Ijkplayer中opengles的操作

入口：IJK\_EGL\_display

IJK\_EGL\_makeCurrent

Egl操作，得到egl->context egl->surface egl->display

1. IJK\_EGL\_display\_internal

Opengles操作

2.1、IJK\_EGL\_prepareRenderer

2.1.1、IJK\_GLES2\_Renderer\_create

根据overlay->format创建不同的着色器

以IJK\_GLES2\_Renderer\_create\_yuv420p为例

IJK\_GLES2\_Renderer\_create\_base

IJK\_GLES2\_getFragmentShader\_yuv420p

{

(

precision highp float;

varying highp vec2 vv2\_Texcoord;

uniform mat3 um3\_ColorConversion;

uniform lowp sampler2D us2\_SamplerX;

uniform lowp sampler2D us2\_SamplerY;

uniform lowp sampler2D us2\_SamplerZ;

void main()

{

mediump vec3 yuv;

lowp vec3 rgb;

yuv.x = (texture2D(us2\_SamplerX, vv2\_Texcoord).r - (16.0 / 255.0));

yuv.y = (texture2D(us2\_SamplerY, vv2\_Texcoord).r - 0.5);

yuv.z = (texture2D(us2\_SamplerZ, vv2\_Texcoord).r - 0.5);

rgb = um3\_ColorConversion \* yuv;

gl\_FragColor = vec4(rgb, 1);

}

)

}

IJK\_GLES2\_getVertexShader\_default

{ precision highp float;

varying highp vec2 vv2\_Texcoord;

attribute highp vec4 av4\_Position;

attribute highp vec2 av2\_Texcoord;

uniform mat4 um4\_ModelViewProjection;

void main()

{

gl\_Position = um4\_ModelViewProjection \* av4\_Position;

vv2\_Texcoord = av2\_Texcoord.xy;

}

}

renderer->av4\_position= glGetAttribLocation(renderer->program, "av4\_Position");顶点信息。

renderer->av2\_texcoord=glGetAttribLocation(renderer->program, "av2\_Texcoord");裁剪信息，vv2\_Texcoord传给fragment shader计算颜色。

renderer->um4\_mvp=glGetUniformLocation(renderer->program, "um4\_ModelViewProjection");坐标转换矩阵

renderer->us2\_sampler[0]=glGetUniformLocation(renderer->program, "us2\_SamplerX");2d纹理

renderer->us2\_sampler[1]=glGetUniformLocation(renderer->program, "us2\_SamplerY");

renderer->us2\_sampler[2]=glGetUniformLocation(renderer->program, "us2\_SamplerZ");

renderer->um3\_color\_conversion=glGetUniformLocation(renderer->program, "um3\_ColorConversion");颜色空间转换矩阵

2.1.2、IJK\_GLES2\_Renderer\_use

func\_use yuv420p\_use

glGenTextures(3, renderer->plane\_textures)创建3个纹理name，name保存在plane\_textures

glActiveTexture激活一个纹理单元

glBindTexture在当前的纹理单元中创建一个纹理对象

glTexParameteri对当前的纹理对象设置参数

glUniform1i(renderer->us2\_sampler[i], i)赋值us2\_sampler，代表一个纹理单元

glUniformMatrix3fv(renderer->um3\_color\_conversion, 1, GL\_FALSE, IJK\_GLES2\_getColorMatrix\_bt709());赋值颜色空间转换矩阵

glUniformMatrix4fv(renderer->um4\_mvp, 1, GL\_FALSE, modelViewProj.m);赋值坐标变换矩阵

glVertexAttribPointer(renderer->av2\_texcoord, 2, GL\_FLOAT, GL\_FALSE, 0, renderer->texcoords)

glEnableVertexAttribArray(renderer->av2\_texcoord);

赋值renderer->av2\_texcoord

glVertexAttribPointer(renderer->av4\_position, 2, GL\_FLOAT, GL\_FALSE, 0, renderer->vertices);

glEnableVertexAttribArray(renderer->av4\_position);

赋值renderer->av4\_position顶点信息

IJK\_EGL\_setSurfaceSize egl->window宽高和format赋值

glViewport(0, 0, egl->width, egl->height)设定绘制区域

2.2、IJK\_GLES2\_Renderer\_renderOverlay

func\_getBufferWidth

func\_uploadTexture yuv420p\_uploadTexture

glBindTexture(GL\_TEXTURE\_2D, renderer->plane\_textures[i]);创建纹理，一个GPU中最多可以有8个纹理单元

glTexImage2D给纹理i传入像素数据

glDrawArrays使用传入顶点中的哪几个顶点进行绘制，而且把用于绘制的点，如何进行图元装配，我们选择三角绘制

eglSwapBuffers把后台的buffer显示到前台，前台buffer回到后台重新着色。