cppGameTemplate

鉴于面向对象课程设计要写一个小游戏，我在网上随便抄了抄来写了一个框架，到时候方便写也方便ai生成，大部分的都只需要进行比较机械重复的工作就好了

本地调试

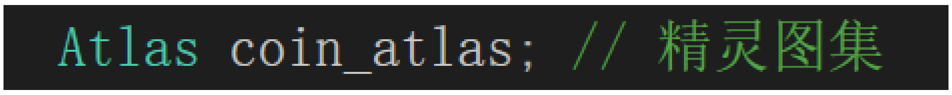
可视化库

在网上看教程用的是`easyx`，对于其中的一些功能想要查阅的话可以直接看文档，需要安装的话直接执行文件夹中的`easyx`的可执行文件就安装好了，在使用的时候只需要`#include<graphics.h>`来导入头文件即可

开发流程

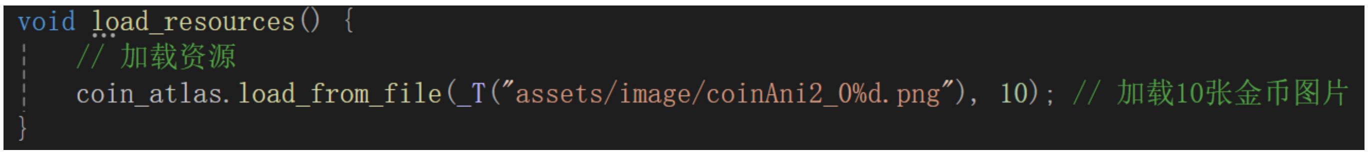
素材导入

把老师给的素材弄成一个个对应的图片，也就是俗称的精灵，序列帧图片拆分教程，然后在`main.cpp`中去声明这个全局变量



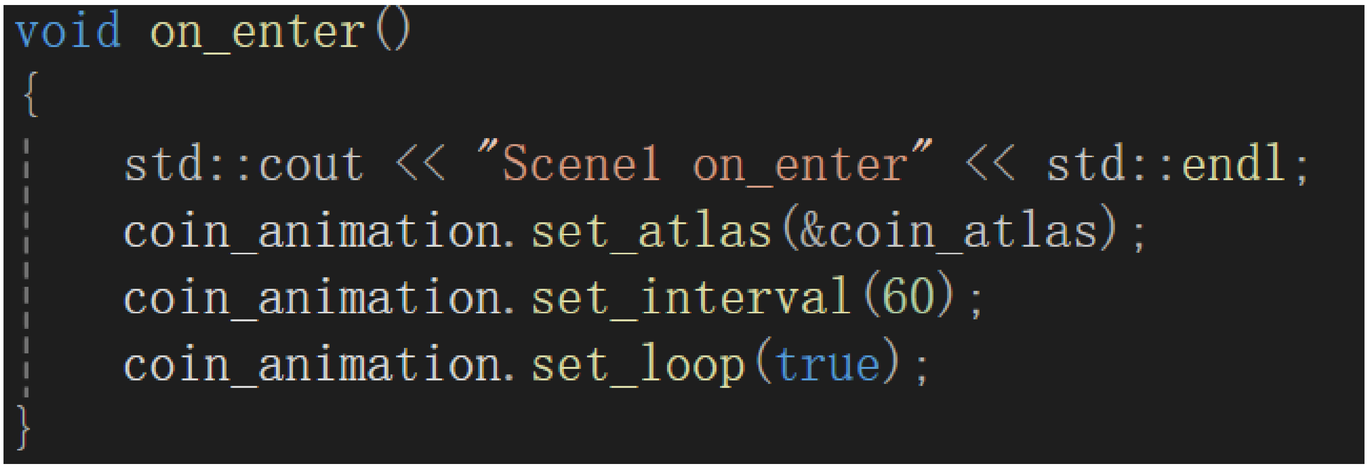
```
Atlas coin_atlas; // 精灵图集
```

然后在`load_resources`中去加载资源

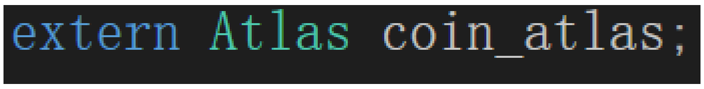


```
void load_resources() {
    // 加载资源
    coin_atlas.load_from_file(_T("assets/image/coinAni2_0%d.png"), 10); // 加载10张金币图片
}
```

在对应的场景中利用图集去加载动画

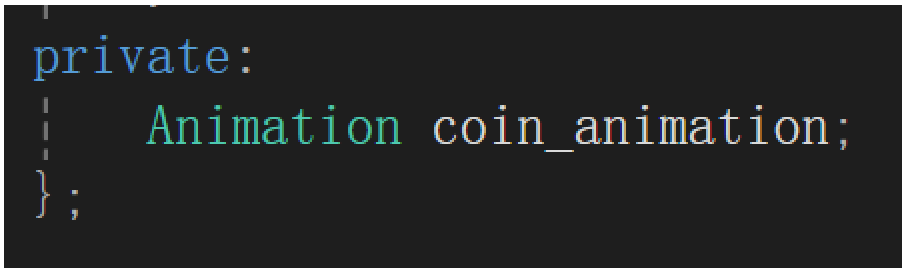


```
void on_enter()
{
    std::cout << "Scene1 on_enter" << std::endl;
    coin_animation.set_atlas(&coin_atlas);
    coin_animation.set_interval(60);
    coin_animation.set_loop(true);
}
```

下面的获取图集的代码 

```
extern Atlas coin_atlas;
```

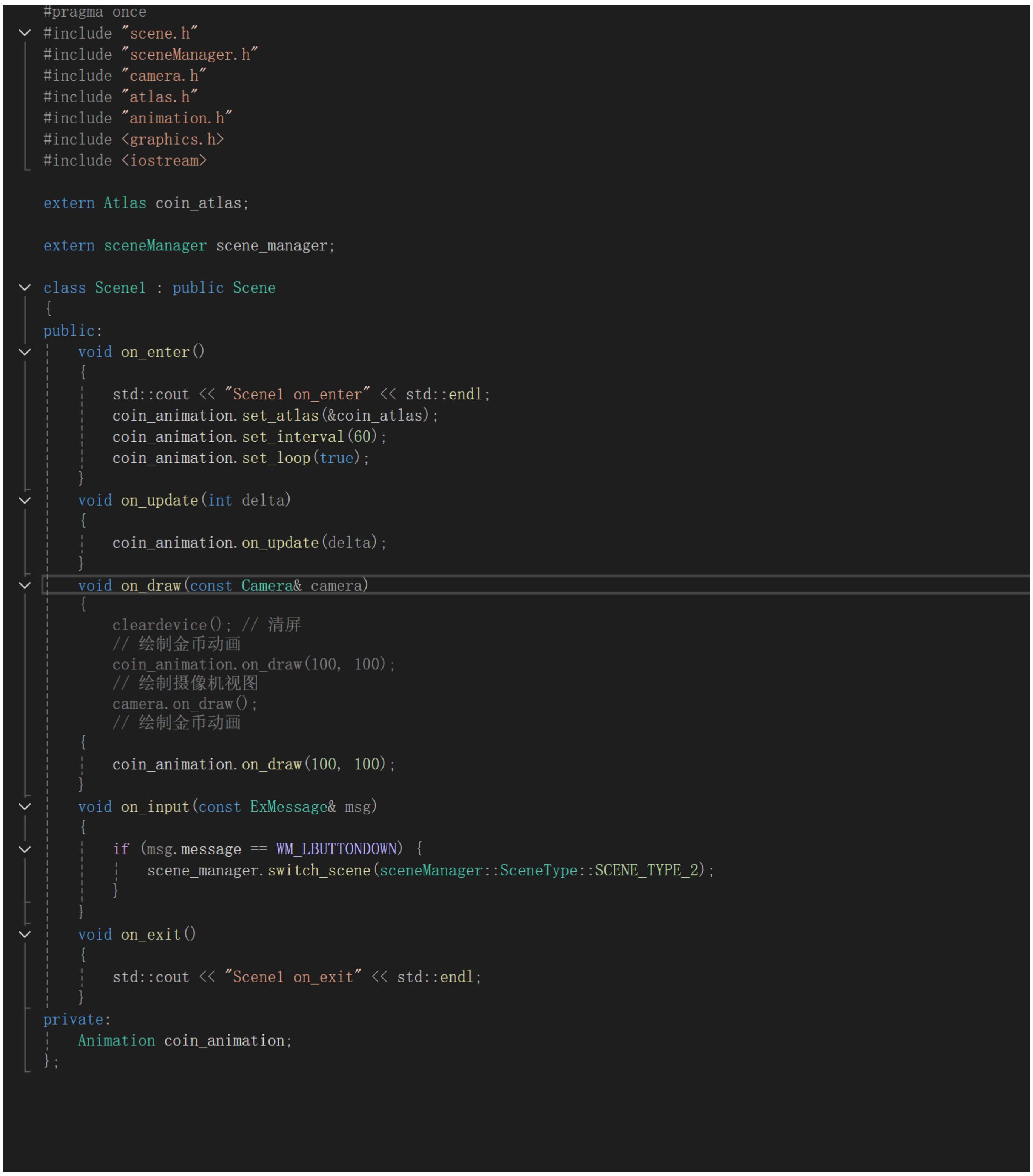
私有的动画对象



```
private:
    Animation coin_animation;
};
```

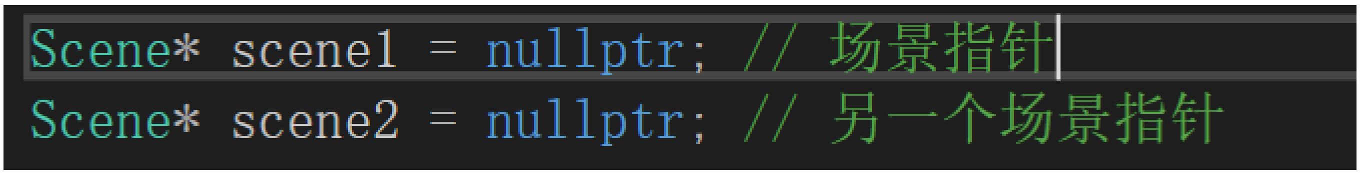
场景创建

在`scene`过滤器下面创建对应的场景头文件，然后创建场景类去实现方法（灰色的不要管，是ai的提示）

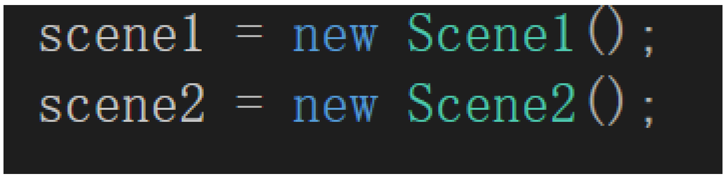


```cpp
#pragma once
#include "scene.h"
#include "sceneManager.h"
#include "camera.h"
#include "atlas.h"
#include "animation.h"
#include <graphics.h>
#include <iostream>

extern Atlas coin_atlas;

extern sceneManager scene_manager;

class Scene1 : public Scene
{
public:
    void on_enter()
    {
        std::cout << "Scene1 on_enter" << std::endl;
        coin_animation.set_atlas(&coin_atlas);
        coin_animation.set_interval(60);
        coin_animation.set_loop(true);
    }
    void on_update(int delta)
    {
        coin_animation.on_update(delta);
    }
    void on_draw(const Camera& camera)
    {
        cleardevice(); // 清屏
        // 绘制金币动画
        coin_animation.on_draw(100, 100);
        // 绘制摄像机视图
        camera.on_draw();
        // 绘制金币动画
    {
        coin_animation.on_draw(100, 100);
    }
    void on_input(const ExMessage& msg)
    {
        if (msg.message == WM_LBUTTONDOWN) {
            scene_manager.switch_scene(sceneManager::SceneType::SCENE_TYPE_2);
        }
    }
    void on_exit()
    {
        std::cout << "Scene1 on_exit" << std::endl;
    }
private:
    Animation coin_animation;
};
```

在`main.cpp`中去声明对应的场景变量，然后实例化

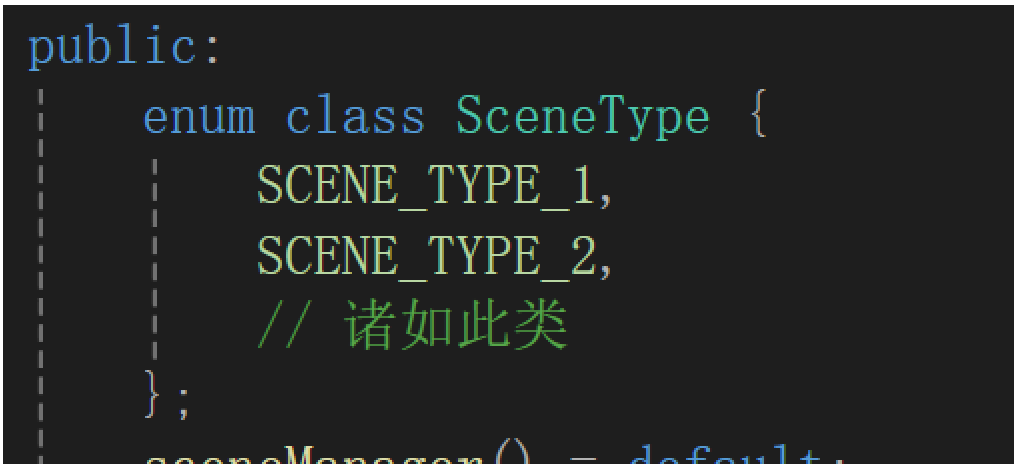


```cpp
Scene* scene1 = nullptr; // 场景指针
Scene* scene2 = nullptr; // 另一个场景指针
```



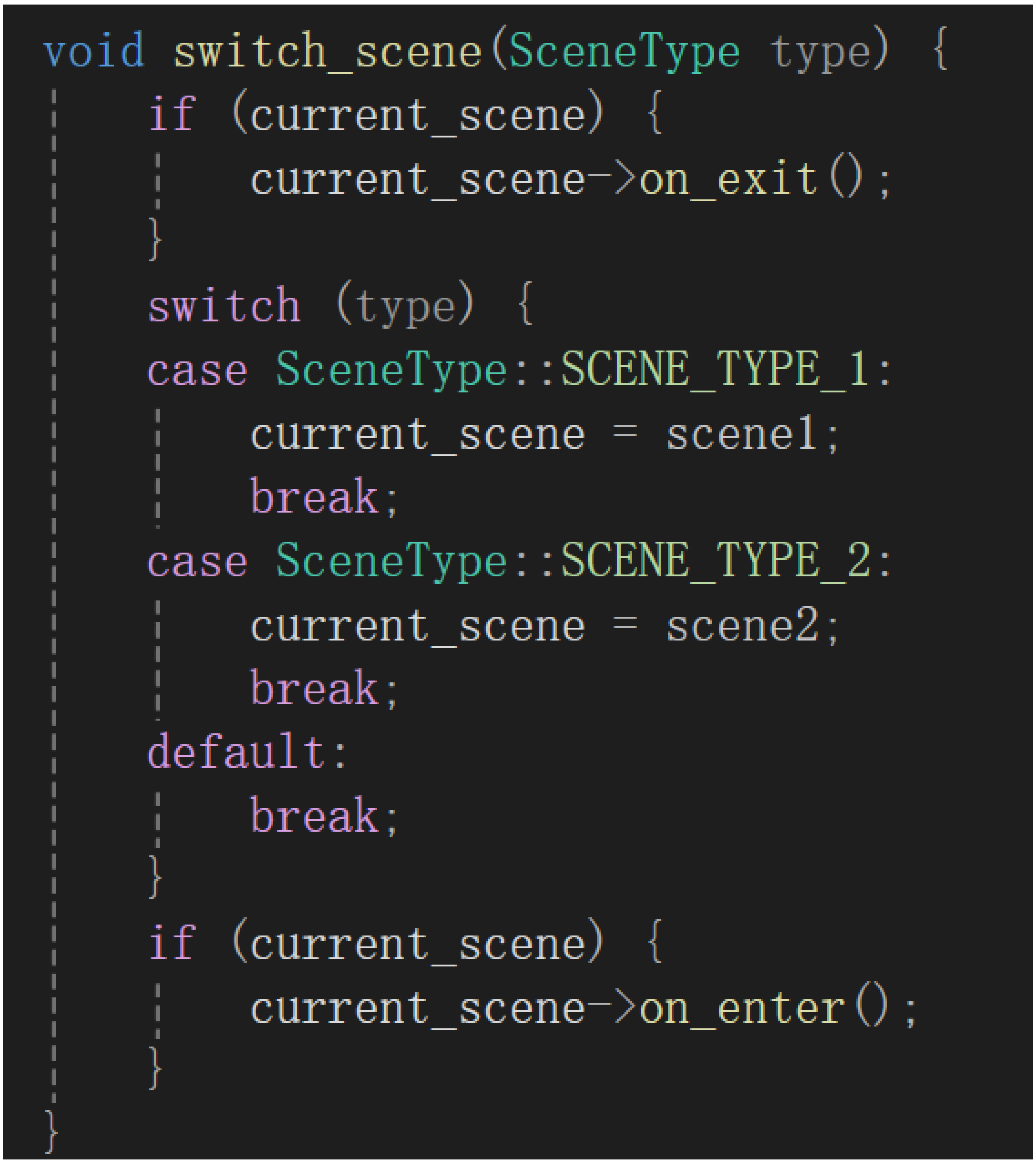
```cpp
scene1 = new Scene1();
scene2 = new Scene2();
```

在场景管理器`sceneManager.h`中修改枚举类，修



```cpp
public:
    enum class SceneType {
        SCENE_TYPE_1,
        SCENE_TYPE_2,
        // 诸如此类
    };
    sceneManager() = default;
```

改`switch_scene`方法



```cpp
void switch_scene(SceneType type) {
    if (current_scene) {
        current_scene->on_exit();
    }
    switch (type) {
    case SceneType::SCENE_TYPE_1:
        current_scene = scene1;
        break;
    case SceneType::SCENE_TYPE_2:
        current_scene = scene2;
        break;
    default:
        break;
    }
    if (current_scene) {
        current_scene->on_enter();
    }
}
```

接着按照自己的逻辑去修改场景的那几个`on_enter`,`on_update`,`on_draw`,`on_input`,`on_exit`方法基本上就完事了，ai写自己手写都很方便