OpenGL大作业报告

—戴自霖

2016-12-20

目录

- 鼠标、键盘事件
- OpenAL播放音频

鼠标、键盘事件

- **鼠标、键盘事件**
- OpenAL播放音频

鼠标、键盘事件

```
glfwSetCursorPosCallback(fwWindow, MouseMotionEvent);
glfwSetScrollCallback(fwWindow, MouseWheelEvent);
glfwSetDropCallback(fwWindow, MouseDropEvent);
glfwSetKeyCallback(fwWindow, KeyEvent);

void MouseMotionEvent(GLFWwindow* w, double x, double y)
void MouseWheelEvent(GLFWwindow* w, double x, double y)
void MouseDropEvent(GLFWwindow* w, int c, const char** p)
void KeyEvent(GLFWwindow *w, int key, int scancode, int
action, int mods)
```

MouseMotionEvent

- 左键按下：绕xOy平面内的轴旋转
- 中键按下：绕z轴旋转
- 右键按下：平移
- 上述操作对应矩阵**左乘**到变换矩阵上

旋转与平移

- 旋转

```
GLM_FUNC_QUALIFIER tmat4x4<T, P> rotate(
    tmat4x4<T, P> const & m, T angle,
    tvec3<T, P> const & v
)
```

- 平移

```
GLM_FUNC_QUALIFIER tmat4x4<T, P> translate(
    tmat4x4<T, P> const & m, tvec3<T, P> const & v
)
```

MouseWheelEvent

- 缩放
- 实际上是z轴上的平移

MouseDropEvent

- 读取拖进去的文件，然后根据得到的类型new一个物体
- 如果已经读取过（根据文件路径判断），直接new
- 根据鼠标位置设置新物体的位置、速度、角速度：gluUnProject

```
GLint gluUnProject (GLdouble winX, GLdouble winY, GLdouble winZ, const
GLdouble *model, const GLdouble *proj, const GLint *view, GLdouble* objX,
GLdouble* objY, GLdouble* objZ)
```

OpenAL播放音频

- 鼠标、键盘事件
- **OpenAL播放音频**

OpenAL

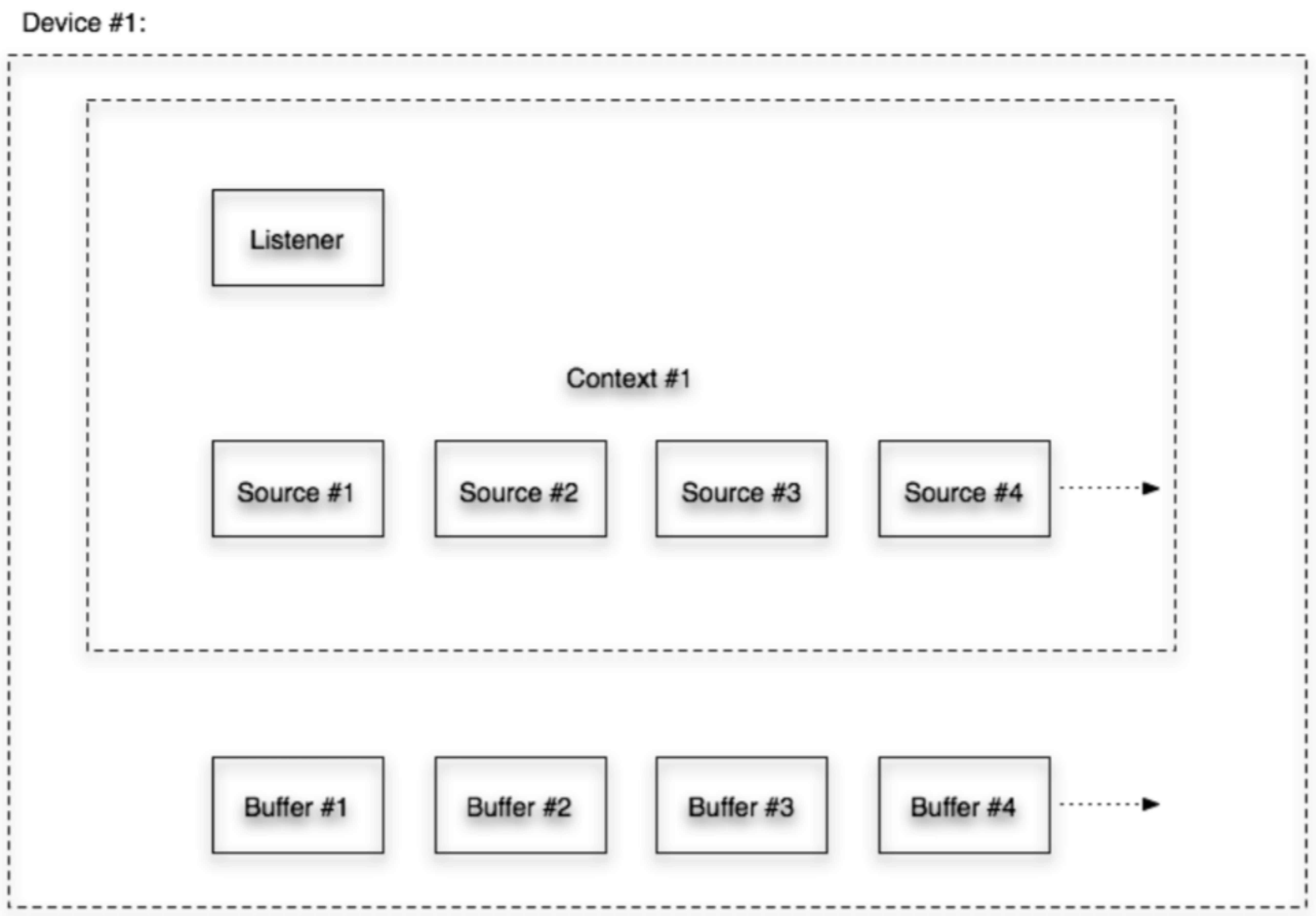
- OpenAL (Open Audio Library) is a cross-platform audio application programming interface (API). It is designed for efficient rendering of multichannel three-dimensional positional audio. Its API style and conventions deliberately resemble those of OpenGL.
- —https://en.wikipedia.org/wiki/OpenAL

OpenAL

- For example,
- gl.h vs. al.h
- glut.h vs. alut.h

OpenAL objects

- Device
- Context
- Listener
- Buffer
- Source
- Each source can have one or more buffers objects attached to it.


Device #1:
Listener
Context #1
Source #1
Source #2
Source #3
Source #4
Buffer #1
Buffer #2
Buffer #3
Buffer #4

播放音频的实现

- 单例模式
- 预先申请source，空间不足时延时播放

单例模式

- 类的实例（instance）

单例模式

```
class TestInstance {
private:
    TestInstance(const TestInstance &);
    TestInstance & operator =
                  (const TestInstance &);
    TestInstance() {}
public:
    ~TestInstance() {}
    static TestInstance *GetInstance() {
        static TestInstance msInstance;
        return &msInstance;
    }
};
```

出错的例子

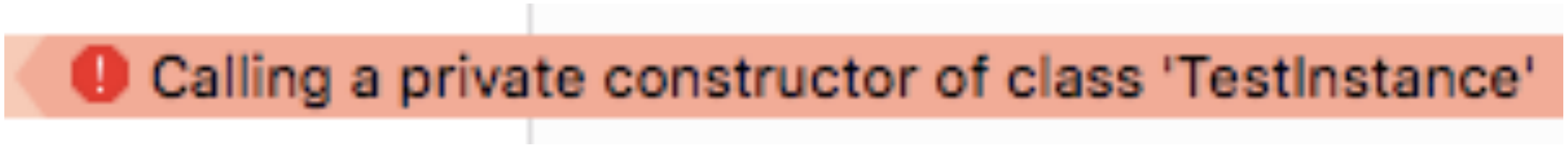
```
int main() {
    TestInstance *ptr1, *ptr2,
                 &ref1 =*TestInstance::GetInstance();
    TestInstance cpy1 = *TestInstance::GetInstance();
    printf("%p\n", &ref1);
    printf("%p\n", (ptr1 = TestInstance::GetInstance()));
    printf("%p\n", (ptr2 = ptr1->GetInstance()));
    printf("%p\n", &cpy1);
}
```

a possible output:

```
0x100502070
0x100502070
0x100502070
0x7fff5fbff780
```

这种用法本身就不遵守单例模式，应该想办法阻止。

所以要阻止这种用法


```
Calling a private constructor of class 'TestInstance'
```