

TossMaster

基于 Flutter 框架、OpenGL ES 3.0 渲染的 3D 跨平台 AR 投掷游戏

浙江大学 2024 学年秋冬学期《计算机图形学》课程项目展示

一、项目内容及 Demo 展示

1.1 核心玩法：投掷游戏

长按蓄力 → 投掷 → 命中

物理运动，包括重力、碰撞检测



1.2 项目亮点：移动端跨平台实现

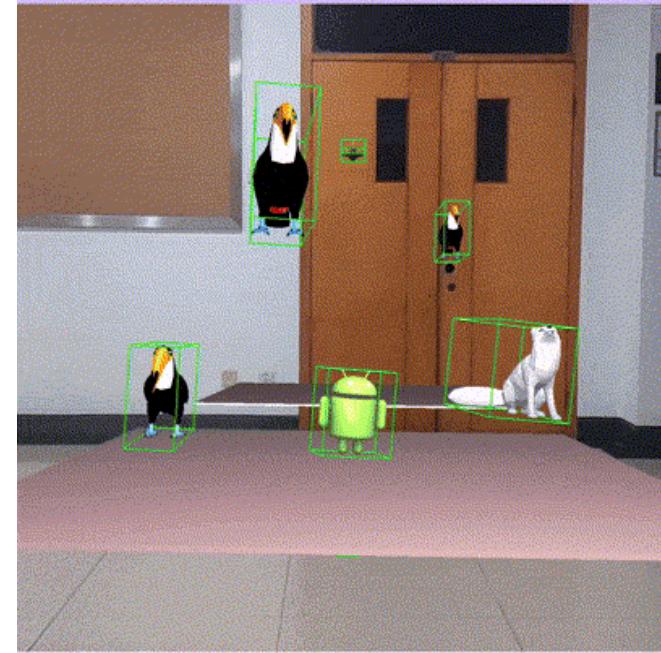
借助先进的 Flutter 框架，在 Android、HarmonyOS 和 iOS 上完成适配。

跨了，但没完全跨，还是得做适配。

本项目构思时的目标就是高级要求中的两条内容：

- (8 分) 不依赖现有引擎，采用 iOS/Android 平台实现。
- (7 分) 与增强现实应用结合。





1.2 项目亮点：AR

场景中的物体能够跟随相机视角，营造出虚拟与现实共享同一空间的体验。

1.3 基本要求：OBJ 模型及其纹理导入

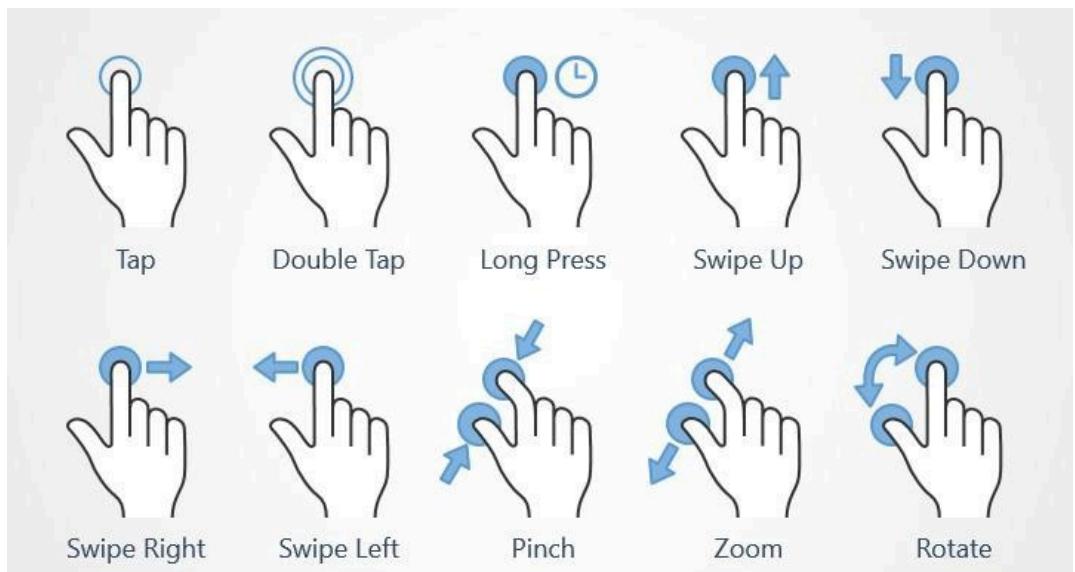
- 用户导入模型，存储在模型库中。
- 导入模型同时可以选择导入纹理和展示用的 GIF。
- 用户选择模型库中的模型，将其放置在场景中。
- 好处：**实例化渲染**，一个 OpenGL 调用渲染多个实例，节约了移动端的内存带宽。



1.4 基本要求：几何变换

用户与画面交互以控制模型：

- 单击选中
- 拖动平移
- 双指缩放
- 双指旋转

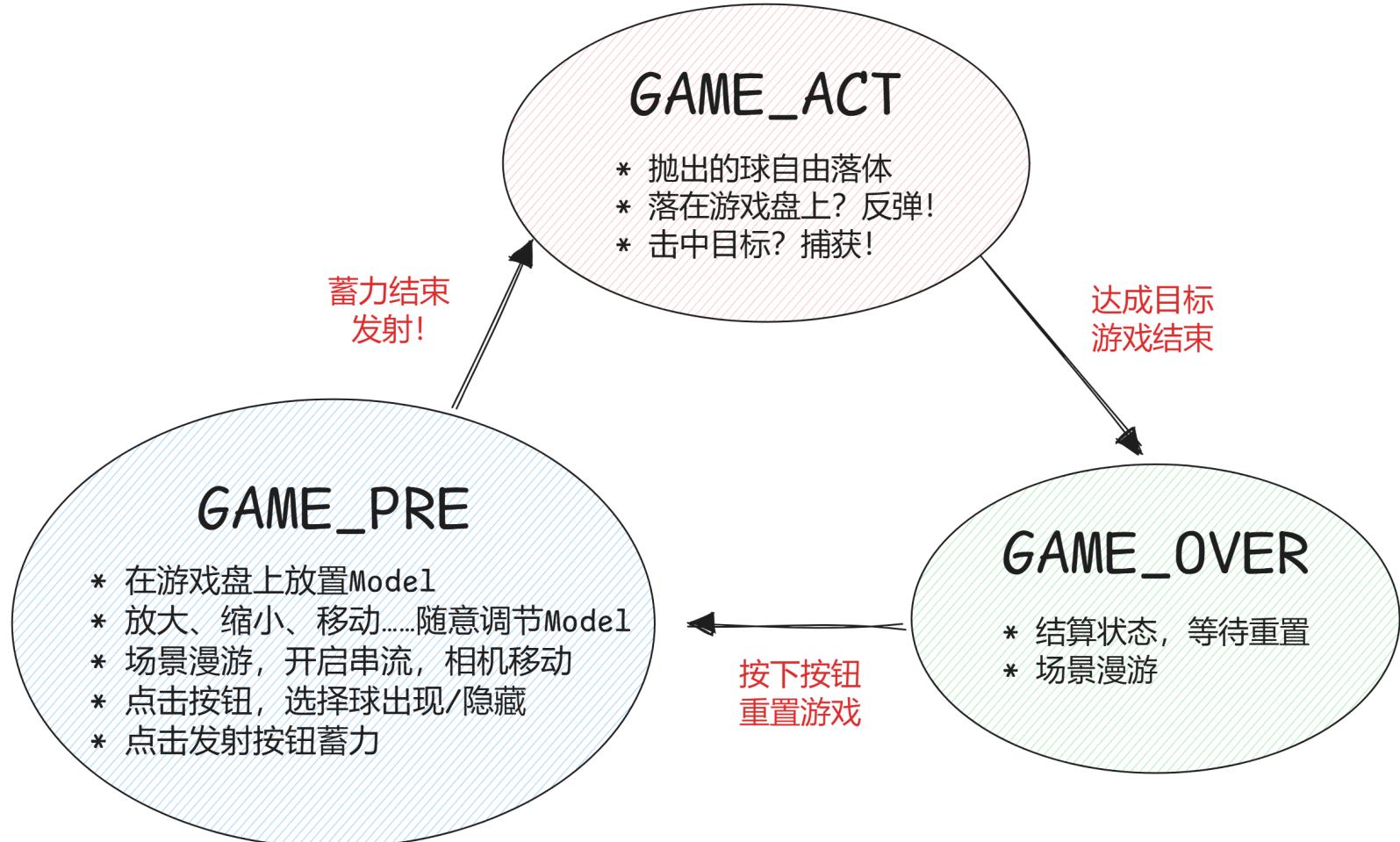


1.5 基本要求：Blinn-Phong 着色的 ADS 光照模型及材质

- Blinn-Phong 着色在 Phong 的基础上节省了大量性能损耗，对移动端意义重大。
- 两个光源，用户可控：
 - 全局光：没有方向，仅有 A（环境光）分量，对每个像素具有相同的光照。
 - 定向光（远距离光）：具有方向和 A（环境光）、D（漫反射光）、S（镜面光）三个反射分量。
- 材质：ADS + 光泽，预置金、银、铜、玉、珍珠材质。



1.6 游戏循环



二、心得与体会

充满着荆棘与坎坷的移动端开发之路

~~强烈不建议无移动端开发经验的同学尝试在跨平台框架上做 OpenGL 开发~~



2.0 捉襟见肘的移动端资源：功耗、带宽与 TBR



可怜的 GPU 尺寸和功耗需求：我打 PC 端？尊嚟假嚟 o_O

- 移动端：分块渲染（TBR, Tile-Based Rendering），将帧缓冲分割为一小块一小块，然后逐块进行渲染。
- 桌面端：即时渲染（IMR, Immediate Mode Rendering），一次性渲染整个帧缓冲，需要大量的带宽。

2.1 缺少基础设施的移动端

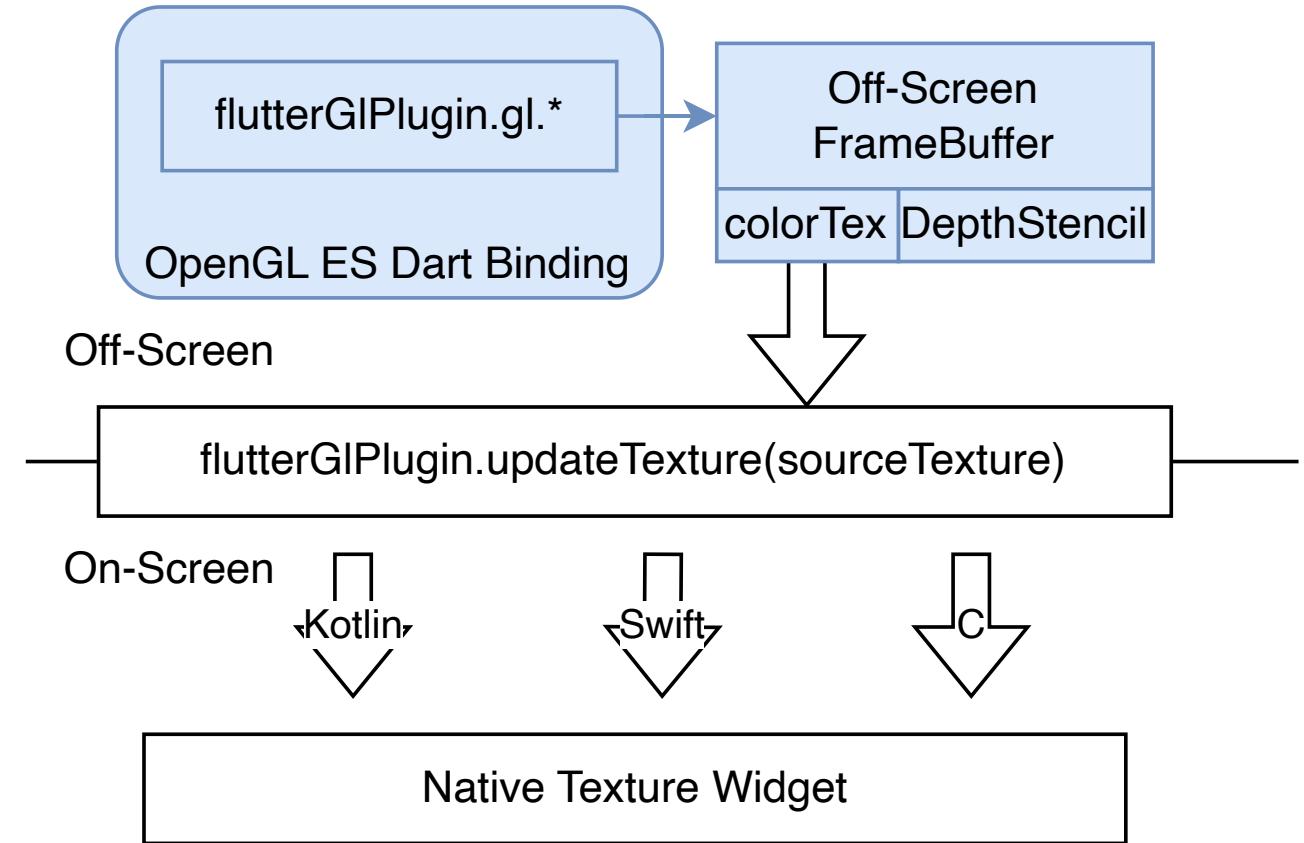
极少有人在如 Flutter 等跨平台框架中直接使用 OpenGL 这类底层库进行开发。

项目	状态
google/dart-gl Dart 原生 GLES2 扩展	停止维护 2022 年
alnitak/flutter_opengl GLSL 玩具罢了 😢 (ShaderToy.com)	上次更新 2022 年
wasabia/flutter_gl 通过 <code>dart:ffi</code> 绑定到 C 接口	上次更新 2022 年



wasabia/flutter_gl 的绘制方式

- 在安卓端需要修一下依赖，适配到 NDK 34 以上。
- 在 Dart 中离屏渲染到 FrameBuffer
- 将 FBO 的颜色纹理附件传递给 Native Texture Widget



2.2 百花齐放的图像编码

从 `startImageStream((image) async {})` 获得的 `image` 可能为：

- iOS: BGRA8888
- Android: YUV420 (适用于视频流的一种编码，将明度与颜色分开存储，在低带宽时能够只显示黑白画面)

然而 OpenGL `glTexImage2D` 只支持 RGB、RGBA 等格式。

format

Specifies the format of the pixel data. The following symbolic values are accepted: `GL_RED`, `GL_RED_INTEGER`, `GL_RG`, `GL_RG_INTEGER`, `GL_RGB`, `GL_RGB_INTEGER`, `GL_RGBA`, `GL_RGBA_INTEGER`, `GL_DEPTH_COMPONENT`, `GL_DEPTH_STENCIL`, `GL_LUMINANCE_ALPHA`, `GL_LUMINANCE`, and `GL_ALPHA`.

Tree	Method Table	CPU Flame Chart
▼ Self Time	Method	
3.53 s (43.31%)	> convertYUV420ToImage - (package:alby-o/image_converter.dart)	
698.94 ms (8.57%)	> PixelUint8.numChannels - (package:alby-o/image_converter.dart)	
495.18 ms (6.07%)	> Image.getPixel - (package:image/src)	
375.86 ms (4.61%)	> ImageDataUint8.setPixelRgb - (package:alby-o/image_converter.dart)	
304.73 ms (3.74%)	> PixelUint8.b= - (package:image/src)	
290.96 ms (3.57%)	> PixelUint8.g= - (package:image/src)	
287.29 ms (3.52%)	> PixelUint8.set - (package:image/src)	
280.40 ms (3.44%)	> ImageDataUint8.getPixel - (package:alby-o/image_converter.dart)	
160.62 ms (1.97%)	> _IntegerImplementation.clamp - (dart:core)	
158.33 ms (1.94%)	> PixelUint8.r - (package:image/src)	
117.94 ms (1.45%)	> ImageDataUint8.rowStride - (package:alby-o/image_converter.dart)	
117.48 ms (1.44%)	> PixelUint8.data - (package:image/src)	
116.11 ms (1.42%)	> PixelUint8.setPosition - (package:alby-o/image_converter.dart)	
105.09 ms (1.29%)	> PixelUint8.a= - (package:image/src)	
93.16 ms (1.14%)	> PixelUint8.moveToNext - (package:alby-o/image_converter.dart)	
79.85 ms (0.98%)	> PixelUint8.r= - (package:alby-o/image_converter.dart)	
69.76 ms (0.86%)	> _IntListMixin.setRange - (dart:typed_list)	
61.95 ms (0.76%)	> copyCrop - (package:image/src)	
48.65 ms (0.60%)	> PixelUint8.g - (package:alby-o/image_converter.dart)	

糟糕的访存模式 (YUV420)

对于转换后 RGBA 图像的每个像素，逐次访问明度和色度平面，并且两个平面的 **Stride** 不同。

让本就不高的带宽雪上加霜



```

for (int h = 0; h < imageHeight; h++) {
    int uvh = (h / 2).floor();
    for (int w = 0; w < imageWidth; w++) {
        int uvw = (w / 2).floor();
        final int yIndex = (h * yRowStride) + (w * yPixelStride);
        final int y = yBuffer[yIndex];
        final int uvIndex = (uvh * uvRowStride) + (uvw * uvPixelStride);
        final int u = uBuffer[uvIndex]; final int v = vBuffer[uvIndex];
        int r = (y + v * 1436 / 1024 - 179).round();
        int g = (y - u * 46549 / 131072 + 44 - v * 93604 / 131072 + 91).round();
        int b = (y + u * 1814 / 1024 - 227).round();
        r = r.clamp(0, 255); g = g.clamp(0, 255); b = b.clamp(0, 255);
        image.setPixelRgb(imageHeight - h - 1, imageWidth - w - 1, r, g, b);
    }
}

```

2.3 Dart 是一门函数式语言

Dart 是一款由 Google 开发的函数式编程语言，
你将在 Flutter 框架中探索无状态和数据的不可
变性.....

$$\text{UI} = f(\text{state})$$

The layout
on the screen

Your
build
methods

The application state

When the state of your app changes (for example, the user flips a switch in the settings screen), you change the state, and that triggers a **redraw of the user interface**.

拒绝重绘！

将所有状态存储在一个 Widget 中，状态变更在 Widget 内部处理。

~~然后代码变成💩山，UI 和程序逻辑混杂在一起，背离函数式编程的初衷。~~

2.4 AR: 如何实现?

- 最初计划：借助 OpenCV 的 ArUco Marker 实现，然而
 - `opencv_dart` 缺少关键的相机姿态估计函数 `solvePnP` 和 `estimatePoseSingleMarkers` 的绑定。
 - OpenCV 相机姿态解析需要先对相机进行大量的标定（Camera Calibration），涉及计算机视觉相关的内容，难以在项目时间内完成。
- 求助 AR 框架：
 - 平台分裂：安卓 ARCore, iOS ARKit
 - `arcore_flutter_plugin` 缺少相机参数接口。
 - `arkit_plugin` 具有接口，但开发人员缺少 iOS 设备，无法测试。
 - `ar_flutter_plugin` 实现了两者的跨平台支持，但年久失修，有严重的依赖问题。
- 手搓 PnP 或 RANSAC 算法？超出课程范围。

换个思路：传感器

移动端设备具有加速度计、陀螺仪，可以感知设备的运动状态。

- 加速度计：离散采样难以获得准确的位移信息。比如使用 $dx = v_x \cdot dt + \frac{1}{2}a_x \cdot dt^2$ 计算，摇一摇直接起飞，平稳地走半天却没有位移变化。
- 陀螺仪：角速度信息 rad/s，可积分得到旋转角度，实测表现良好。旁轴旋转时产生偏移，暂未探究原因。

不同设备的传感器精度和采样率不同，需要进行校准和平滑处理。

2.5 鸿蒙与安卓亦有不同 😞

```
OpenGL Error: 1282  
Error compiling shader:  
S0059: 'binding' qualifier is not allowed in language version 300 es
```

4.3.8.3 Uniform Block Layout

Layout qualifiers can be used for uniform block declarations. The layout qualifier identifiers for uniform blocks are:

layout-qualifier-id
shared
packed
std140
row_major
column_major

4.4.5 Uniform and Shader Storage Block Layout

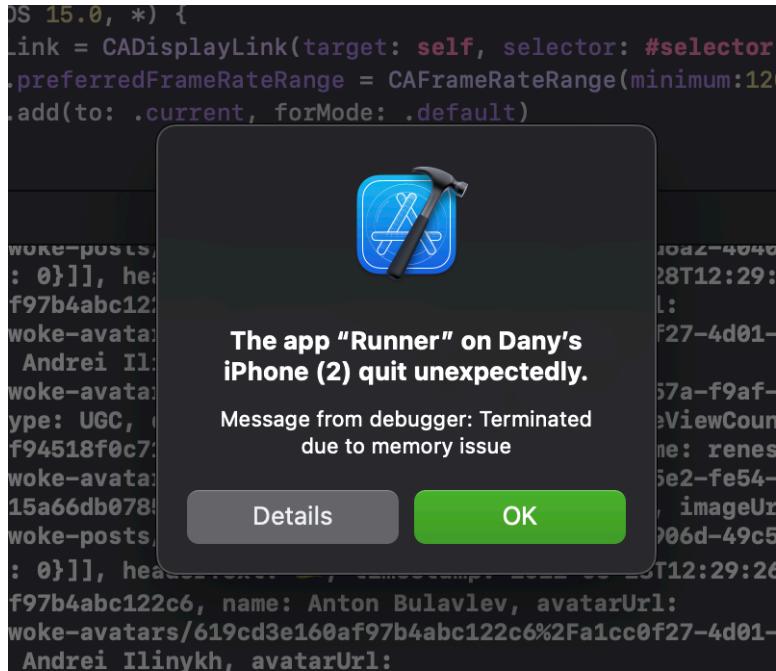
Layout qualifiers can be used for uniform and shader storage block declarations. The layout qualifier identifiers for these blocks are:

layout-qualifier-id
shared
packed
std140
std430
row_major
column_major
binding = integer-constant

2.6 寸土寸金的移动端存储

S0032: no default precision defined for variable 'varyingNormal'

- 着色器中必须指定默认精度，为 `Float16`。
- 帧缓冲区爆内存：善用 `glClear`、
`glInvalidateFramebuffer`。



Welcome Play



TossMaster: 基于 Flutter 框架、OpenGL ES 3.0 渲染的 3D 跨平台 AR 投掷游戏

浙江大学 2024 学年秋冬学期《计算机图形学》课程项目展示