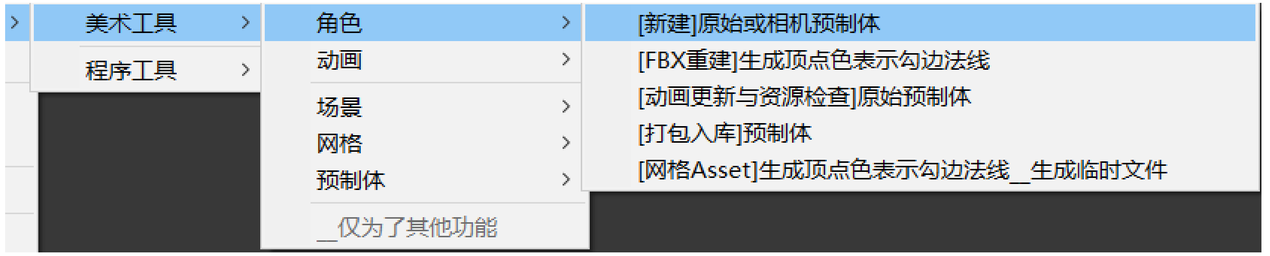




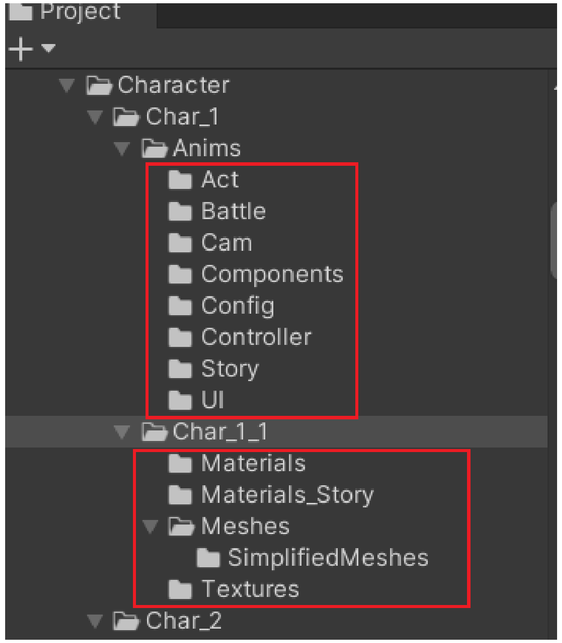
自动生成美术和入库预制体

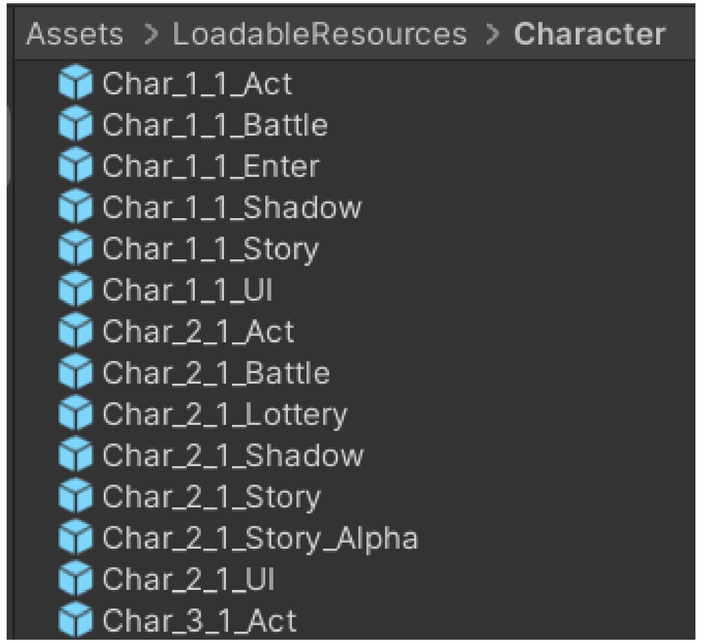
1. 资产放在规定位置，并遵循命名规范
2. 新建美术预制体，自动关联网格、材质、贴图、动画（通过配表）
3. 控制台会输出所有不规范和错误日志
4. 重新生成FBX，以支持更好的描边效果
5. 打包入库，拼接运行时的组件

◆ 可快速创建同一角色的多种预制体
◆ 方便拆解成多个不同质量等级的模板
◆ 可批量调整规则，快速迭代性能优化方案



	A	B	C	D
1	动画名	文件名称	描述	动画循环
2	CombatIdle	Battle@CombatIdle		1
3	CombatRun	Battle@CombatRun	布料需要左右的摆动	1
4	CombatRun_End	Battle@CombatRun_End		0
5	Attack	Battle@Attack	如果是连击做在一起，后腰可以做些表演回IdlePose	0
6	Hurt	Battle@Hurt	前方受击	0
7	FlyHit	Battle@FlyHit	最后pose与StandUp连接	1



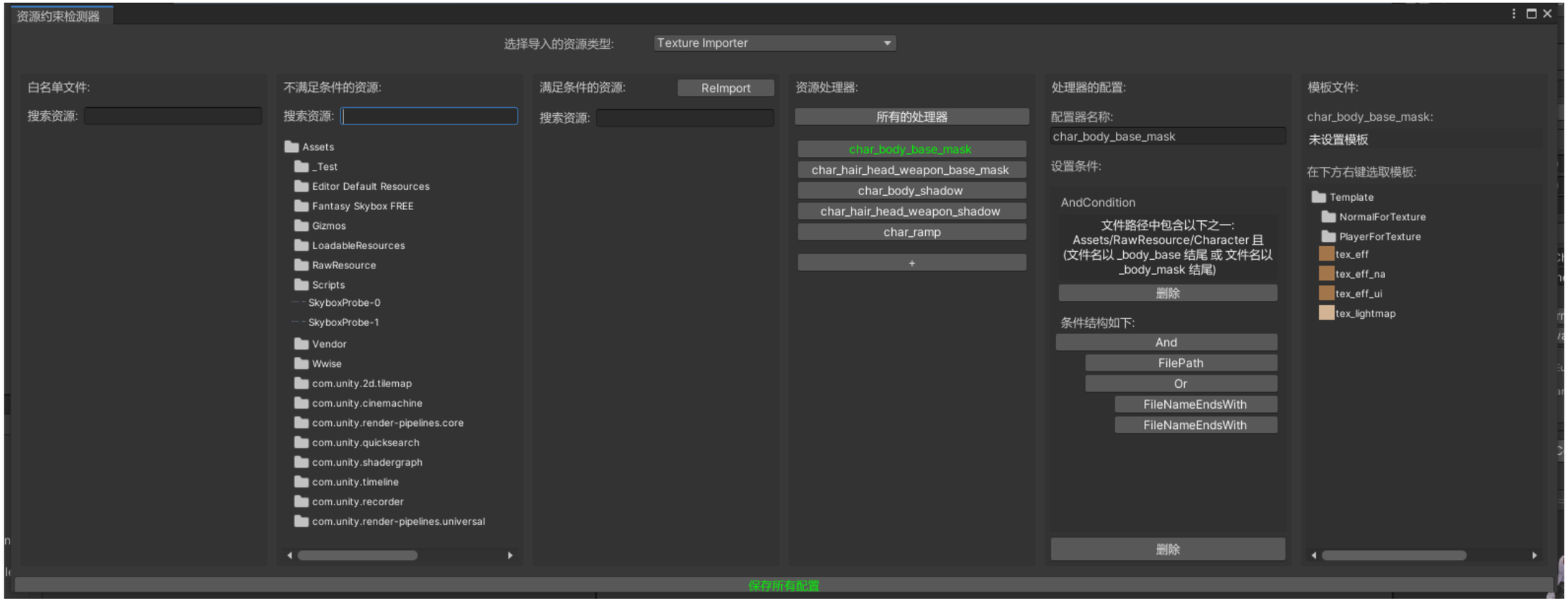


资产约束工具

用来约束美术资产的规格、命名、参数设置

工具运行模式：主动式（资源导入时），被动式（定期或手动检测），命令行式（脚本调用，可集成在CI/CD）

方便批量调整资产，灵活修改约束条件，不限游戏类型和项目规模大小



渲染管线

让游戏运行时，心向所往
控制好CPU，安排好GPU

资产管理

让每一份资产，物超所值
减少包体、内存、加载、编译...

Shader库

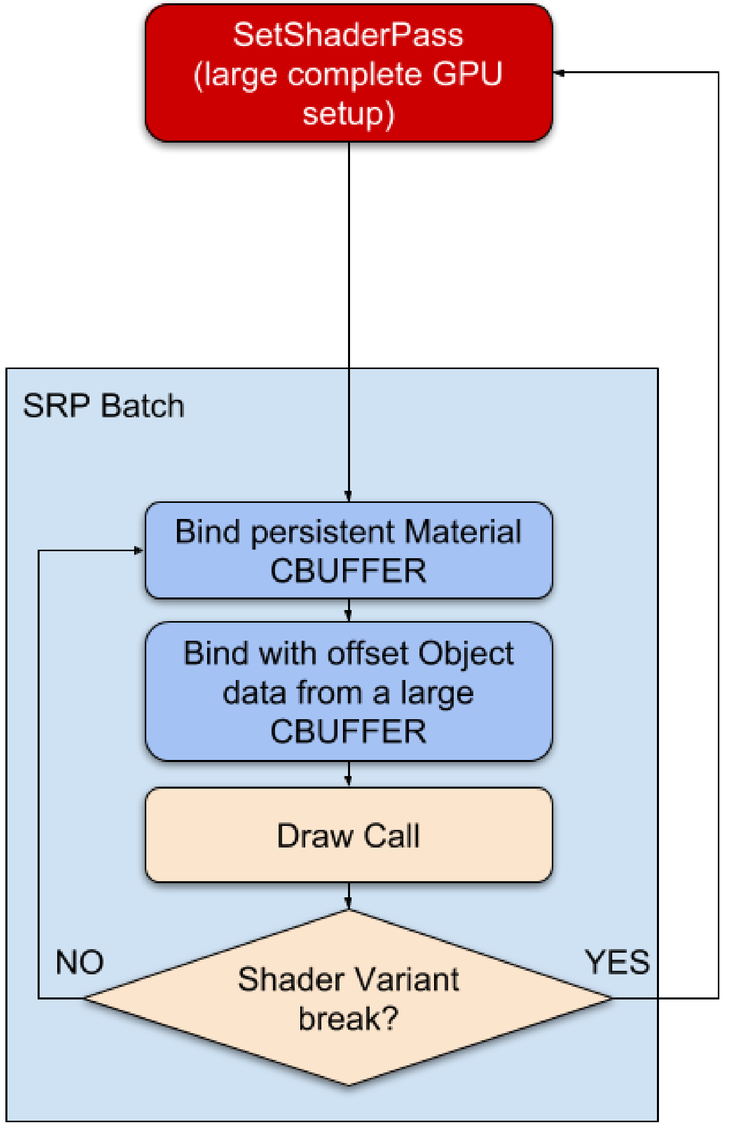
让效果品质，如你所愿
不仅“美”，还要“好”

工业化

让开发团队，得心应手
提高效率、降低成本与风险

SRP Batcher的优势

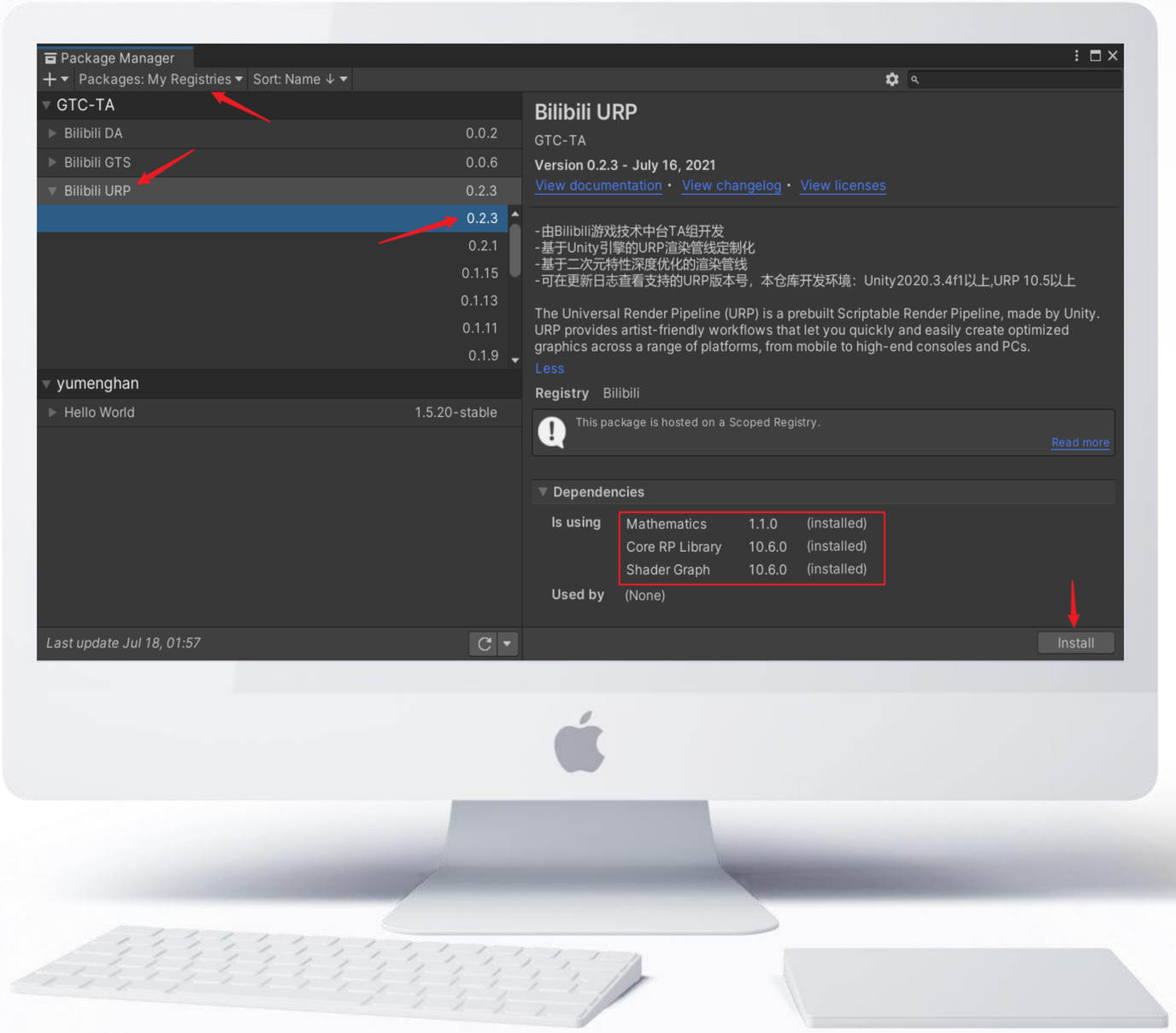
- 减少的不再是DrawCall数量，而是GPU Setup次数
- 合批级别不再是Material，而是Shader变体
- 尽量减少场景上Shader变体数量
- 粒子系统不能SRP Batcher
- 少用半透材质
- 以增加材质属性的方式以减少变体数量



配合Unity2020.3.14f1一起服用，效果更好
SRP Batcher实现思路可参考官方文档：https://docs.unity3d.com/Manual/SRPBatcher.html

BURP的优势

- 保持URP基本逻辑和功能
- 基于二次元特性深度优化
- 可集成更高版本的特性
- 根据项目业务需求定制
- 新增模块均以开关控制
- 项目间优化方案共享
- 开放式合作开发



注：BURP目前仅支持Unity2020.3.4f1以上的版本
访问私有注册表服务器可参考GTC游戏基础平台：https://info.bilibili.co/pages/viewpage.action?pageId=213458755

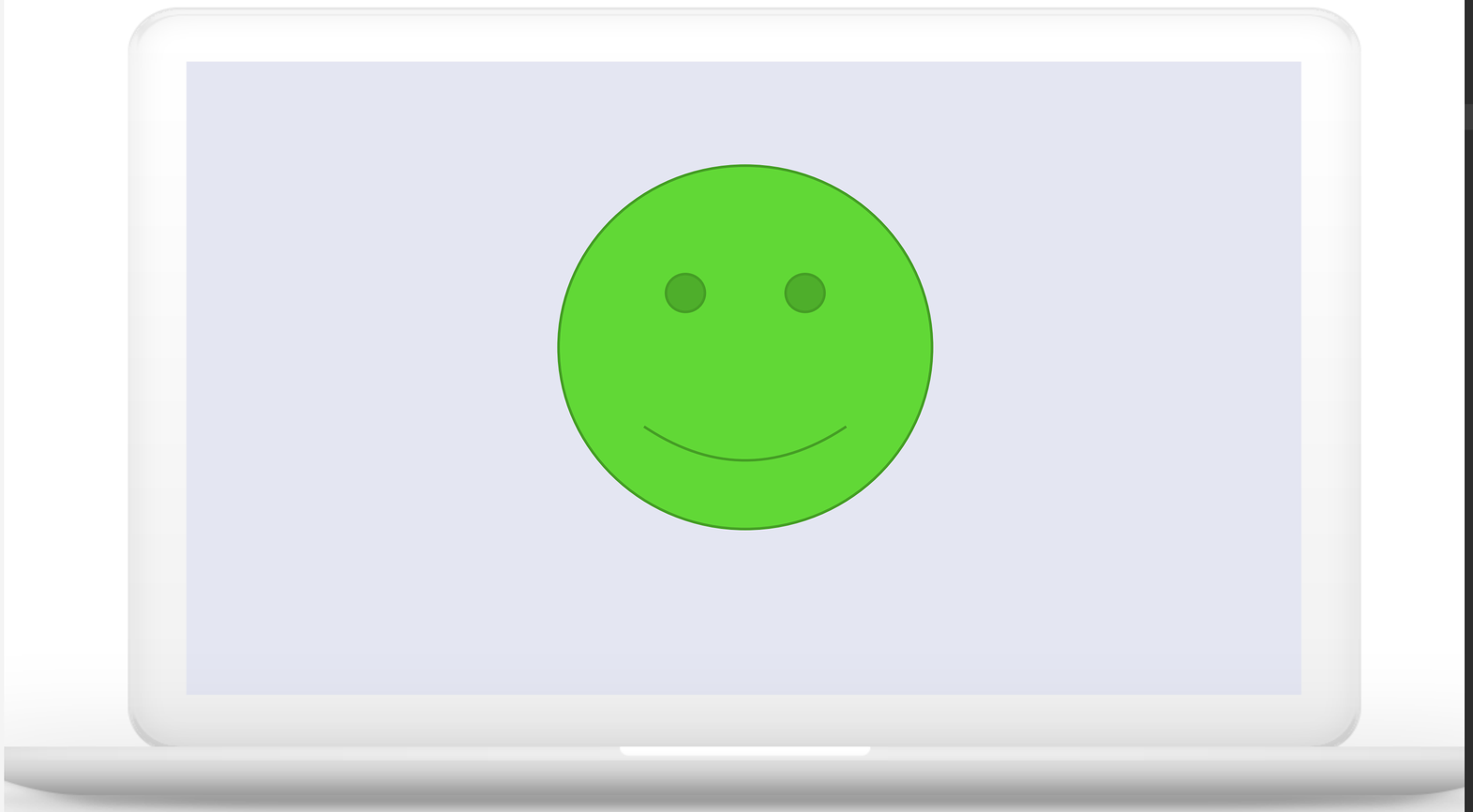
BURP已优化特性

- 点光源衰减兼容性
- 新增Rendering Layer Mask Filter
- 新增轮廓光
- 优化后期处理效果
- 优化Android与Editor点光源衰减不一致
- 优化RT+后期处理未保留Alpha数据
- 优化Overlay相机无法开启Depth、Opaque贴图

注：Rendering Layer Mask属于Renderer组件属性

目录

BURP 小案例



Frame Debug

Disable | Editor

- UniversalRenderPipeline.RenderSingleCamera: Phantom 45
  - ScriptableRenderer.Execute: ForwardRenderer_LobbyChar 45
    - ScriptableRenderPass.Configure 1
    - MainLightShadow 6
      - ShadowLoopNewBatcher.Draw 6
        - SRP Batch
        - SRP Batch
        - SRP Batch
        - SRP Batch
        - SRP Batch
        - SRP Batch
    - ScriptableRenderPass.Configure 2
    - DepthNormalPrepass 3
      - RenderLoopNewBatcher.Draw 3
        - SRP Batch
        - SRP Batch
        - SRP Batch
    - ColorGradingLUT 2
    - SSRimLight 1
      - Draw Mesh
    - ScriptableRenderPass.Configure 1
    - DrawOpaqueObjects 4
      - RenderLoopNewBatcher.Draw 4
        - SRP Batch
        - SRP Batch
        - SRP Batch
        - SRP Batch
    - CharOutlineRenderObjects 3
      - RenderLoopNewBatcher.Draw 3
        - SRP Batch
        - SRP Batch
        - SRP Batch
    - CopyColor 2
    - DrawTransparentObjects 2
      - RenderLoopNewBatcher.Draw 2
        - SRP Batch
        - SRP Batch
    - Render PostProcessing Effects 18
      - UberPostProcess 18
        - Bloom 17
        - Draw Mesh
- UniversalRenderPipeline.RenderSingleCamera: UICamera 56
  - ScriptableRenderer.Execute: ForwardRenderer_UI 56

资产管理

让每一份资产，物超所值

减少包体、内存、加载、编译...

渲染管线

让游戏运行时，心向所往

控制好CPU，安排好GPU

Shader库

让效果品质，如你所愿

不仅“美”，还要“好”

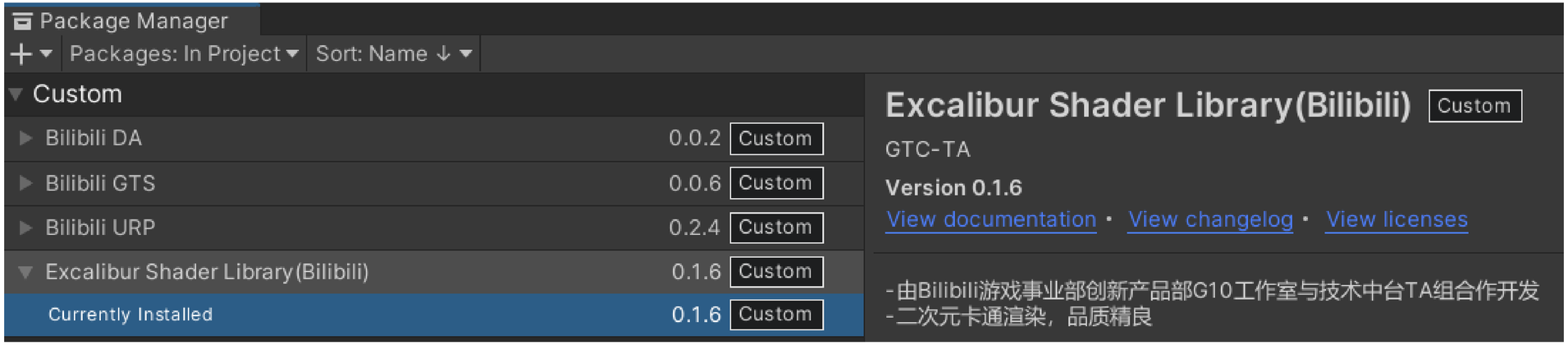
工业化

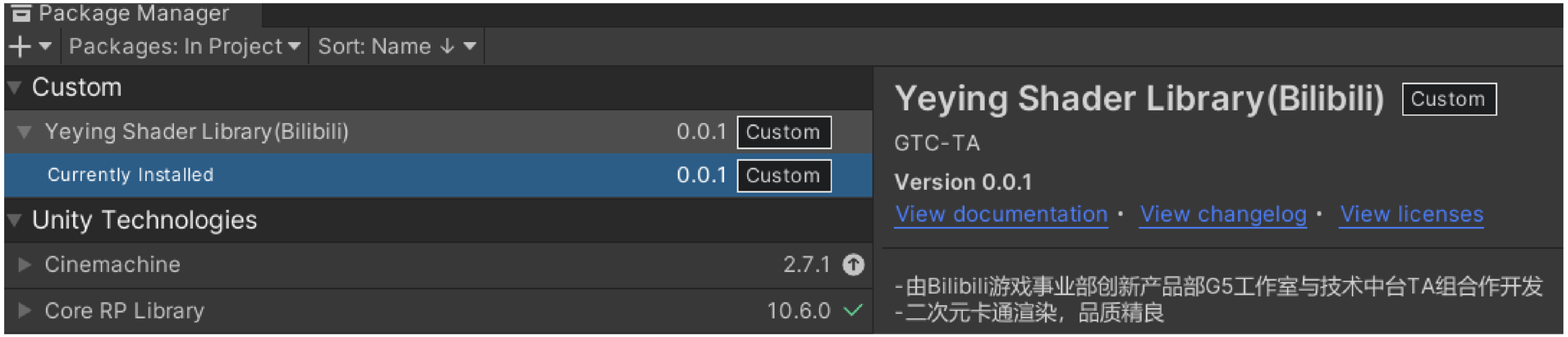
让开发团队，得心应手

提高效率、降低成本与风险

二次元Shader源码“托管”库

- 与游戏业务解耦，可快速迭代，易维护，易验证
- 独属于项目与技术中台，非经项目允许杜绝外传，确保资产的安全性
- 高效支持SRP Batcher，Shader源码级性能优化
- 结合BURP实现更优解





Shader结构

- URP通用渲染流程使用Pass有：实时投射阴影、深度图或深度法线图、基本渲染
- 其他Pass需要渲染管线配合调用
- 不需要烘焙，无Meta Pass
- 不需要2d渲染，无Universal2D
- 不需要延迟渲染，无GBuffer
- 为所有Pass，包含同一份Input文件，抛开变体隔离，列出所有材质属性在 UnityPerMaterial 的CBUFFER中

RolePBR.shader

```
1   //角色卡通渲染
2   Shader "Excalibur/BURP/Char/Role_PBR"
3   {
4       Properties
5       {...}
146
147     SubShader
148     {
149         Tags {"RenderType"="Opaque" "IgnoreProjector" = "True" "Ren
150         LOD 100
151
152         // 基本渲染
153         pass
154         {...}
185
186         // 描边
187         pass
188         {...}
213
214         // 平面阴影
215         pass
216         {...}
246
247         // 深度图
248         Pass
249         {...}
271
272         // 深度法线图
273         Pass
274         {...}
299
300         // 实时投射阴影
301         Pass
302         {...}
323     }
324     CustomEditor "Excalibur.ShaderLib.ShaderGUI.RolePBRShaderGUI"
325 }
```

减少变体

- 尽量减少不必要的变体。比如：

  _ADDITIONAL_LIGHT_SHADOWS
  _SHADOWS_SOFT
  DIRLIGHTMAP_COMBINED
  _SCREEN_SPACE_OCCLUSION

- multi_compile_fog 可替换成 _ FOG_LINEAR
- 材质相差过大，可拆分Shader，降低复杂度
- 材质属性开关代替变体
- 变体组合减少2^n
- 变体由脚本控制，对运行时合批几乎不影响

```
// 基本渲染
pass
{
    Name "BURP_CHAR_PBR"
    Tags {"LightMode"="UniversalForward"}

    Blend [_SrcBlend][_DstBlend]
    ZWrite [_ZWrite] // 无论是不透或半透渲染，均打开写深度，以避免特效可能会把角色遮挡
    Cull [_Cull]

    HLSLPROGRAM
    #pragma only_renderers gles3 glcore d3d11 metal vulkan
    #pragma target 3.0

    #pragma vertex BasicVert
    #pragma fragment BasicFrag

    // DETAIL_SWITCH:有无细节效果，区分于战斗之外的高品质效果
    #pragma multi_compile __ DETAIL_SWITCH

    // 默认多个变体不会同时出现
    // DITHER_SWITCH:有无抖动效果
    // DISSOLVE_SWITCH:有无溶解效果
    // FLASH_SWITCH: 有无闪光
    #pragma multi_compile __ DISSOLVE_SWITCH DITHER_SWITCH FLASH_SWITCH

    #pragma multi_compile _ _SCREEN_SPACE_RIM_LIGHT

    #include "RolePBRInput.hlsl"
    #include "RolePBRInc.hlsl"

    ENDHLSL
}
```

编码优化Tips

- 拒绝条件语句，用Lerp代替
- Half精度明确足够的，不要用float
- 数据类型不同不要轻易赋值
- Pow(x,y)函数确保x>0，或x=0,y>0
- Varyings降低不必要的变量，减少寄存器数量
- PBR渲染可借鉴标准Shader写法
- 变体声明，尽量用_local，全局变体数量有限
- 明确不会被脚本控制的，尽量用_feature
- 明确常驻变体，可以通过 #define声明

```
// 基本渲染
pass
{
    Name "BURP_CHAR_PBR"
    Tags {"LightMode"="UniversalForward"}

    Blend [_SrcBlend][_DstBlend]
    ZWrite [_ZWrite] // 无论是不透或半透渲染，均打开写深度，以避免特效可能会把角色遮挡
    Cull [_Cull]

    HLSLPROGRAM
    #pragma only_renderers gles3 glcore d3d11 metal vulkan
    #pragma target 3.0

    #pragma vertex BasicVert
    #pragma fragment BasicFrag

    // DETAIL_SWITCH:有无细节效果，区分于战斗之外的高品质效果
    #pragma multi_compile __ DETAIL_SWITCH

    // 默认多个变体不会同时出现
    // DITHER_SWITCH:有无抖动效果
    // DISSOLVE_SWITCH:有无溶解效果
    // FLASH_SWITCH: 有无闪光
    #pragma multi_compile __ DISSOLVE_SWITCH DITHER_SWITCH FLASH_SWITCH

    #pragma multi_compile _ _SCREEN_SPACE_RIM_LIGHT

    #include "RolePBRInput.hlsl"
    #include "RolePBRInc.hlsl"

    ENDHLSL
}
```

SRP Batcher性能对比

SRP关闭

Statistics
Audio:
Level: -∞ dB (MUTED) DSP load: 0.0%
Clipping: 0.0% Stream load: 0.0%
Graphics: 66.2 FPS (15.1ms)
CPU: main 15.1ms render thread 8.0ms
Batches: 685 Saved by batching: 858
Tris: 1.4M Verts: 1.6M
Screen: 1280x720 - 10.5 MB
SetPass calls: 515 Shadow casters: 46
Visible skinned meshes: 22
Animation components playing: 0
Animator components playing: 18

Frame Debug
Disable Editor
UniversalRenderPipeline.RenderSingleCamera: Camera 709
ScriptableRenderer.Execute: ForwardRenderer_BattleS 709
ScriptableRenderPass.Configure 1
MainLightShadow 72
ScriptableRenderPass.Configure 1
DepthNormalPrepass 248
ColorGradingLUT 1
SSRimLight 1
ScriptableRenderPass.Configure 1
DrawOpaqueObjects 290
RenderLoop.Draw 290
CharOutlineRenderObjects 20
CharPlanarShadowRenderObjects 20
Camera.RenderSkybox 1
CopyColor 2
DrawTransparentObjects 32
Render PostProcessing Effects 19
UniversalRenderPipeline.RenderSingleCamera: UICamera 14
ScriptableRenderer.Execute: ForwardRenderer_UI 14

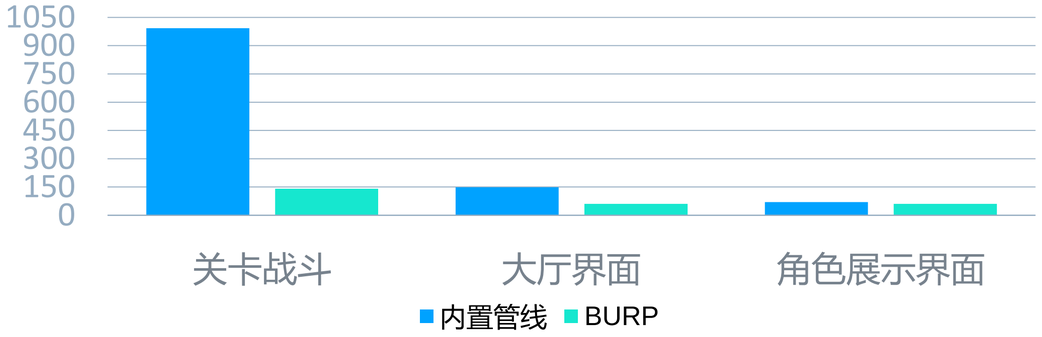
SRP开启

Statistics
Audio:
Level: -∞ dB (MUTED) DSP load: 0.0%
Clipping: 0.0% Stream load: 0.0%
Graphics: 76.7 FPS (13.0ms)
CPU: main 13.0ms render thread 6.3ms
Batches: 1510 Saved by batching: 6
Tris: 1.4M Verts: 1.6M
Screen: 1280x720 - 10.5 MB
SetPass calls: 225 Shadow casters: 76
Visible skinned meshes: 22
Animation components playing: 0
Animator components playing: 18

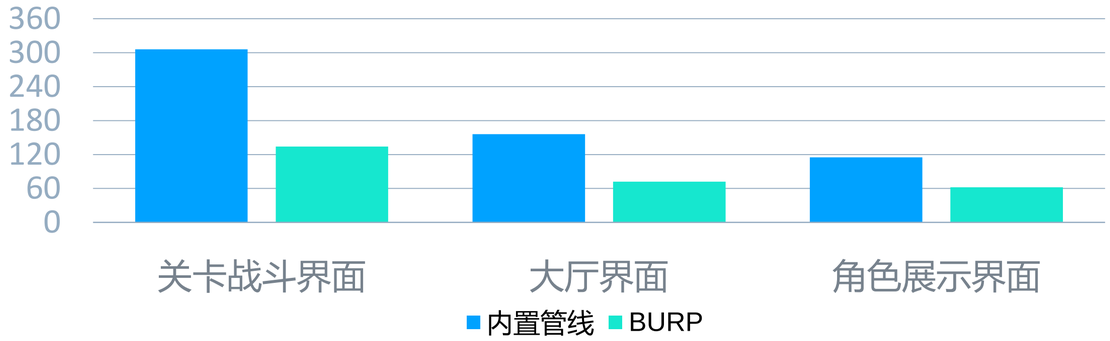
Frame Debug
Disable Editor
UniversalRenderPipeline.RenderSingleCamera: Camera 226
ScriptableRenderer.Execute: ForwardRenderer_BattleS 226
ScriptableRenderPass.Configure 1
MainLightShadow 23
ScriptableRenderPass.Configure 1
DepthNormalPrepass 70
ColorGradingLUT 1
SSRimLight 1
ScriptableRenderPass.Configure 1
DrawOpaqueObjects 77
CharOutlineRenderObjects 4
CharPlanarShadowRenderObjects 4
Camera.RenderSkybox 1
CopyColor 2
DrawTransparentObjects 21
Render PostProcessing Effects 19
UniversalRenderPipeline.RenderSingleCamera: UICamera 14
ScriptableRenderer.Execute: ForwardRenderer_UI 14

对比Unity2019内置渲染管线

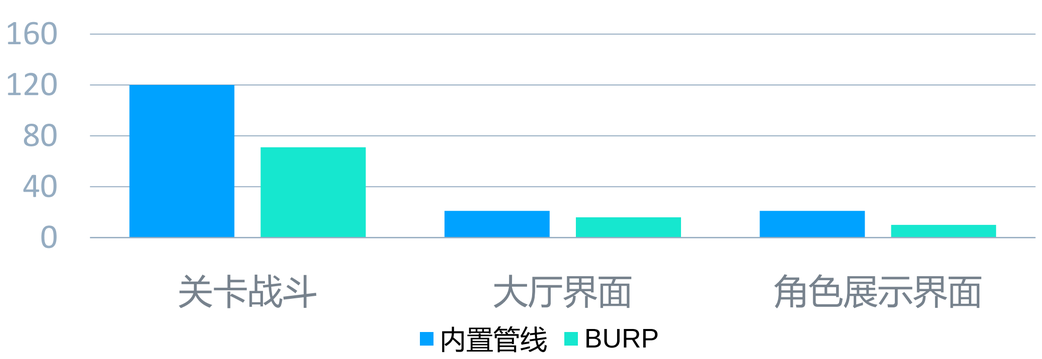
Editor下Set Pass对比(单位：次)



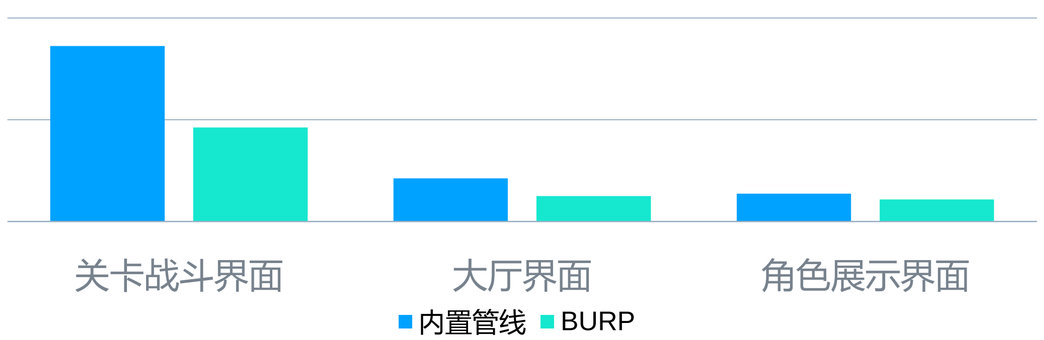
Android设备下Set Pass对比(单位：次)



Editor下渲染面数对比(单位：k)



Android设备下渲染面数对比(单位：k)



资产管理

让每一份资产，物超所值

减少包体、内存、加载、编译...

渲染管线

让游戏运行时，心向所往

控制好CPU，安排好GPU

Shader库

让效果品质，如你所愿

不仅“美”，还要“好”

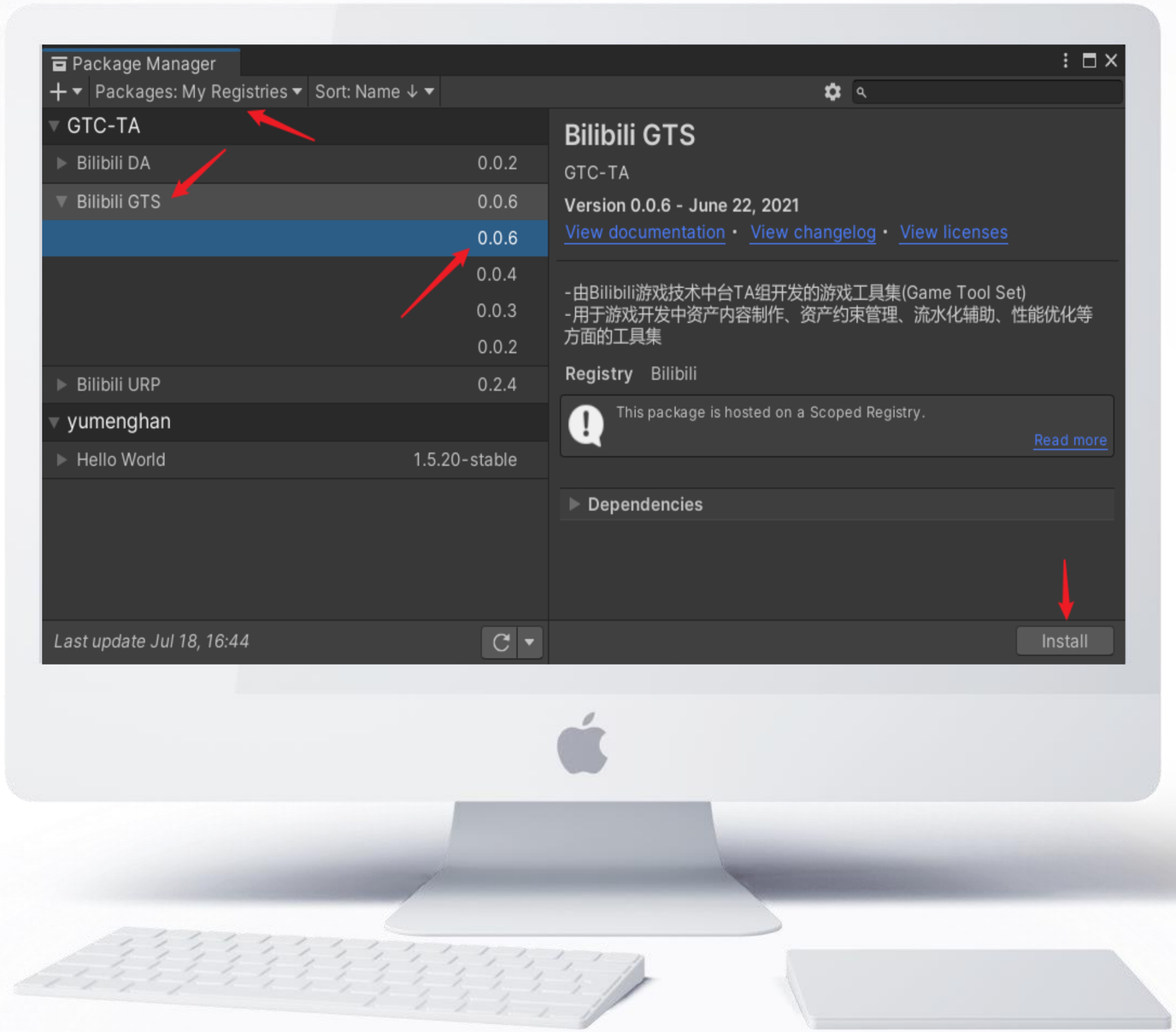
工业化

让开发团队，得心应手

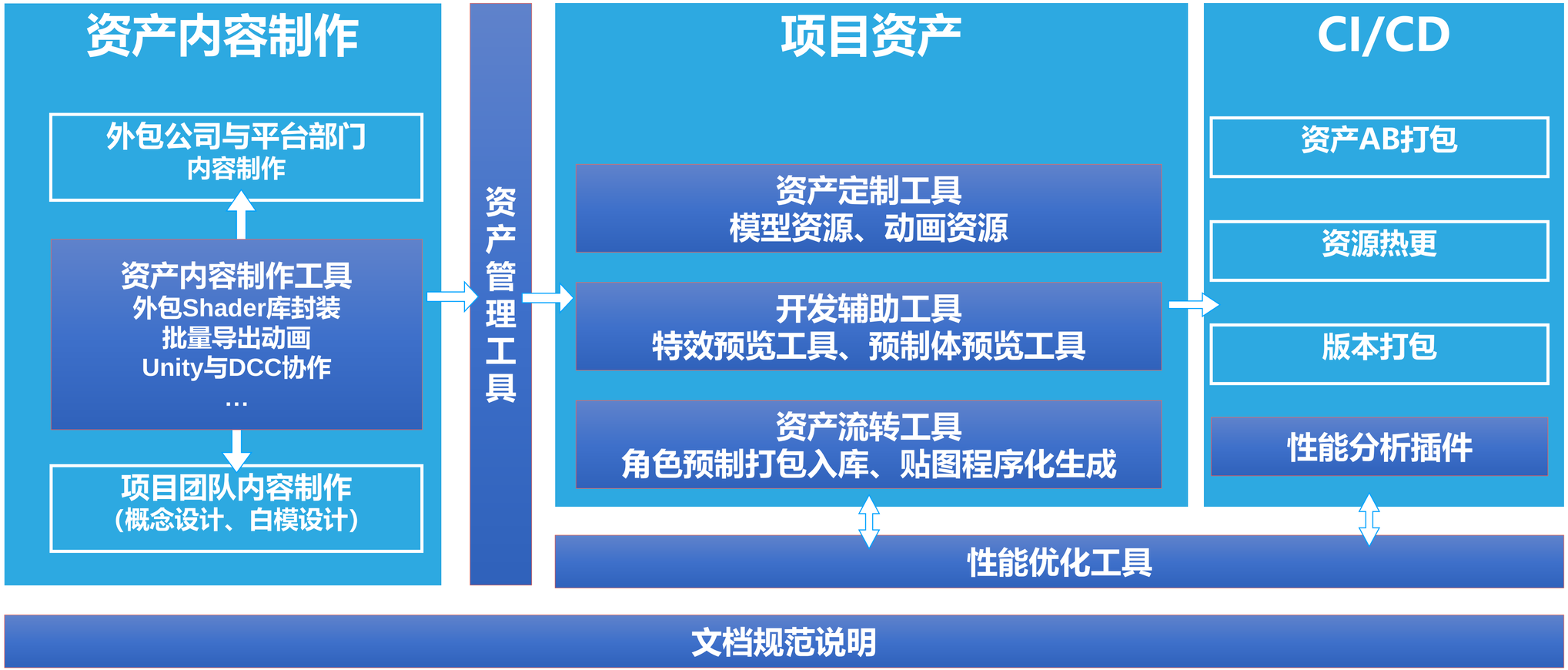
提高效率、降低成本与风险

工具的重要性

- 减少无聊冗长的重复劳动
- 增强协作的效率和安全性
- 保证开发环境和编译版本的强健性
- 维护游戏品质的一致性
- 减少新人的出错率和学习成本

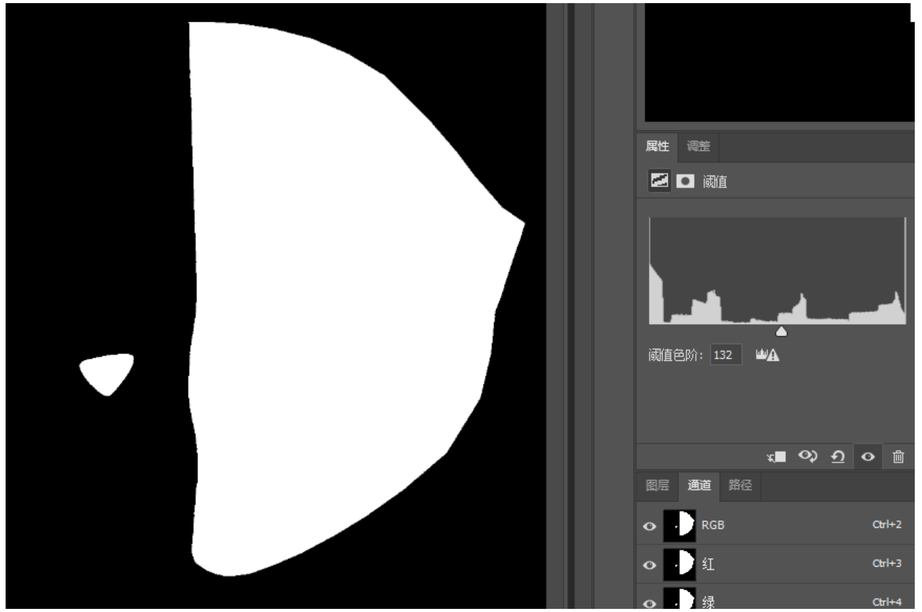


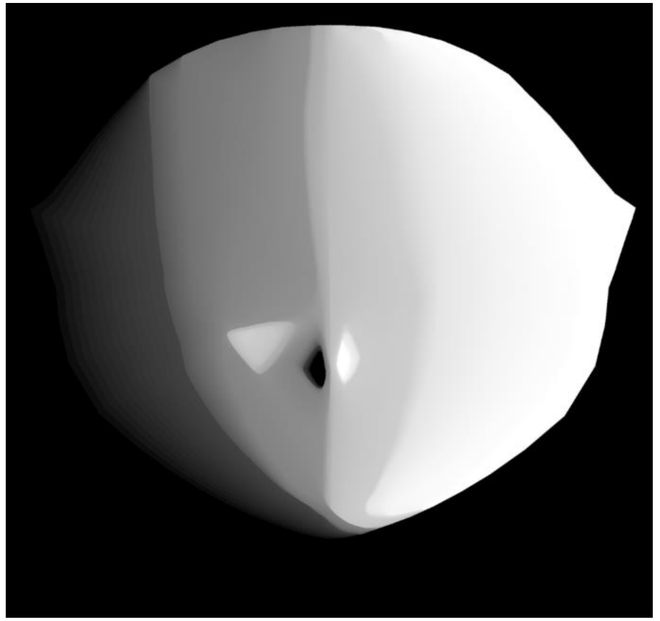
工业化流程



角色脸部阴影阈值图

- 自定义脸部阴影过渡效果
- 可通过遮罩图屏蔽眼睛及其他非过渡区域
- 美术提供几张特定角色的阴影图和遮罩图
- 简单设置，一键生成
- 支持高斯模糊处理

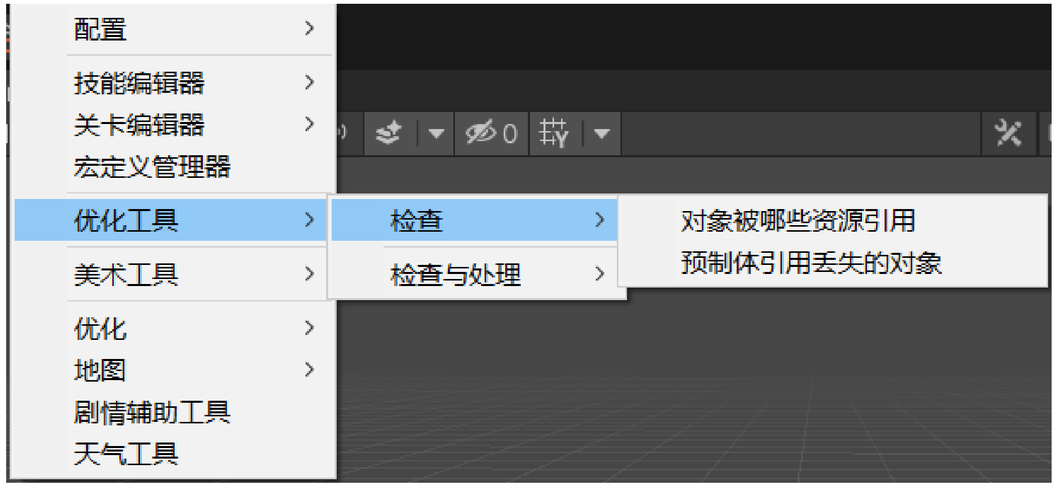




目录

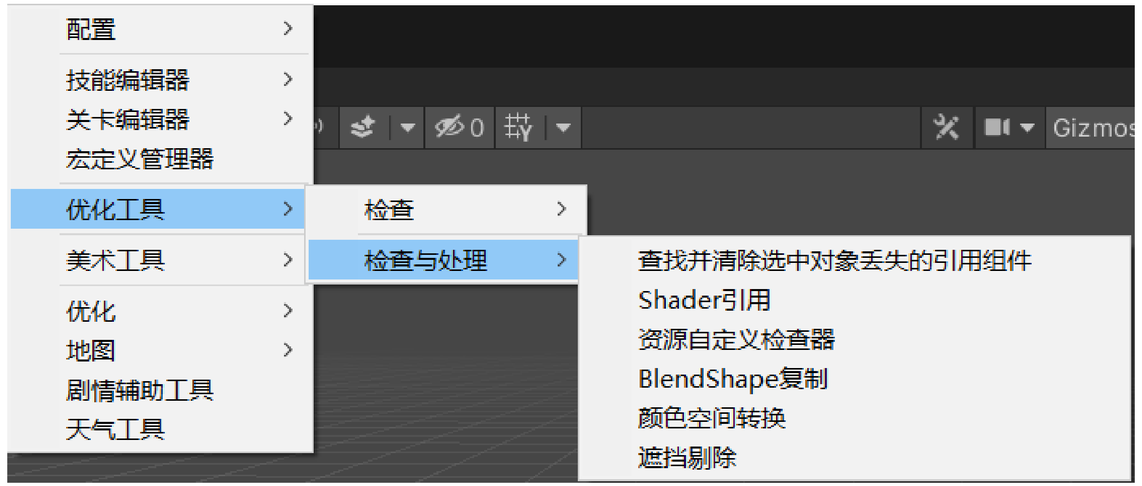
材质ID通道合并工具

- 由两张贴图八个通道合并为一个通道
- 支持Mipmap生成
- 材质ID可减少自发光贴图
- 材质ID可控制二维Ramp贴图的渐变采样区域
- 材质ID可细分控制Bloom强度



性能优化辅助工具

- 由两张贴图八个通道合并为一个通道
- 支持Mipmap生成
- 材质ID可减少自发光贴图
- 材质ID可控制二维Ramp贴图的渐变采样区域
- 材质ID可细分控制Bloom强度



彩蛋

资产管理

让每一份资产，物超所值

减少包体、内存、加载、编译...

渲染

让游戏运行

控制好CPU

...er库

...质，如你所愿

...还要“好”

工业化

让开发团队，得心应手

提高效率、降低成本与风险

1 2 3 4

场景优化

- LOD 和 自定义LOD 组件
- 遮挡剔除(OC)
- 减化模型，网格精简工具（Simple Mesh Combine、Simple LOD），远景转面片(Imposter)
- 网格合并(Mesh Baker) 按簇归类
- 动态加载碰撞触发器，按区域加载卸载
- GPU Instance
- Terrain->Mesh, E3DMeshPainter
- 大场景 HLOD + Streaming Octahedral Imposters

角色优化

- LOD 战斗与展示不同的模型精度，不同的Shader复杂度，不同的贴图质量
- 网格减化，用于特殊效果，比如残影、半隐
- 半透与不透效果Mesh拆分，或SubMesh
- 投射阴影可通过Shader渲染平面阴影
- 材质捕获（MatCap）代替高光计算
- 脸部、皮肤、头发不走PBR渲染
- 尝试base,normal,mask贴图尺寸依次降级
- ASTC格式，尝试从6*6，8*8，甚至12*12

后期处理优化

- 减少相机使用，相机CPU开销较高
- 多个相机后期处理避免重复，尝试通过调整材质属性融合
- Shader Pass整合，只是单像素调色可以整合到ColorGradingLUT，或单独Pass渲染出一张混合图，在最后Uber/Final Pass整合，减少Blit计算
- Bloom降采样从较低的分辨率开始
- 通过质量分级策略降低后期对中低端设备性能的巨大开销
- 景深、模糊、扭曲等开销大户不要常态开启

优化策略

/// 其他优化

UI优化

- 质量分级策略，控制游戏分辨率，推荐高中低为：1080p,720p,480p
- 可酌情考虑UI使用更高分辨率，以实现更好的交互体验
- Canvas分组策略：按层划分，动静分离
- UI贴图关闭Mipmap

特效优化

- 质量分级策略，控制好粒子数量
- 自定义特效LOD
- 减少屏占比和Overdraw，尤其是半透材质
- 简单特效可考虑由Shader替换
- 减少使用物理碰撞、粒子光源

实时阴影优化

- 质量分级策略，控制阴影贴图大小、级联等级，附加光源数量及是否投射阴影
- 光照烘焙模式若选择ShadowMask，可通过Frame Debug验证场景实时阴影的正确性

Thanks

这里是个广告位

欢迎技术合作

开放式合作开发BURP：https://git.bilibili.co/gtc/ta/burp
开放式合作开发BGTS：https://git.bilibili.co/gtc/ta/bgts
Unity包私有注册表服务器：http://packages-gtc.bilibiligame.net