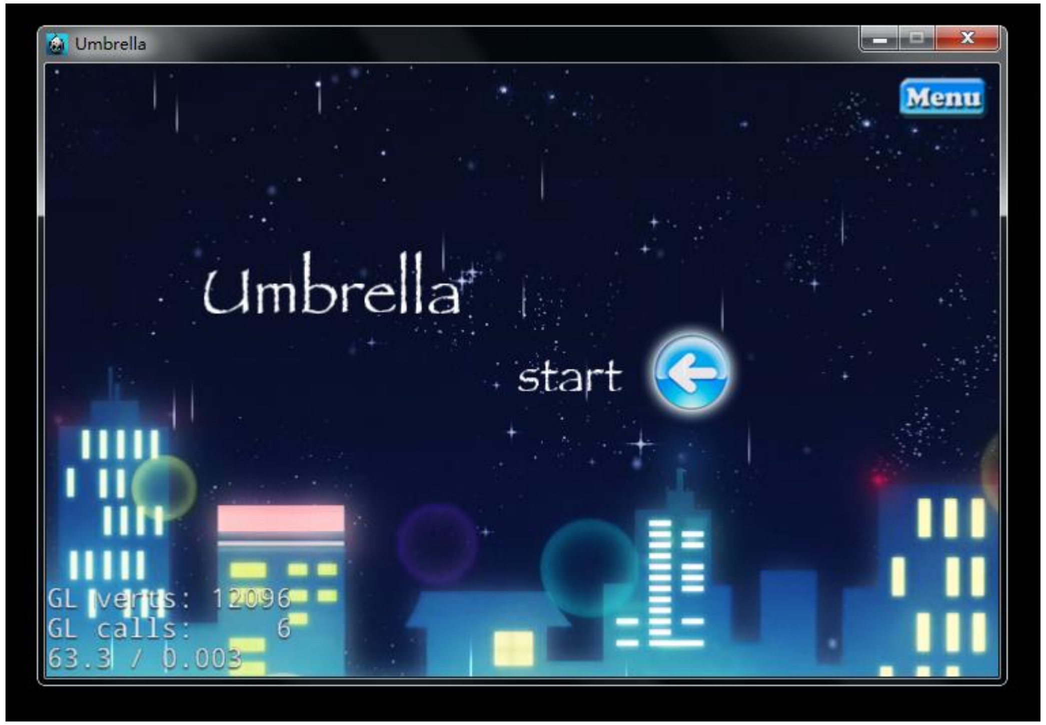
Umbrella

Team 61

1452712

Wang Jiahui



```
GL verts:  12096
GL calls:      6
63.3 / 0.003
```

Catalog

Introduction

- 项目内容

- 主要设计模式

I. 工厂模式
II. 装饰模式
III. 观察者模式
IV. 策略模式 (结合简单工厂模式)
V. 单例模式

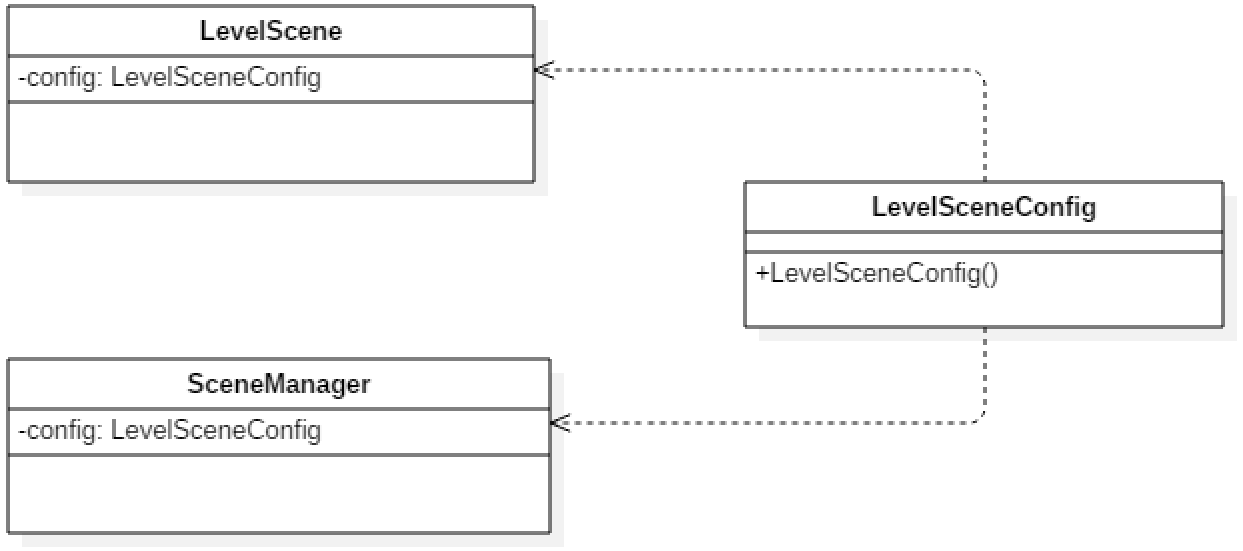
Basic Principle

- 基本原则

I. 面向对象编程 (OOP)
II. 可扩展性 (Extensible)
III. 开放封闭原则 (OCP)
IV. 高内聚低耦合

Design Pattern

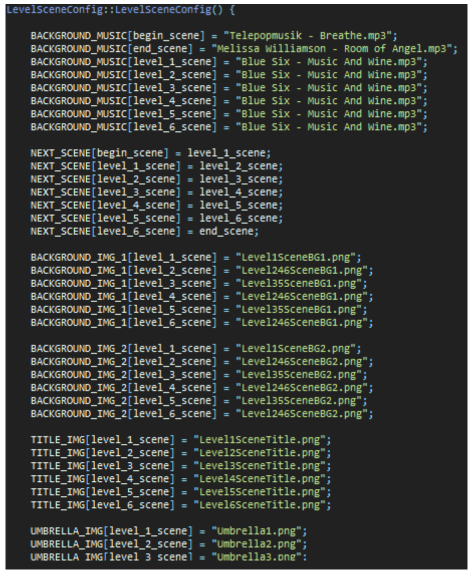
- OOP —— Composition



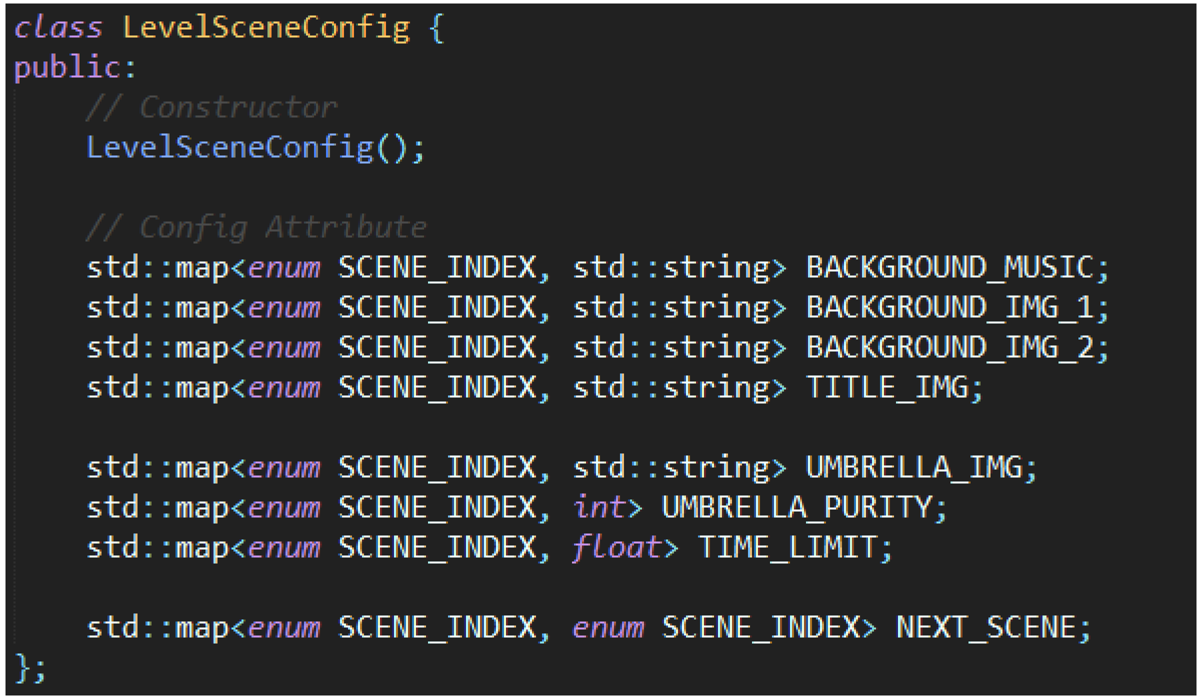
Design Pattern

- **OOP —— Composition**

  LevelSceneConfig



```
LevelSceneConfig::LevelSceneConfig() {

    BACKGROUND_MUSIC[begin_scene] = "Telepopmusik - Breathe.mp3";
    BACKGROUND_MUSIC[end_scene] = "Melissa Williamson - Room of Angel.mp3";
    BACKGROUND_MUSIC[level_1_scene] = "Blue Six - Music And Wine.mp3";
    BACKGROUND_MUSIC[level_2_scene] = "Blue Six - Music And Wine.mp3";
    BACKGROUND_MUSIC[level_3_scene] = "Blue Six - Music And Wine.mp3";
    BACKGROUND_MUSIC[level_4_scene] = "Blue Six - Music And Wine.mp3";
    BACKGROUND_MUSIC[level_5_scene] = "Blue Six - Music And Wine.mp3";
    BACKGROUND_MUSIC[level_6_scene] = "Blue Six - Music And Wine.mp3";

    NEXT_SCENE[begin_scene] = level_1_scene;
    NEXT_SCENE[level_1_scene] = level_2_scene;
    NEXT_SCENE[level_2_scene] = level_3_scene;
    NEXT_SCENE[level_3_scene] = level_4_scene;
    NEXT_SCENE[level_4_scene] = level_5_scene;
    NEXT_SCENE[level_5_scene] = level_6_scene;
    NEXT_SCENE[level_6_scene] = end_scene;

    BACKGROUND_IMG_1[level_1_scene] = "Level1SceneBG1.png";
    BACKGROUND_IMG_1[level_2_scene] = "Level246SceneBG1.png";
    BACKGROUND_IMG_1[level_3_scene] = "Level35SceneBG1.png";
    BACKGROUND_IMG_1[level_4_scene] = "Level246SceneBG1.png";
    BACKGROUND_IMG_1[level_5_scene] = "Level35SceneBG1.png";
    BACKGROUND_IMG_1[level_6_scene] = "Level246SceneBG1.png";

    BACKGROUND_IMG_2[level_1_scene] = "Level1SceneBG2.png";
    BACKGROUND_IMG_2[level_2_scene] = "Level246SceneBG2.png";
    BACKGROUND_IMG_2[level_3_scene] = "Level35SceneBG2.png";
    BACKGROUND_IMG_2[level_4_scene] = "Level246SceneBG2.png";
    BACKGROUND_IMG_2[level_5_scene] = "Level35SceneBG2.png";
    BACKGROUND_IMG_2[level_6_scene] = "Level246SceneBG2.png";

    TITLE_IMG[level_1_scene] = "Level1SceneTitle.png";
    TITLE_IMG[level_2_scene] = "Level2SceneTitle.png";
    TITLE_IMG[level_3_scene] = "Level3SceneTitle.png";
    TITLE_IMG[level_4_scene] = "Level4SceneTitle.png";
    TITLE_IMG[level_5_scene] = "Level5SceneTitle.png";
    TITLE_IMG[level_6_scene] = "Level6SceneTitle.png";

    UMBRELLA_IMG[level_1_scene] = "Umbrella1.png";
    UMBRELLA_IMG[level_2_scene] = "Umbrella2.png";
    UMBRELLA_IMG[level_3_scene] = "Umbrella3.png";
```

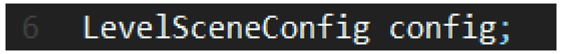


```
class LevelSceneConfig {
public:
    // Constructor
    LevelSceneConfig();

    // Config Attribute
    std::map<enum SCENE_INDEX, std::string> BACKGROUND_MUSIC;
    std::map<enum SCENE_INDEX, std::string> BACKGROUND_IMG_1;
    std::map<enum SCENE_INDEX, std::string> BACKGROUND_IMG_2;
    std::map<enum SCENE_INDEX, std::string> TITLE_IMG;

    std::map<enum SCENE_INDEX, std::string> UMBRELLA_IMG;
    std::map<enum SCENE_INDEX, int> UMBRELLA_PURITY;
    std::map<enum SCENE_INDEX, float> TIME_LIMIT;

    std::map<enum SCENE_INDEX, enum SCENE_INDEX> NEXT_SCENE;
};
```

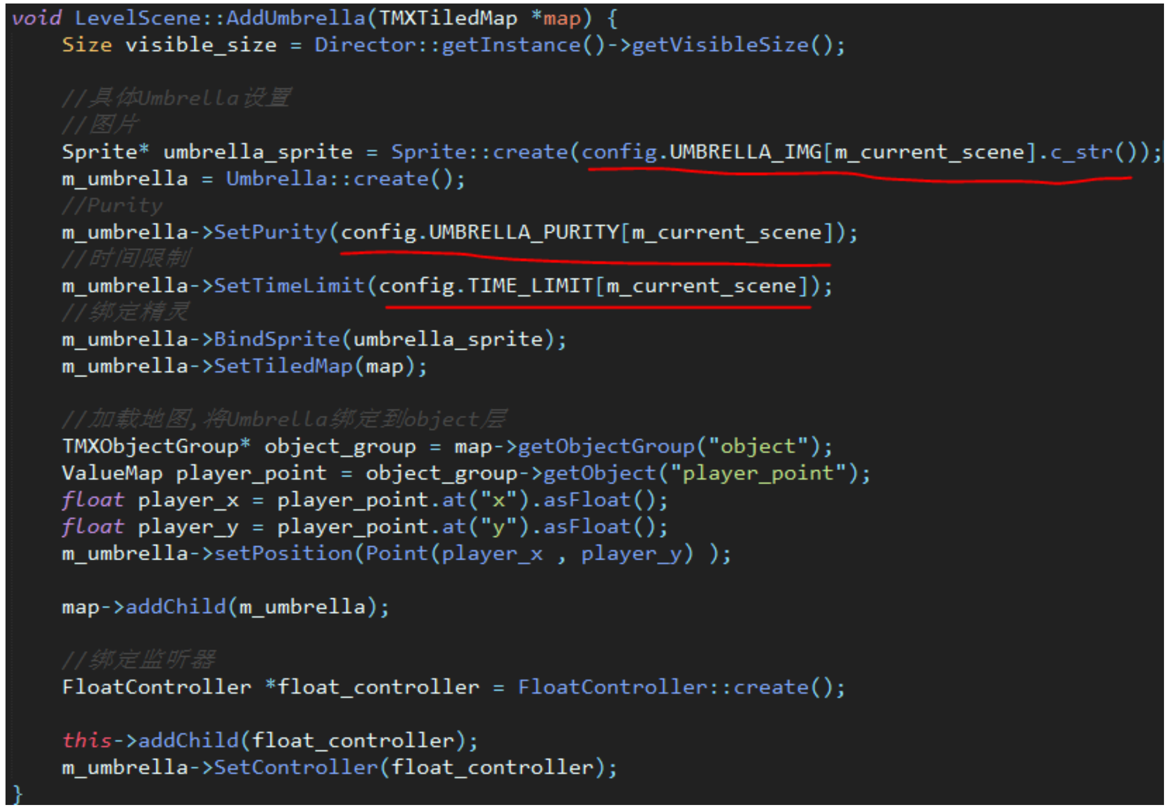
Design Pattern

- **OOP —— Composition**

  LevelScene

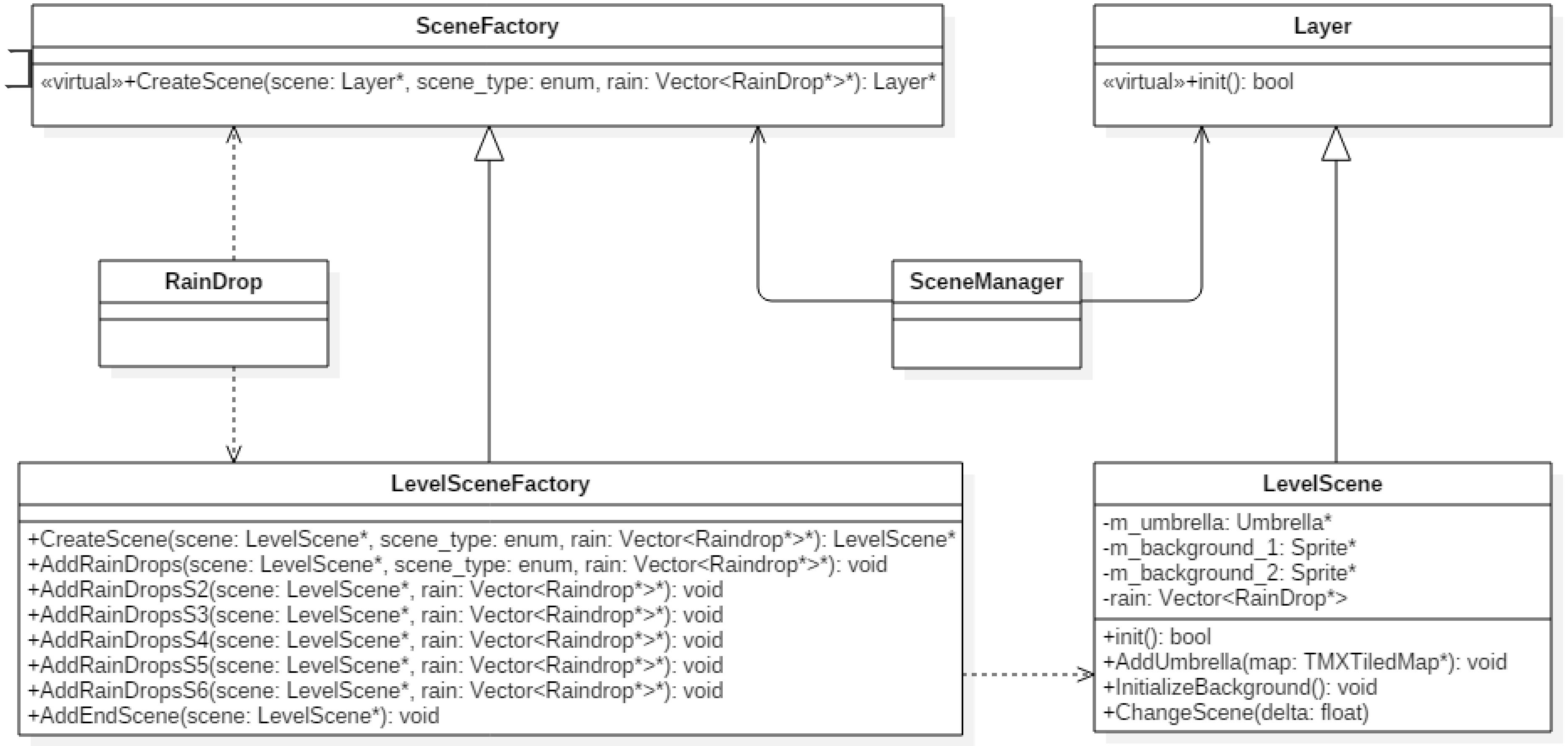


```
LevelSceneConfig config;
```



```
void LevelScene::AddUmbrella(TMXTiledMap *map) {
    Size visible_size = Director::getInstance()->getVisibleSize();

    //具体Umbrella设置
    //图片
    Sprite* umbrella_sprite = Sprite::create(config.UMBRELLA_IMG[m_current_scene].c_str());
    m_umbrella = Umbrella::create();
    //Purity
    m_umbrella->SetPurity(config.UMBRELLA_PURITY[m_current_scene]);
    //时间限制
    m_umbrella->SetTimeLimit(config.TIME_LIMIT[m_current_scene]);
    //绑定精灵
    m_umbrella->BindSprite(umbrella_sprite);
    m_umbrella->SetTiledMap(map);

    //加载地图,将Umbrella绑定到object层
    TMXObjectGroup* object_group = map->getObjectGroup("object");
    ValueMap player_point = object_group->getObject("player_point");
    float player_x = player_point.at("x").asFloat();
    float player_y = player_point.at("y").asFloat();
    m_umbrella->setPosition(Point(player_x , player_y) );

    map->addChild(m_umbrella);

    //绑定监听器
    FloatController *float_controller = FloatController::create();

    this->addChild(float_controller);
    m_umbrella->SetController(float_controller);
}
```

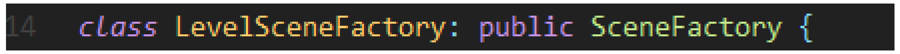
Design Pattern


SceneFactory
«virtual»+CreateScene(scene: Layer*, scene_type: enum, rain: Vector<RainDrop*>*): Layer*
Layer
«virtual»+init(): bool
RainDrop
SceneManager
LevelSceneFactory
+CreateScene(scene: LevelScene*, scene_type: enum, rain: Vector<Raindrop*>*): LevelScene*
+AddRainDrops(scene: LevelScene*, scene_type: enum, rain: Vector<Raindrop*>*): void
+AddRainDropsS2(scene: LevelScene*, rain: Vector<Raindrop*>*): void
+AddRainDropsS3(scene: LevelScene*, rain: Vector<Raindrop*>*): void
+AddRainDropsS4(scene: LevelScene*, rain: Vector<Raindrop*>*): void
+AddRainDropsS5(scene: LevelScene*, rain: Vector<Raindrop*>*): void
+AddRainDropsS6(scene: LevelScene*, rain: Vector<Raindrop*>*): void
+AddEndScene(scene: LevelScene*): void
LevelScene
-m_umbrella: Umbrella*
-m_background_1: Sprite*
-m_background_2: Sprite*
-rain: Vector<RainDrop*>
+init(): bool
+AddUmbrella(map: TMXTiledMap*): void
+InitializeBackground(): void
+ChangeScene(delta: float)

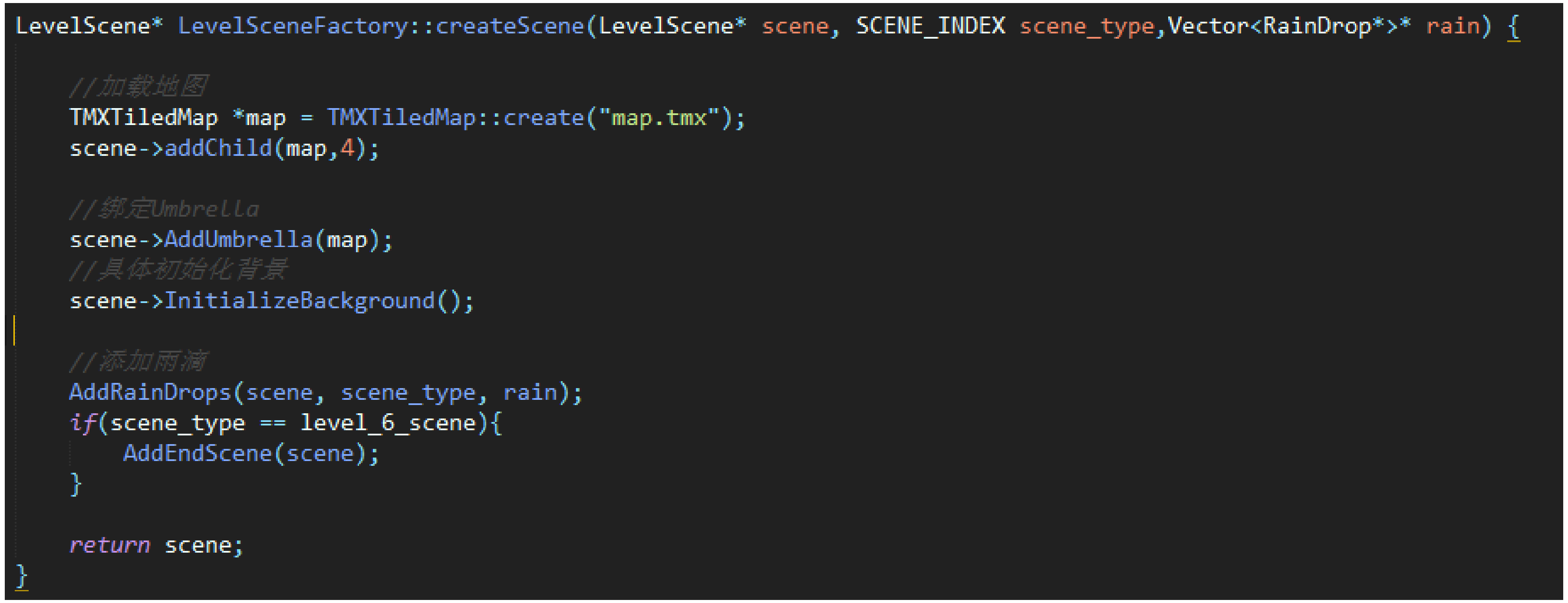
Design Pattern

- 工厂模式

  LevelSceneFactory



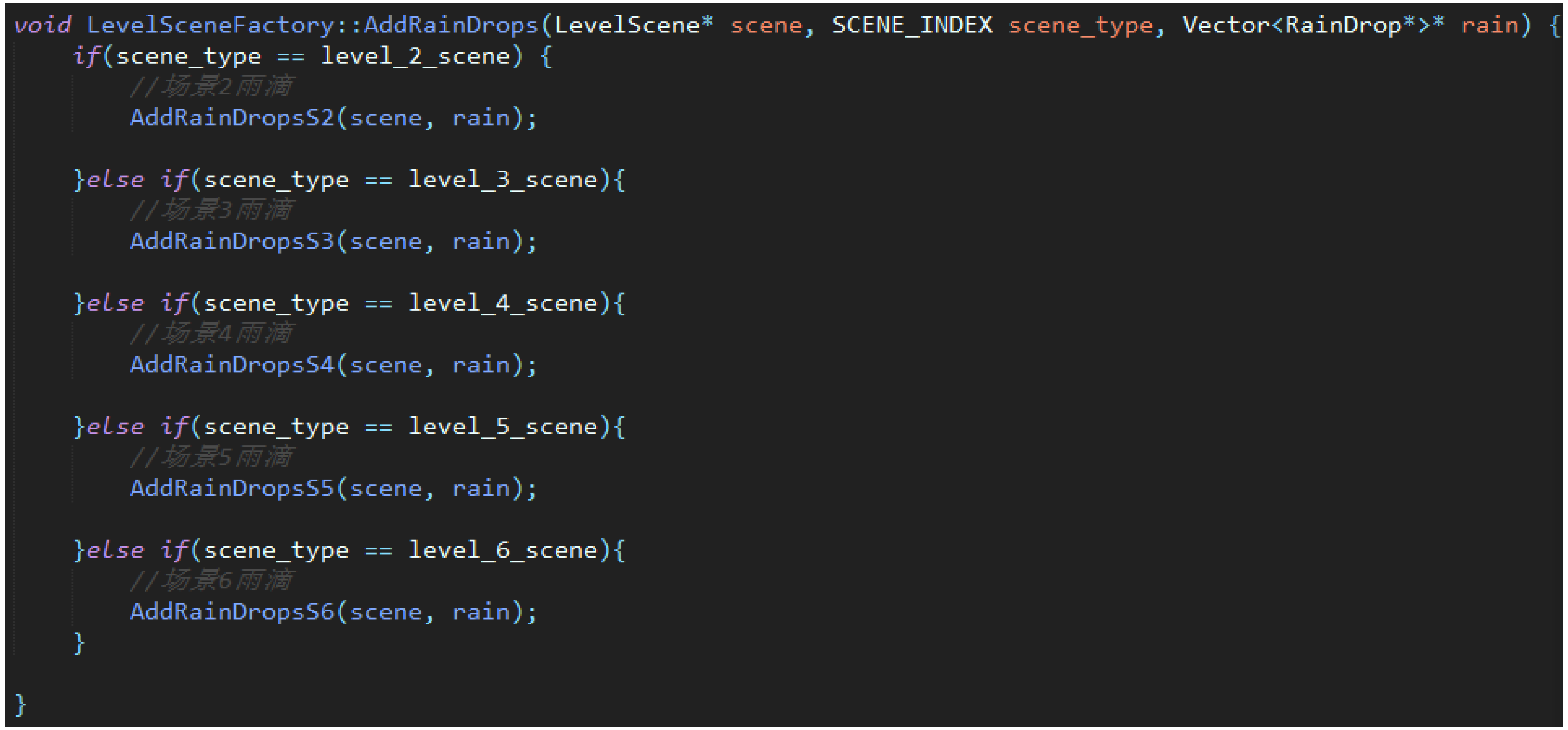
```
class LevelSceneFactory: public SceneFactory {
```



```
LevelScene* LevelSceneFactory::createScene(LevelScene* scene, SCENE_INDEX scene_type,Vector<RainDrop*>* rain) {

    //加载地图
    TMXTiledMap *map = TMXTiledMap::create("map.tmx");
    scene->addChild(map,4);

    //绑定Umbrella
    scene->AddUmbrella(map);
    //具体初始化背景
    scene->InitializeBackground();

    //添加雨滴
    AddRainDrops(scene, scene_type, rain);
    if(scene_type == level_6_scene){
        AddEndScene(scene);
    }

    return scene;
}
```

Design Pattern

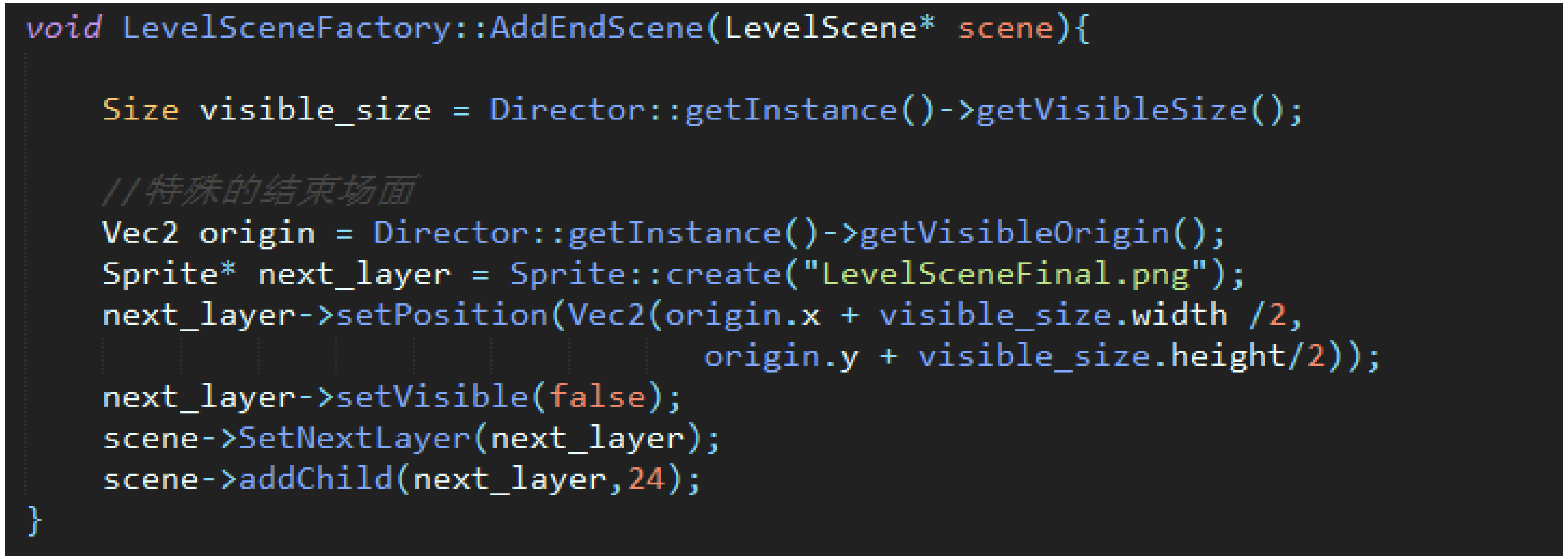
- 工厂模式



```
void LevelSceneFactory::AddRainDrops(LevelScene* scene, SCENE_INDEX scene_type, Vector<RainDrop*>* rain) {
    if(scene_type == level_2_scene) {
        //场景2 雨滴
        AddRainDropsS2(scene, rain);

    }else if(scene_type == level_3_scene){
        //场景3 雨滴
        AddRainDropsS3(scene, rain);

    }else if(scene_type == level_4_scene){
        //场景4 雨滴
        AddRainDropsS4(scene, rain);

    }else if(scene_type == level_5_scene){
        //场景5 雨滴
        AddRainDropsS5(scene, rain);

    }else if(scene_type == level_6_scene){
        //场景6 雨滴
        AddRainDropsS6(scene, rain);
    }

}
```

Design Pattern

- 工厂模式

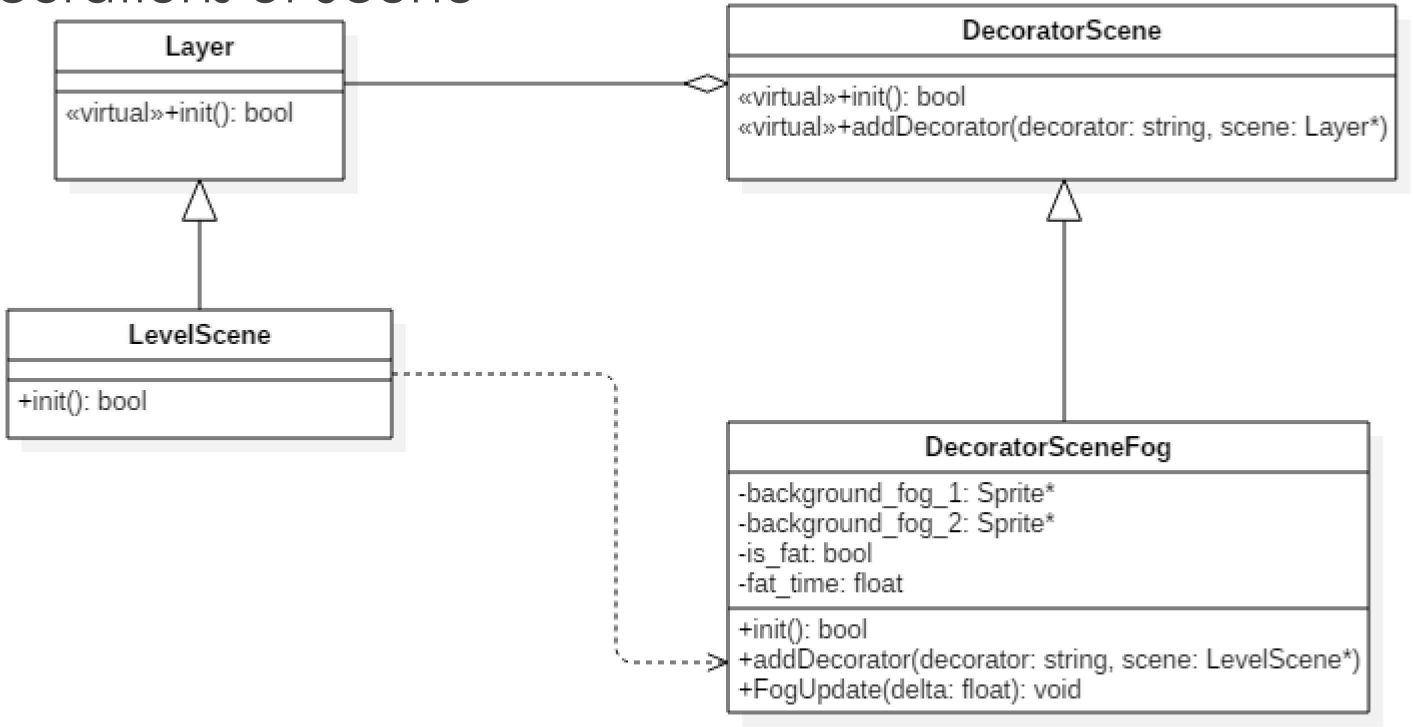


```
void LevelSceneFactory::AddEndScene(LevelScene* scene){

    Size visible_size = Director::getInstance()->getVisibleSize();

    //特殊的结束场面
    Vec2 origin = Director::getInstance()->getVisibleOrigin();
    Sprite* next_layer = Sprite::create("LevelSceneFinal.png");
    next_layer->setPosition(Vec2(origin.x + visible_size.width /2,
                                   origin.y + visible_size.height/2));
    next_layer->setVisible(false);
    scene->SetNextLayer(next_layer);
    scene->addChild(next_layer,24);
}
```

Design Pattern

- 装饰模式

Extra Decorations of Scene

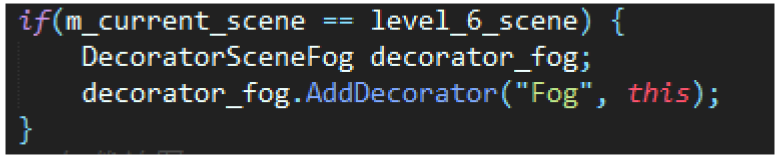


Layer
«virtual»+init(): bool
DecoratorScene
«virtual»+init(): bool
«virtual»+addDecorator(decorator: string, scene: Layer*)
LevelScene
+init(): bool
DecoratorSceneFog
-background_fog_1: Sprite*
-background_fog_2: Sprite*
-is_fat: bool
-fat_time: float
+init(): bool
+addDecorator(decorator: string, scene: LevelScene*)
+FogUpdate(delta: float): void

Design Pattern

- 装饰模式

LevelScene::init()

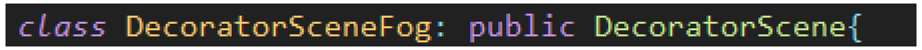


```
if(m_current_scene == level_6_scene) {
    DecoratorSceneFog decorator_fog;
    decorator_fog.AddDecorator("Fog", this);
}
```

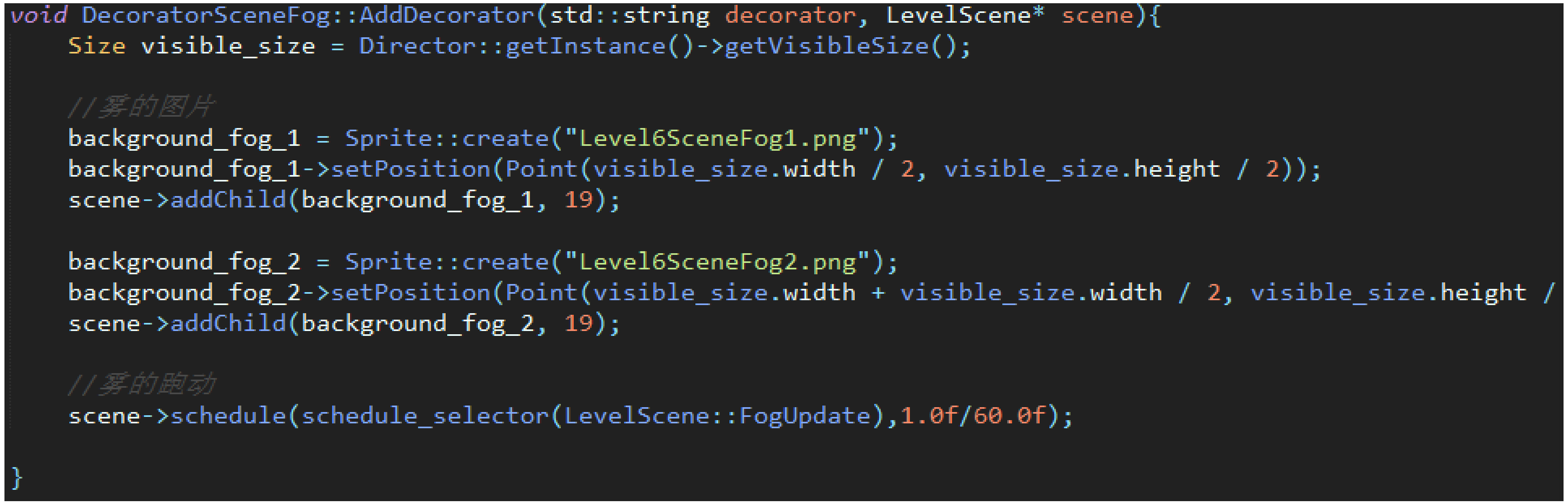
Design Pattern

- 装饰模式

DecoratorSceneFog



```
class DecoratorSceneFog: public DecoratorScene{
```

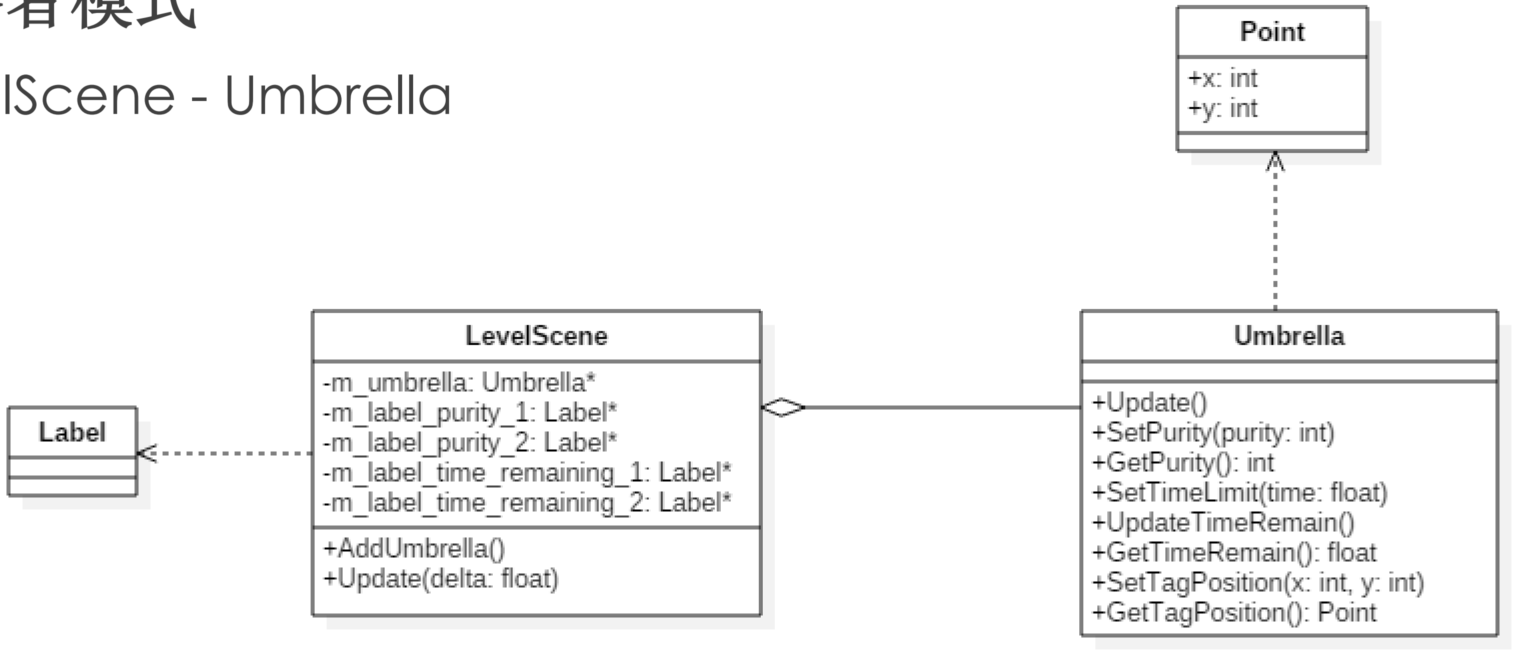


```
void DecoratorSceneFog::AddDecorator(std::string decorator, LevelScene* scene){
    Size visible_size = Director::getInstance()->getVisibleSize();

    //雾的图片
    background_fog_1 = Sprite::create("Level6SceneFog1.png");
    background_fog_1->setPosition(Point(visible_size.width / 2, visible_size.height / 2));
    scene->addChild(background_fog_1, 19);

    background_fog_2 = Sprite::create("Level6SceneFog2.png");
    background_fog_2->setPosition(Point(visible_size.width + visible_size.width / 2, visible_size.height /
    scene->addChild(background_fog_2, 19);

    //雾的跑动
    scene->schedule(schedule_selector(LevelScene::FogUpdate),1.0f/60.0f);

}
```

Design Pattern

- 观察者模式

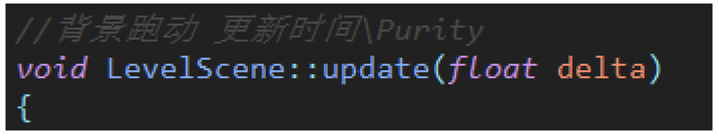
  LevelScene - Umbrella



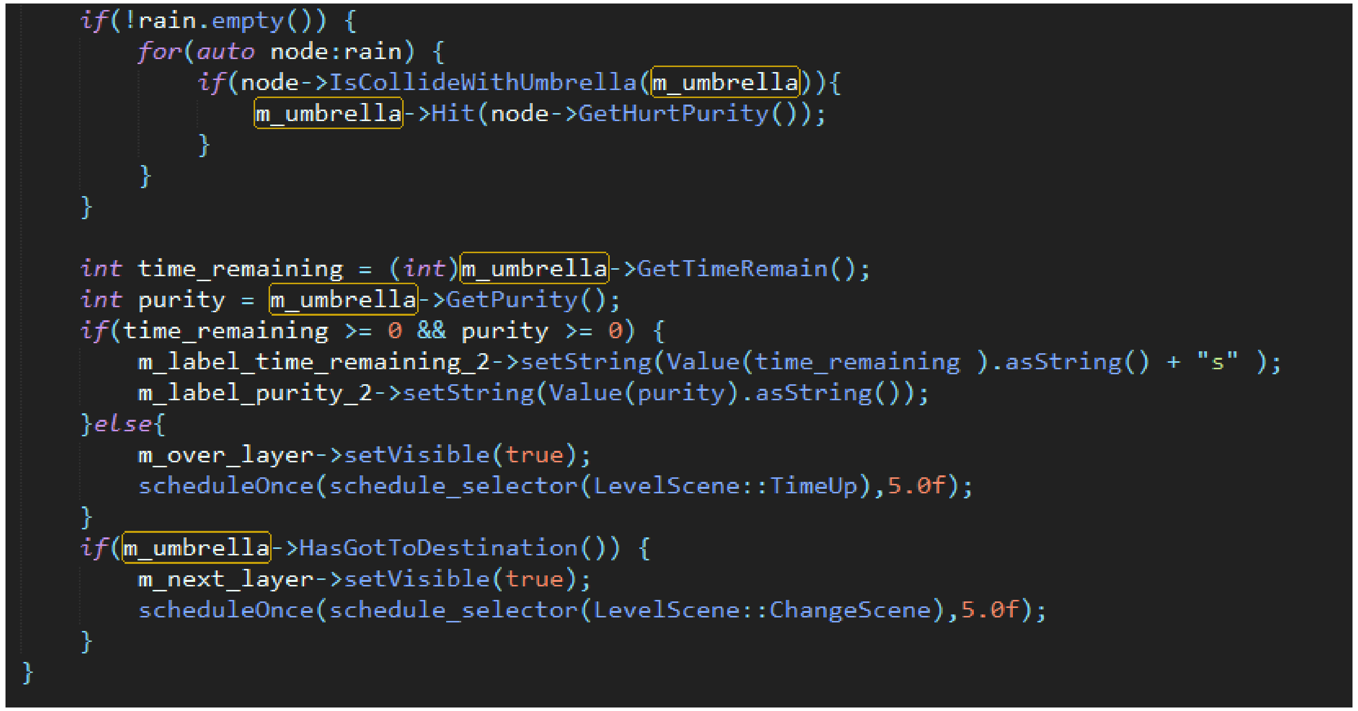
Design Pattern

- 观察者模式

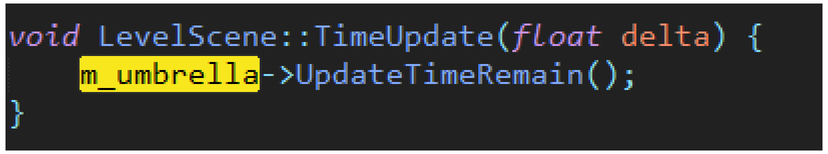
  LevelScene



```cpp
//背景跑动 更新时间\Purity
void LevelScene::update(float delta)
{
```



```cpp
    if(!rain.empty()) {
        for(auto node:rain) {
            if(node->IsCollideWithUmbrella(m_umbrella)){
                m_umbrella->Hit(node->GetHurtPurity());
            }
        }
    }

    int time_remaining = (int)m_umbrella->GetTimeRemain();
    int purity = m_umbrella->GetPurity();
    if(time_remaining >= 0 && purity >= 0) {
        m_label_time_remaining_2->setString(Value(time_remaining ).asString() + "s" );
        m_label_purity_2->setString(Value(purity).asString());
    }else{
        m_over_layer->setVisible(true);
        scheduleOnce(schedule_selector(LevelScene::TimeUp),5.0f);
    }
    if(m_umbrella->HasGotToDestination()) {
        m_next_layer->setVisible(true);
        scheduleOnce(schedule_selector(LevelScene::ChangeScene),5.0f);
    }
}
```

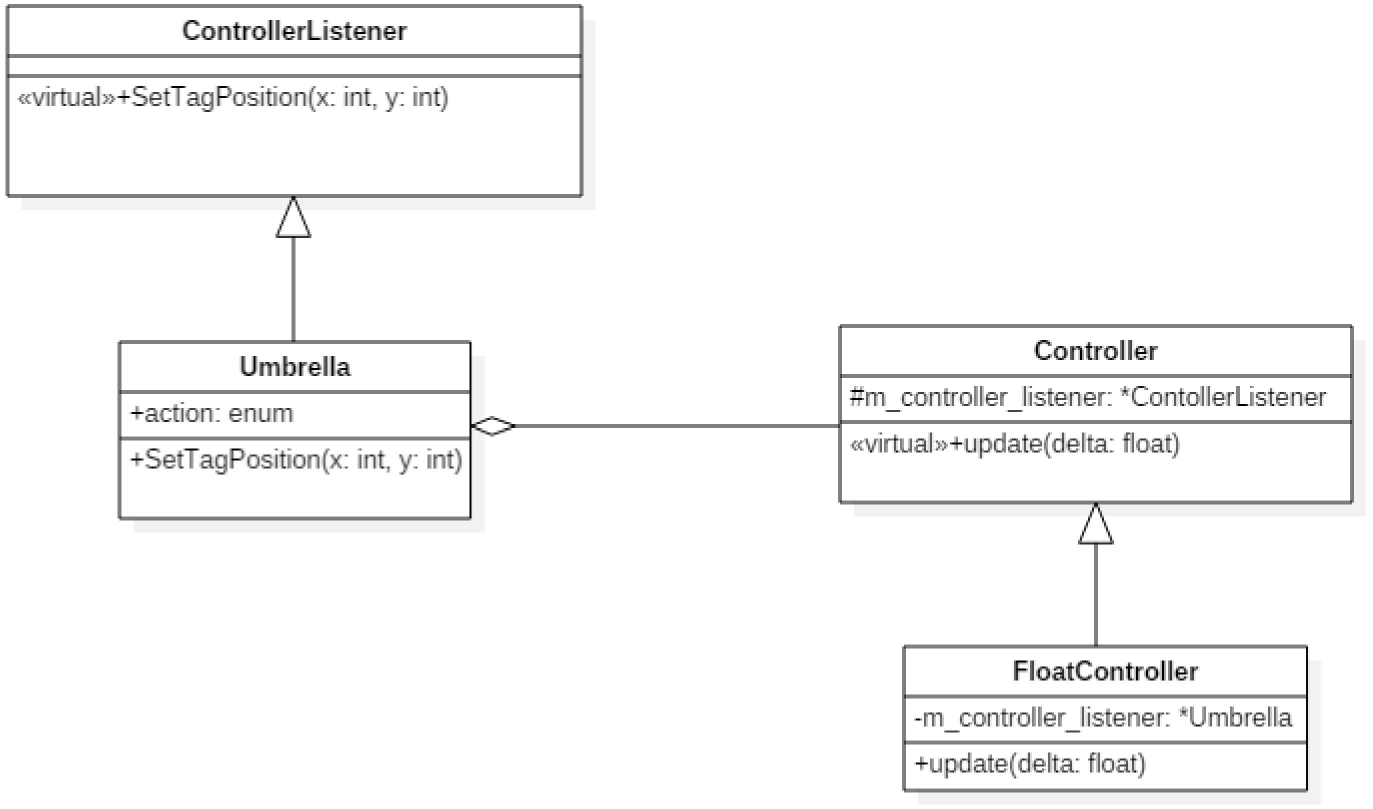


```cpp
void LevelScene::TimeUpdate(float delta) {
    m_umbrella->UpdateTimeRemain();
}
```

Design Pattern

- 策略模式

  Extra Actions

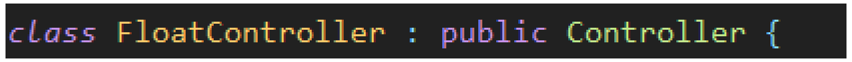


ControllerListener
«virtual»+SetTagPosition(x: int, y: int)
Umbrella
+action: enum
+SetTagPosition(x: int, y: int)
Controller
#m_controller_listener: *ContollerListener
«virtual»+update(delta: float)
FloatController
-m_controller_listener: *Umbrella
+update(delta: float)

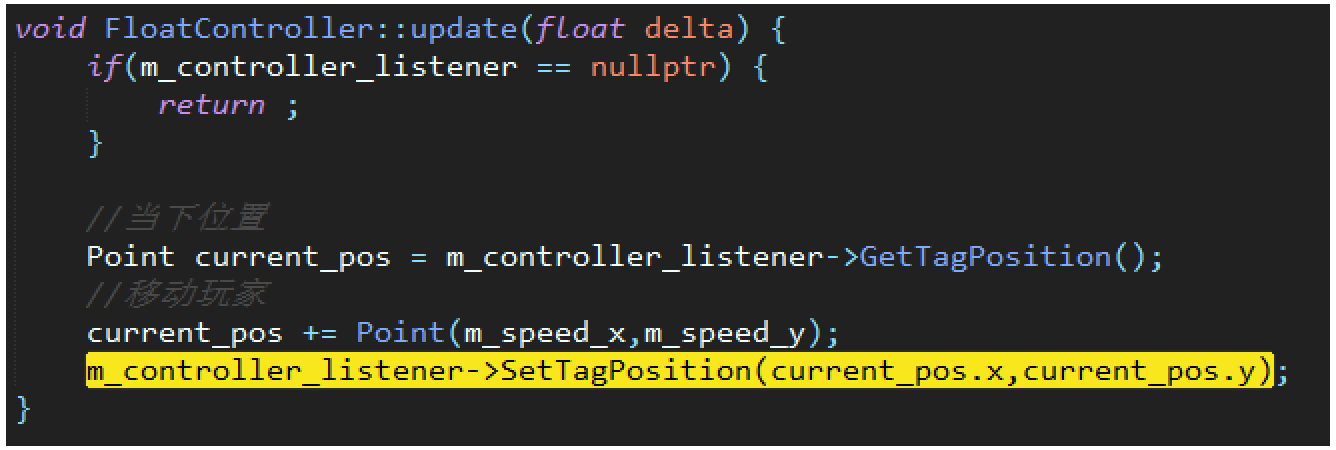
Design Pattern

- 策略模式

FloatController



```
class FloatController : public Controller {
```

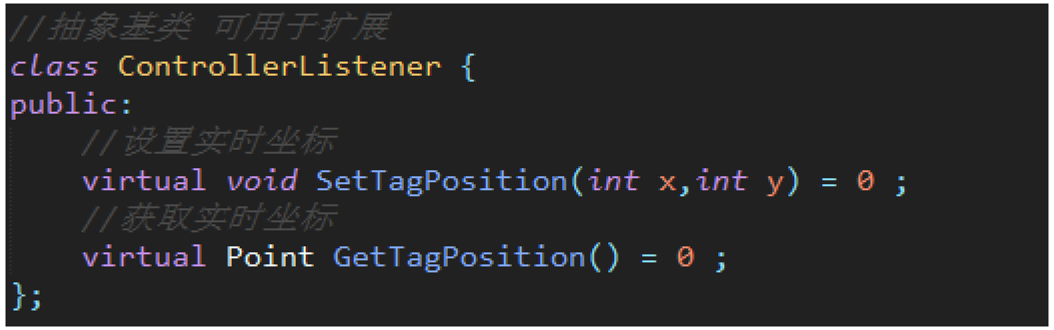


```
void FloatController::update(float delta) {
    if(m_controller_listener == nullptr) {
        return ;
    }

    //当下位置
    Point current_pos = m_controller_listener->GetTagPosition();
    //移动玩家
    current_pos += Point(m_speed_x,m_speed_y);
    m_controller_listener->SetTagPosition(current_pos.x,current_pos.y);
}
```

Design Pattern

- 策略模式

ControllerListener - Umbrella: 简单工厂模式

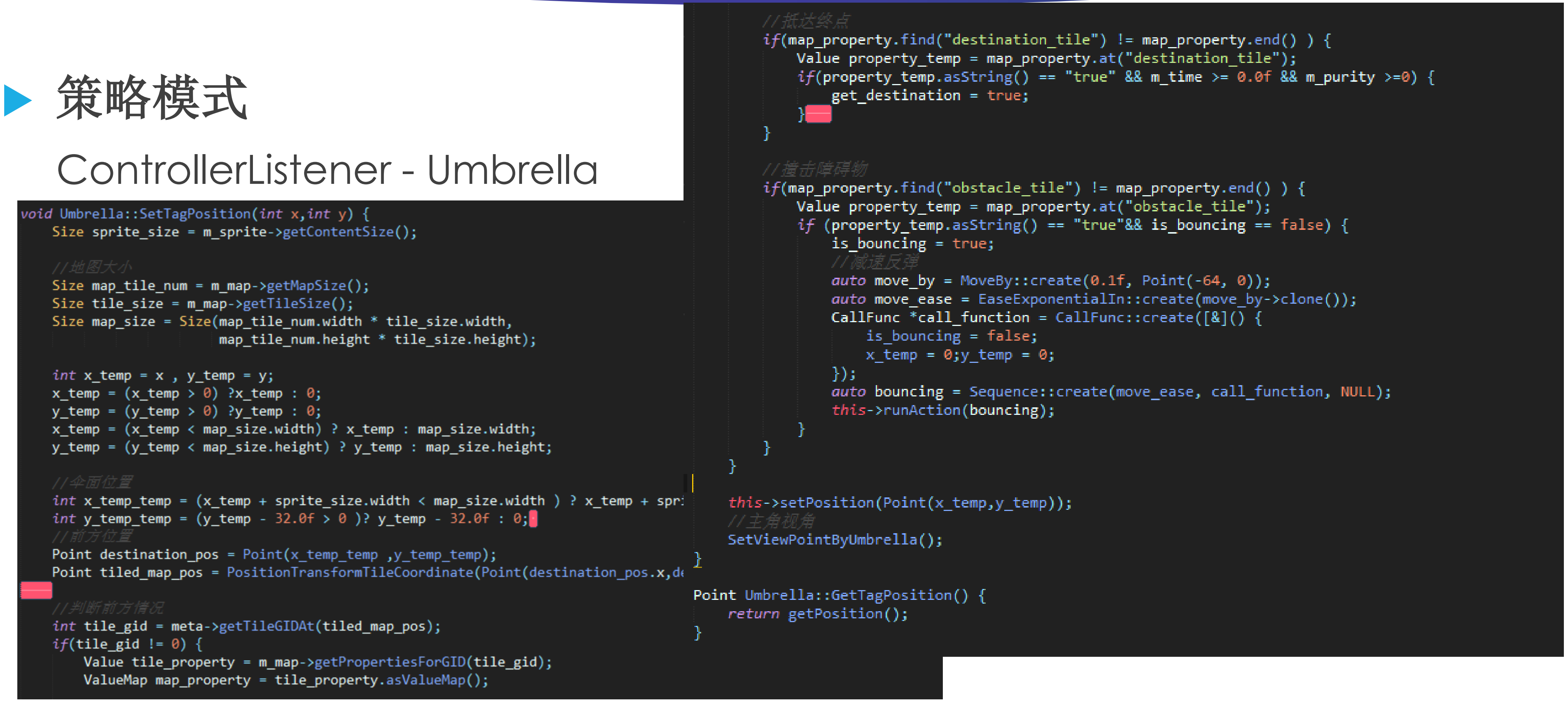


```
//抽象基类 可用于扩展
class ControllerListener {
public:
    //设置实时坐标
    virtual void SetTagPosition(int x,int y) = 0 ;
    //获取实时坐标
    virtual Point GetTagPosition() = 0 ;
};
```

Design Pattern

- 策略模式

ControllerListener - Umbrella



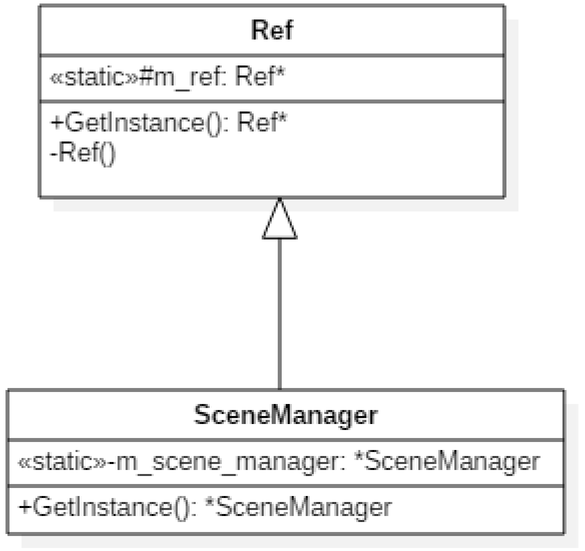
```
void Umbrella::SetTagPosition(int x,int y) {
    Size sprite_size = m_sprite->getContentSize();

    //地图大小
    Size map_tile_num = m_map->getMapSize();
    Size tile_size = m_map->getTileSize();
    Size map_size = Size(map_tile_num.width * tile_size.width,
                         map_tile_num.height * tile_size.height);

    int x_temp = x , y_temp = y;
    x_temp = (x_temp > 0) ?x_temp : 0;
    y_temp = (y_temp > 0) ?y_temp : 0;
    x_temp = (x_temp < map_size.width) ? x_temp : map_size.width;
    y_temp = (y_temp < map_size.height) ? y_temp : map_size.height;

    //伞面位置
    int x_temp_temp = (x_temp + sprite_size.width < map_size.width ) ? x_temp + spri
    int y_temp_temp = (y_temp - 32.0f > 0 )? y_temp - 32.0f : 0;
    //前方位置
    Point destination_pos = Point(x_temp_temp ,y_temp_temp);
    Point tiled_map_pos = PositionTransformTileCoordinate(Point(destination_pos.x,de

    //判断前方情况
    int tile_gid = meta->getTileGIDAt(tiled_map_pos);
    if(tile_gid != 0) {
        Value tile_property = m_map->getPropertiesForGID(tile_gid);
        ValueMap map_property = tile_property.asValueMap();
```

```
        //抵达终点
        if(map_property.find("destination_tile") != map_property.end() ) {
            Value property_temp = map_property.at("destination_tile");
            if(property_temp.asString() == "true" && m_time >= 0.0f && m_purity >=0) {
                get_destination = true;
            }
        }

        //撞击障碍物
        if(map_property.find("obstacle_tile") != map_property.end() ) {
            Value property_temp = map_property.at("obstacle_tile");
            if (property_temp.asString() == "true"&& is_bouncing == false) {
                is_bouncing = true;
                //减速反弹
                auto move_by = MoveBy::create(0.1f, Point(-64, 0));
                auto move_ease = EaseExponentialIn::create(move_by->clone());
                CallFunc *call_function = CallFunc::create([&]() {
                    is_bouncing = false;
                    x_temp = 0;y_temp = 0;
                });
                auto bouncing = Sequence::create(move_ease, call_function, NULL);
                this->runAction(bouncing);
            }
        }
    }

    this->setPosition(Point(x_temp,y_temp));
    //主角视角
    SetViewPointByUmbrella();
}

Point Umbrella::GetTagPosition() {
    return getPosition();
}
```

Design Pattern

- 单例模式

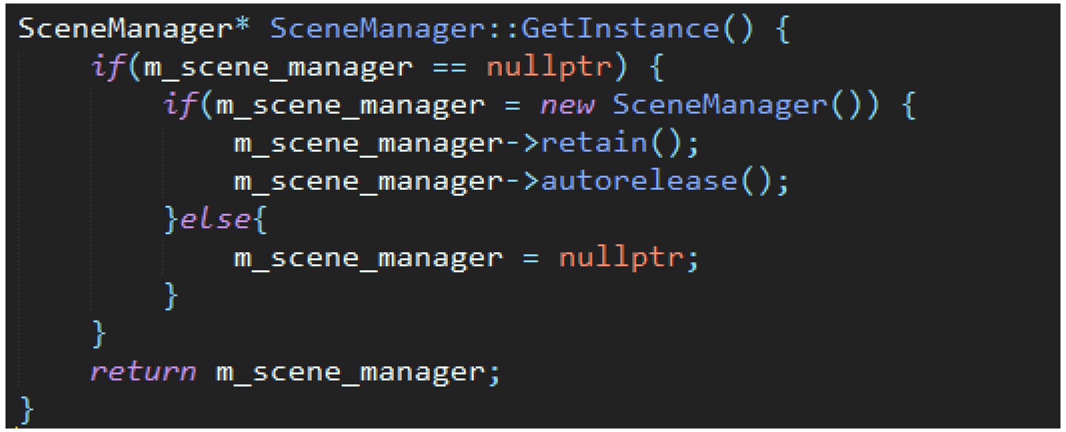
  SceneManager


Ref
«static»#m_ref: Ref*
+GetInstance(): Ref*
-Ref()
SceneManager
«static»-m_scene_manager: *SceneManager
+GetInstance(): *SceneManager

Design Pattern

- 单例模式

SceneManager::GetInstance()



```
SceneManager* SceneManager::GetInstance() {
    if(m_scene_manager == nullptr) {
        if(m_scene_manager = new SceneManager()) {
            m_scene_manager->retain();
            m_scene_manager->autorelease();
        }else{
            m_scene_manager = nullptr;
        }
    }
    return m_scene_manager;
}
```

Demo

- **Github:**

https://github.com/1452712/DP-Project/

Q&A

Thank You !