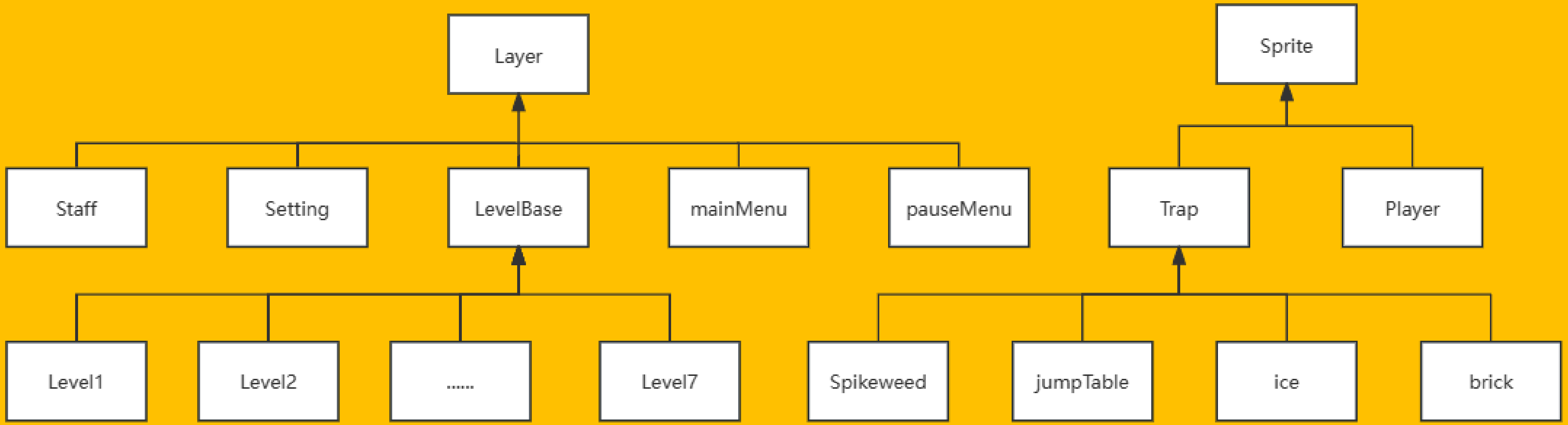
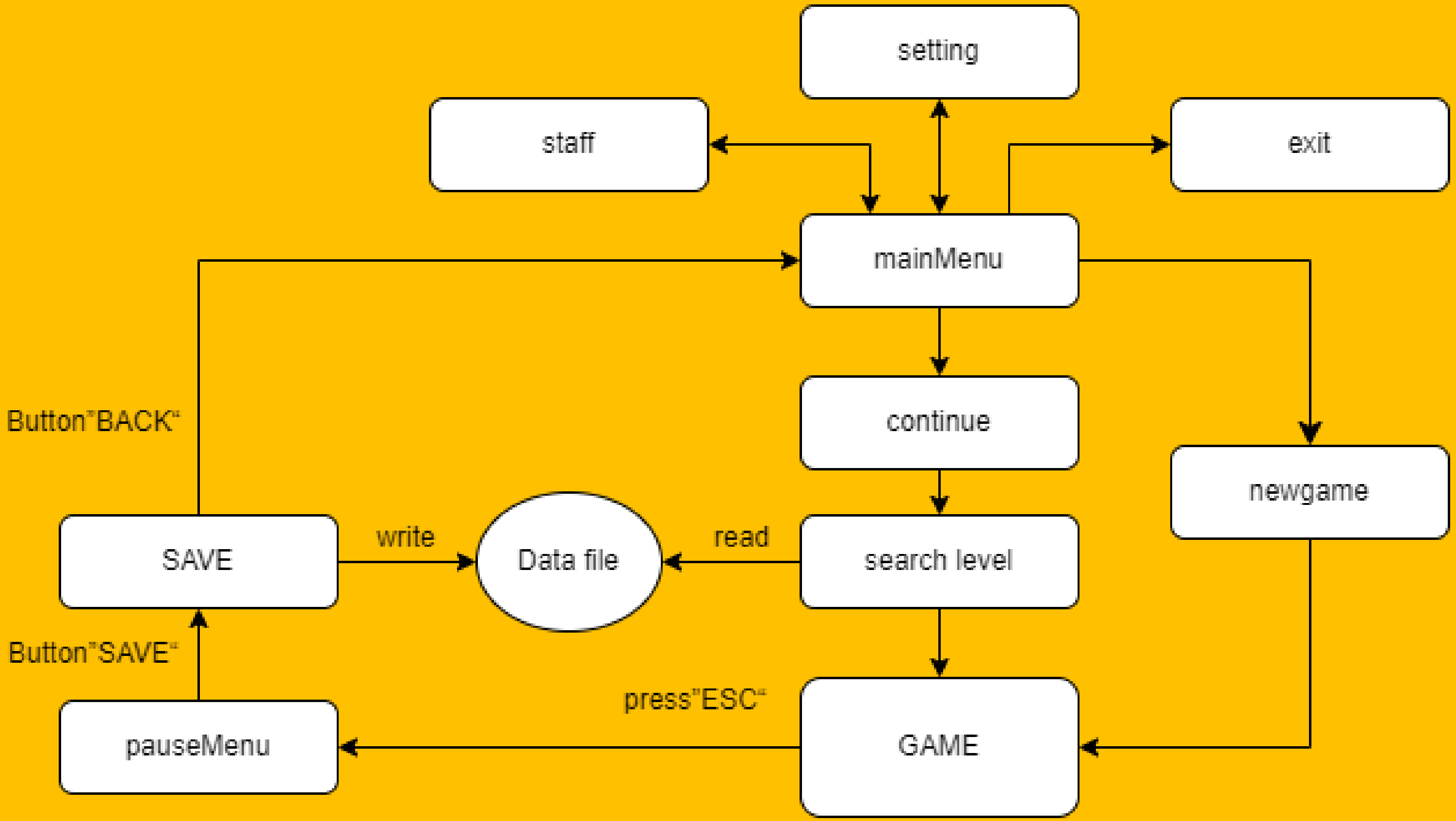
Celeste final report

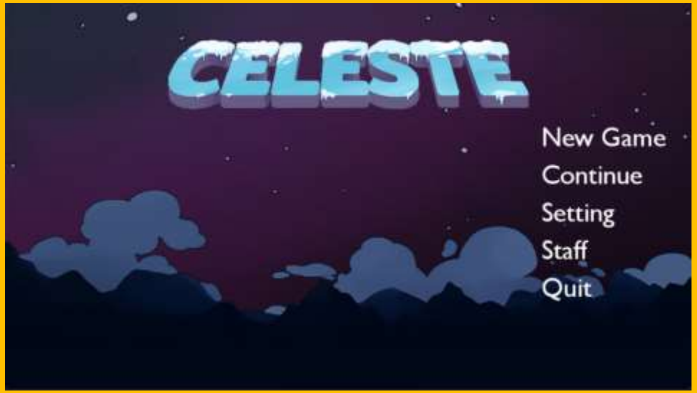
霍君安 龚思嘉 周宇哲

Class diagram UML



Game flowchart

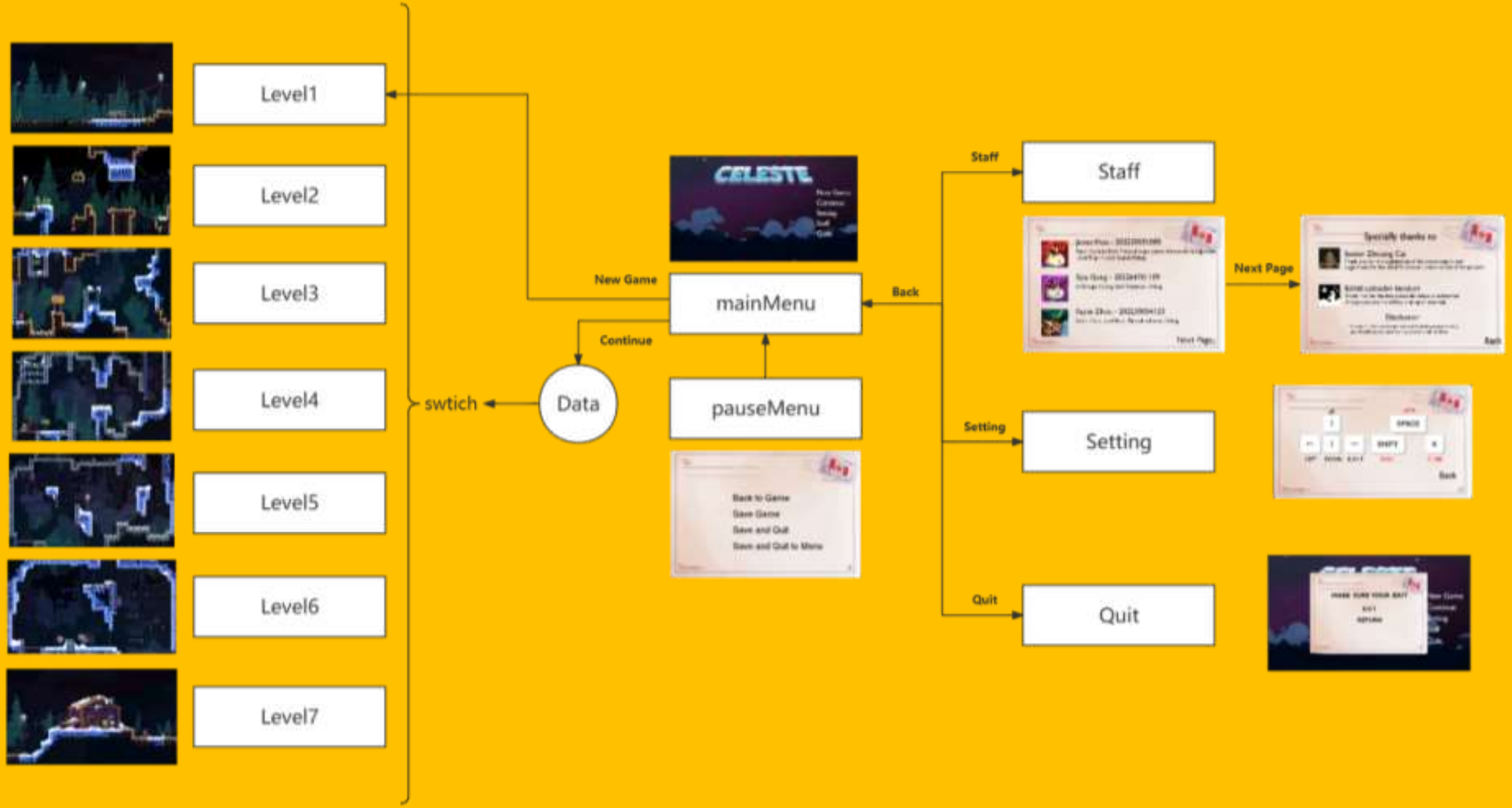




Main scene

Game scene

Flow chat


Level1
Level2
Level3
Level4
Level5
Level6
Level7
swtich
Data
New Game
Continue
mainMenu
pauseMenu
Back
Staff
Next Page
Setting
Quit
CELESTE
Back to Game
Save Game
Save and Quit
Save and Quit to Menu

核心函数

组成部分

Scene 1

Main Menu
Scene and
Other Scene

Level 2

Level Base
Level
End layer

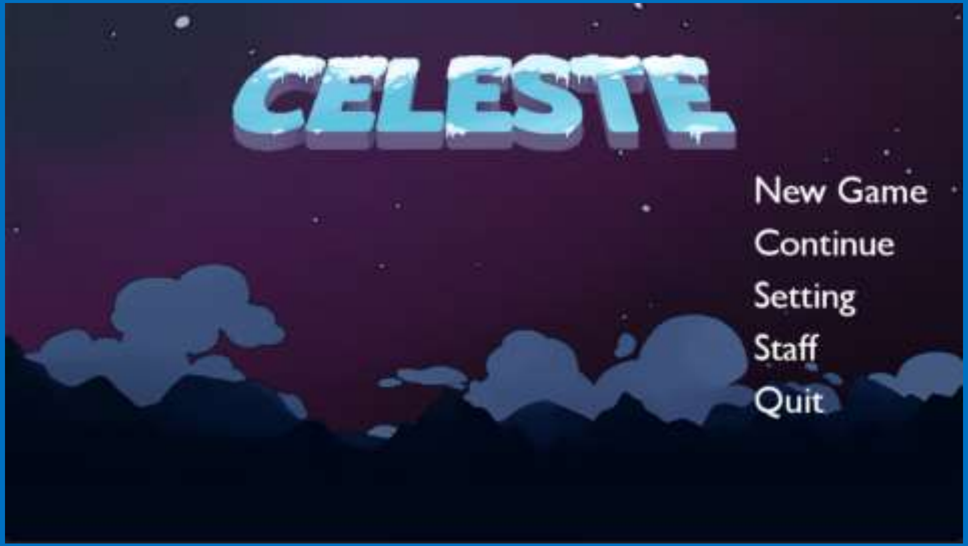
Trap 3

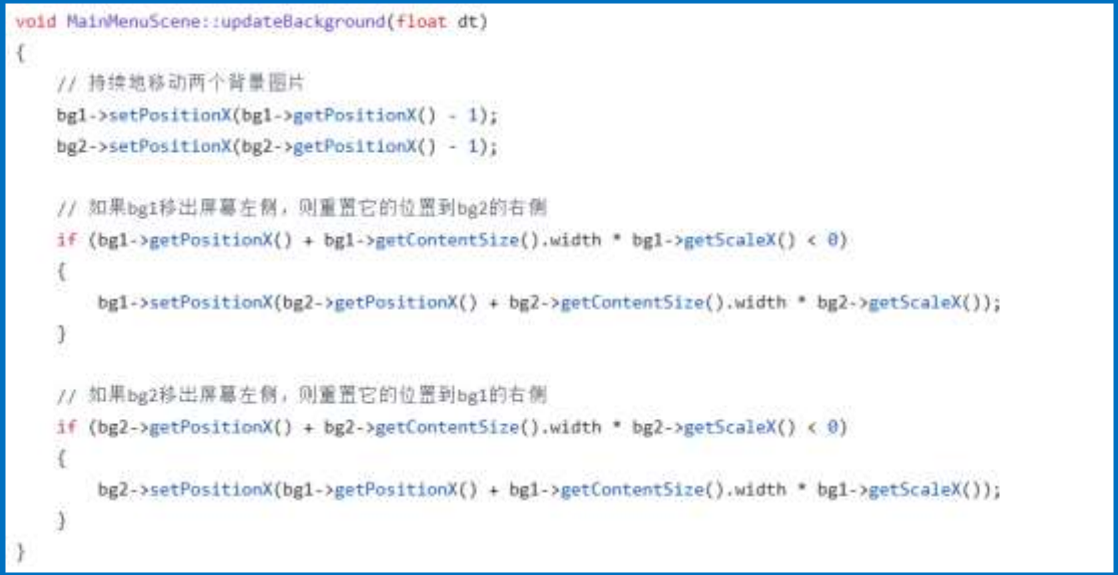
Four kinds
of traps

Player 4

Only
important
functions

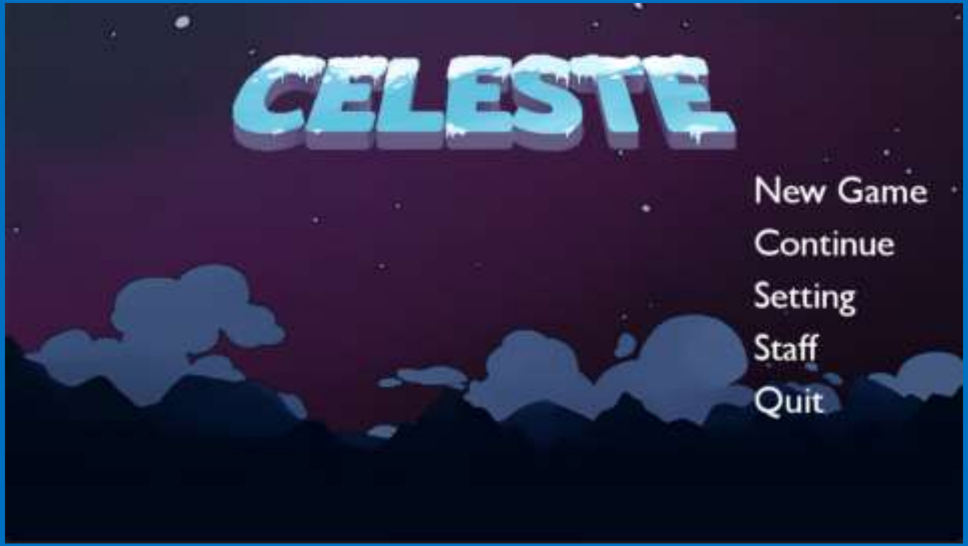
Main Menu Scene

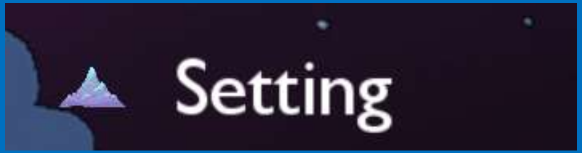


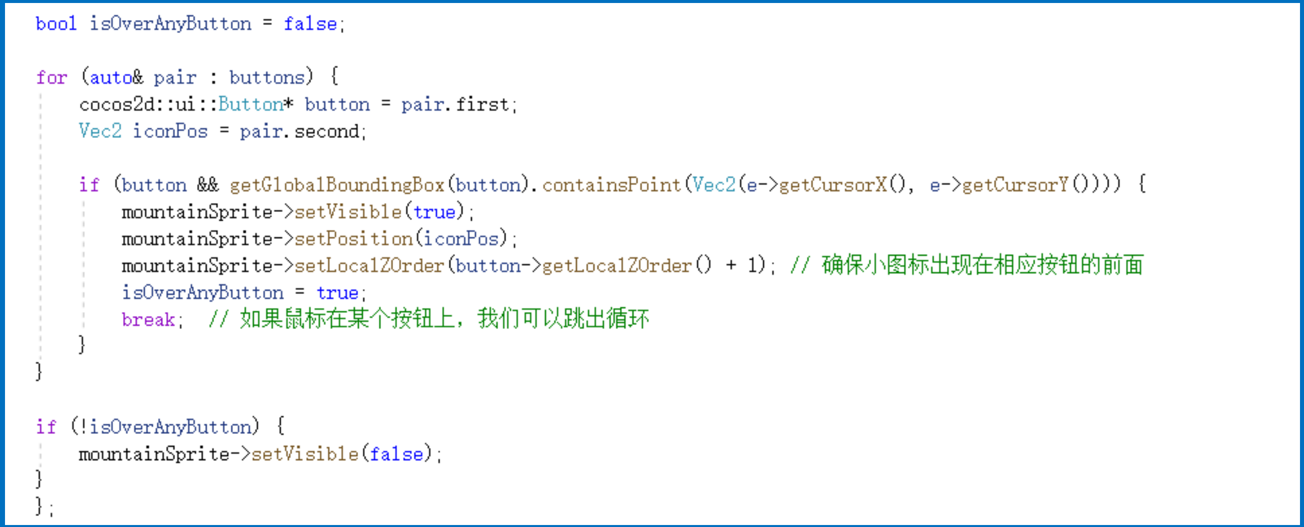


```
void MainMenuScene::updateBackground(float dt)
{
    // 持续地移动两个背景图片
    bg1->setPositionX(bg1->getPositionX() - 1);
    bg2->setPositionX(bg2->getPositionX() - 1);

    // 如果bg1移出屏幕左侧，则重置它的位置到bg2的右侧
    if (bg1->getPositionX() + bg1->getContentSize().width * bg1->getScaleX() < 0)
    {
        bg1->setPositionX(bg2->getPositionX() + bg2->getContentSize().width * bg2->getScaleX());
    }

    // 如果bg2移出屏幕左侧，则重置它的位置到bg1的右侧
    if (bg2->getPositionX() + bg2->getContentSize().width * bg2->getScaleX() < 0)
    {
        bg2->setPositionX(bg1->getPositionX() + bg1->getContentSize().width * bg1->getScaleX());
    }
}
```

Main Menu Scene

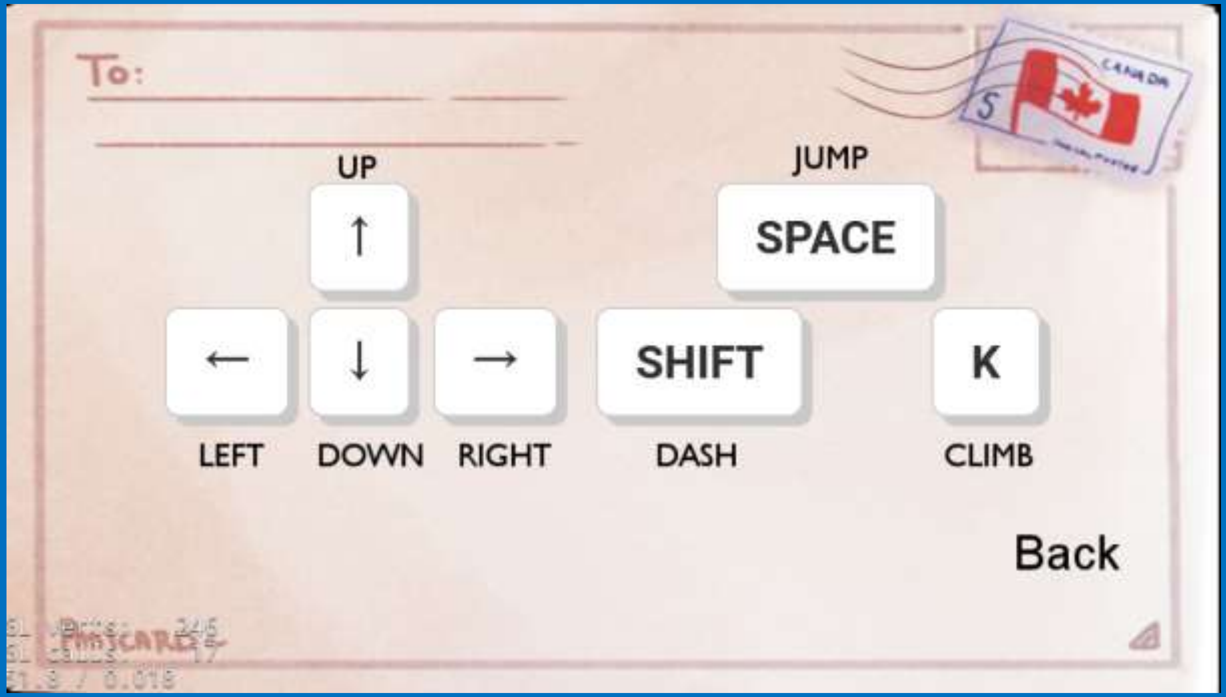






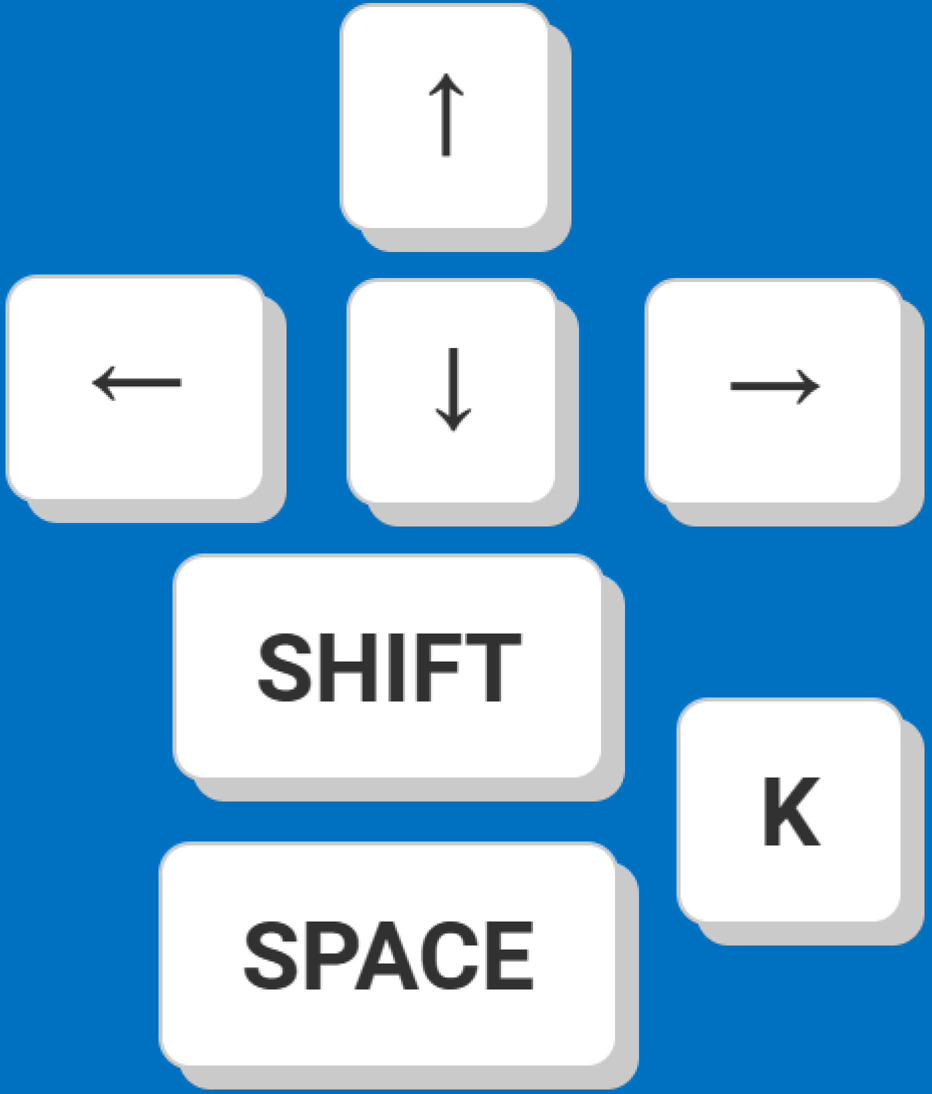
```
bool isOverAnyButton = false;

for (auto& pair : buttons) {
    cocos2d::ui::Button* button = pair.first;
    Vec2 iconPos = pair.second;

    if (button && getGlobalBoundingBox(button).containsPoint(Vec2(e->getCursorX(), e->getCursorY()))) {
        mountainSprite->setVisible(true);
        mountainSprite->setPosition(iconPos);
        mountainSprite->setLocalZOrder(button->getLocalZOrder() + 1); // 确保小图标出现在相应按钮的前面
        isOverAnyButton = true;
        break;  // 如果鼠标在某个按钮上，我们可以跳出循环
    }
}

if (!isOverAnyButton) {
    mountainSprite->setVisible(false);
}
};
```

Setting

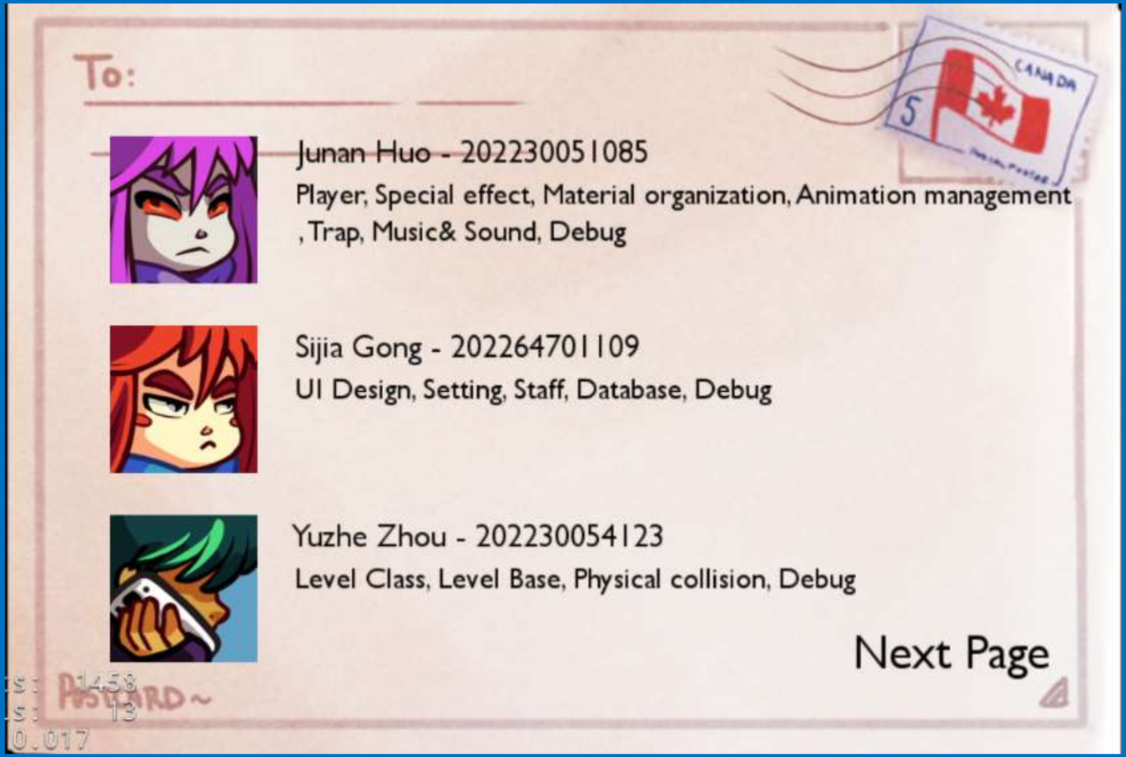

To:

UP ↑
LEFT ←
DOWN ↓
RIGHT →
JUMP SPACE
DASH SHIFT
CLIMB K

Back



Staff



To:

Junan Huo - 202230051085

Player, Special effect, Material organization, Animation management , Trap, Music& Sound, Debug

Sijia Gong - 202264701109

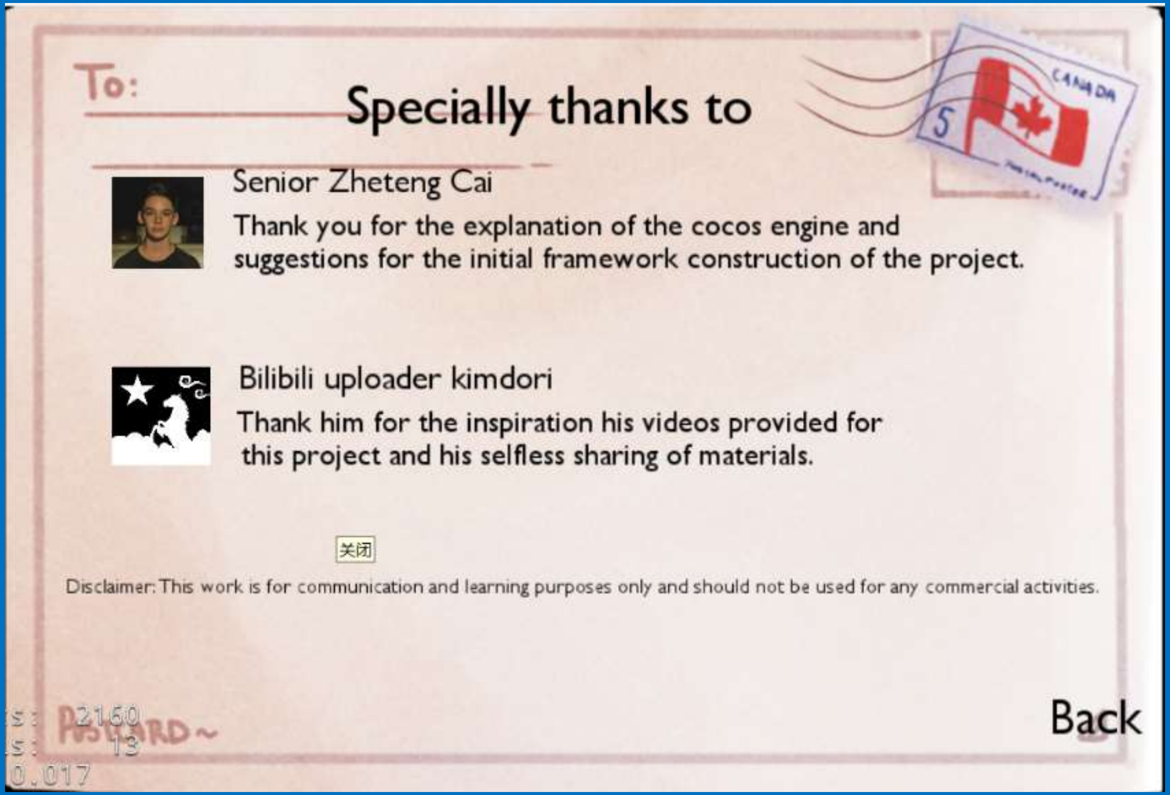
UI Design, Setting, Staff, Database, Debug

Yuzhe Zhou - 202230054123

Level Class, Level Base, Physical collision, Debug

Next Page

Staff



To:

Specially thanks to

Senior Zheteng Cai

Thank you for the explanation of the cocos engine and suggestions for the initial framework construction of the project.

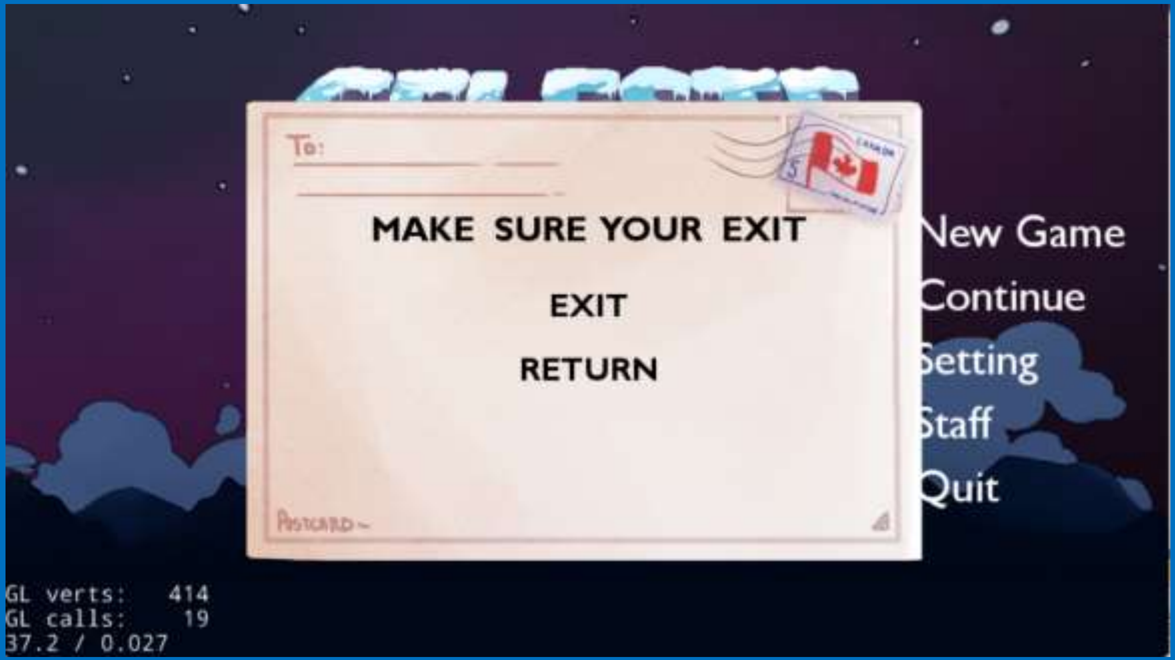
Bilibili uploader kimdori

Thank him for the inspiration his videos provided for this project and his selfless sharing of materials.

关闭

Disclaimer: This work is for communication and learning purposes only and should not be used for any commercial activities.

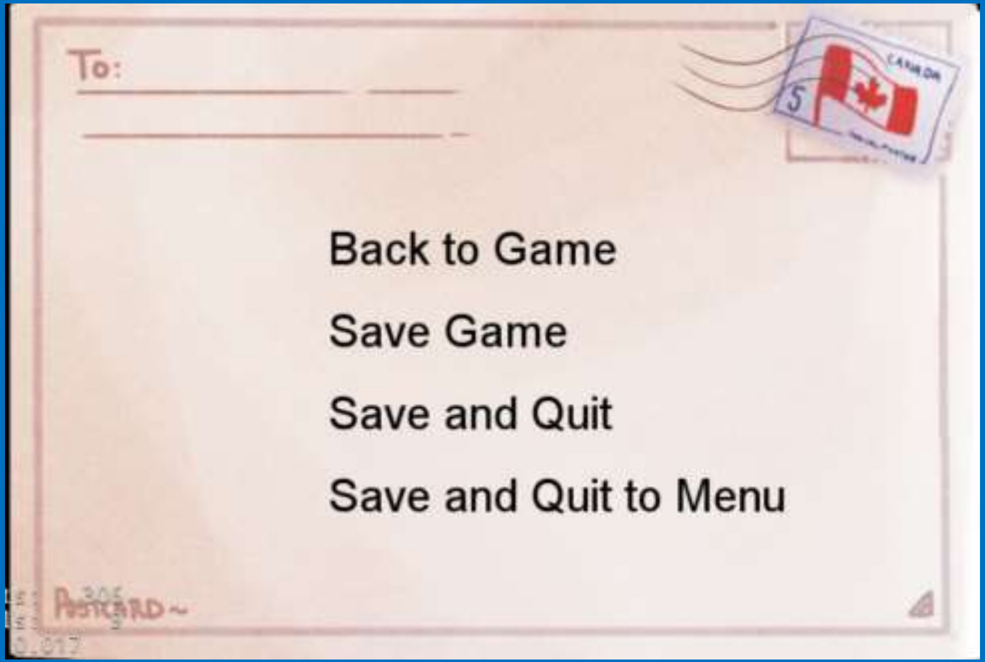
Quit

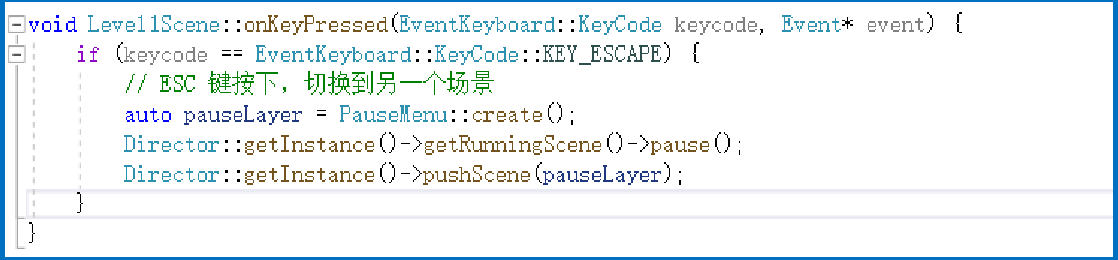


```
GL verts:    414
GL calls:     19
37.2 / 0.027
```

```
exitDialog->setVisible(false); // 初始时隐藏它
quit->addTouchEventListener([=](Ref* sender, cocos2d::ui::Widget::TouchEventType type) {
    if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
        exitDialog->setVisible(true);
    }
```

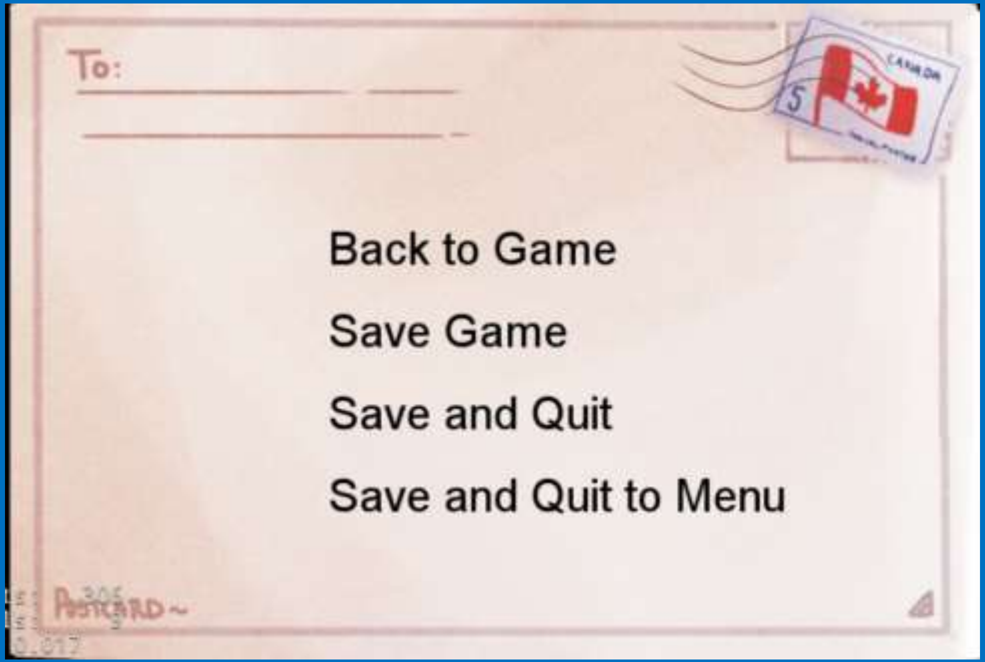
Pause Menu

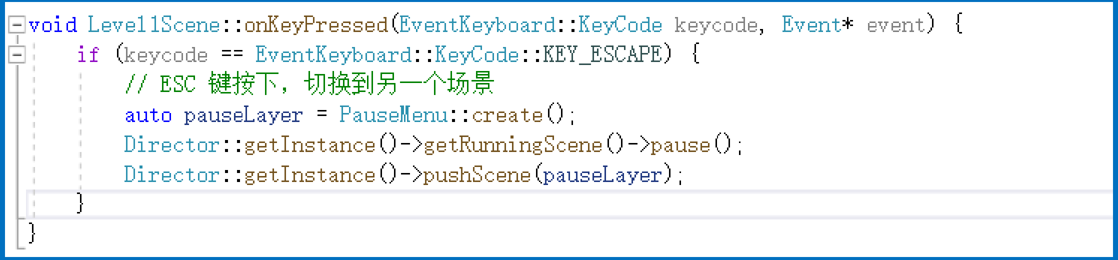




```
void Level1Scene::onKeyPressed(EventKeyboard::KeyCode keycode, Event* event) {
    if (keycode == EventKeyboard::KeyCode::KEY_ESCAPE) {
        // ESC 键按下，切换到另一个场景
        auto pauseLayer = PauseMenu::create();
        Director::getInstance()->getRunningScene()->pause();
        Director::getInstance()->pushScene(pauseLayer);
    }
}
```

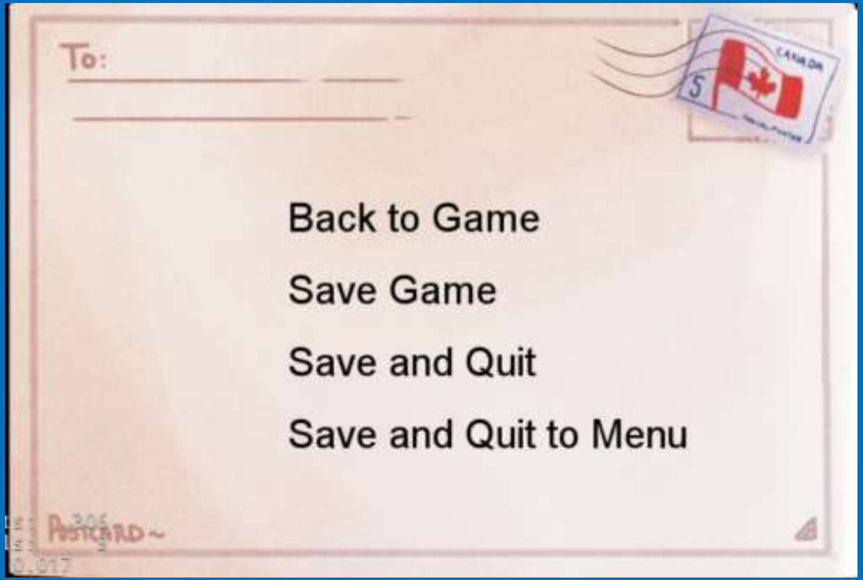
Pause Menu

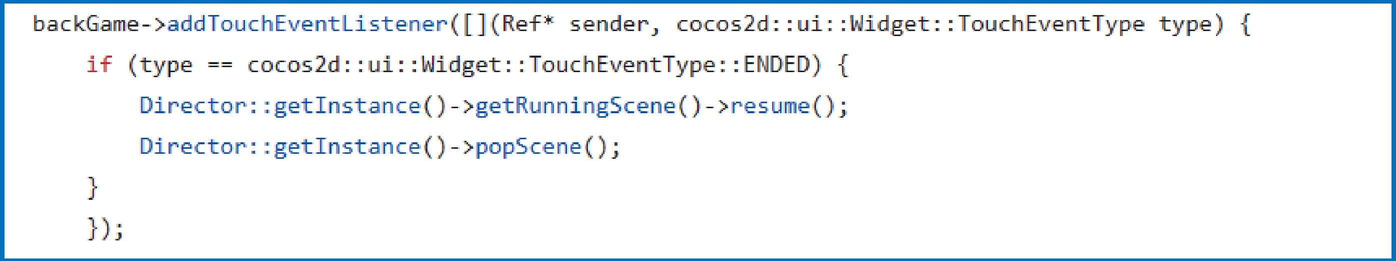




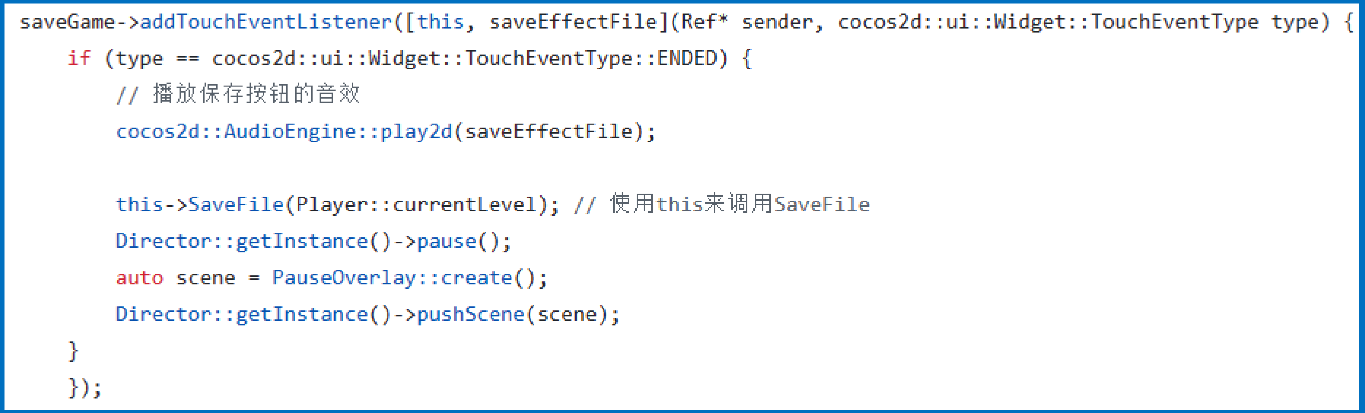
```
void Level1Scene::onKeyPressed(EventKeyboard::KeyCode keycode, Event* event) {
    if (keycode == EventKeyboard::KeyCode::KEY_ESCAPE) {
        // ESC 键按下，切换到另一个场景
        auto pauseLayer = PauseMenu::create();
        Director::getInstance()->getRunningScene()->pause();
        Director::getInstance()->pushScene(pauseLayer);
    }
}
```

Pause Menu(data save)

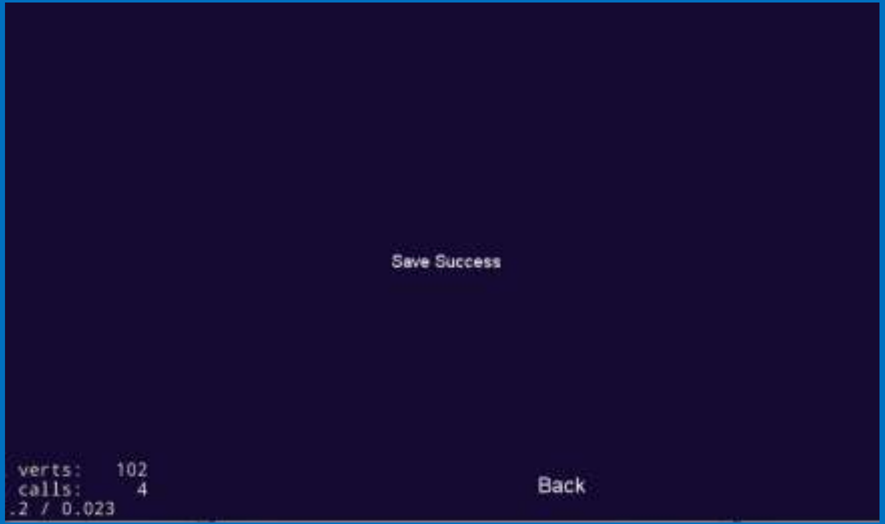




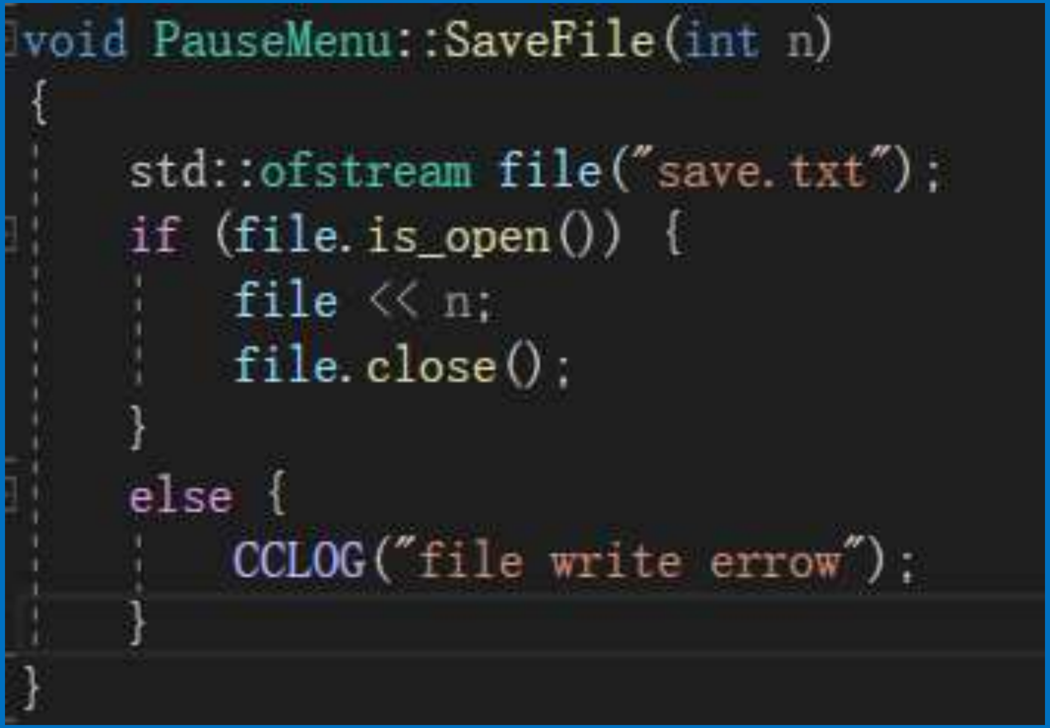
```
backGame->addTouchEventListener([](Ref* sender, cocos2d::ui::Widget::TouchEventType type) {
    if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
        Director::getInstance()->getRunningScene()->resume();
        Director::getInstance()->popScene();
    }
    });
```



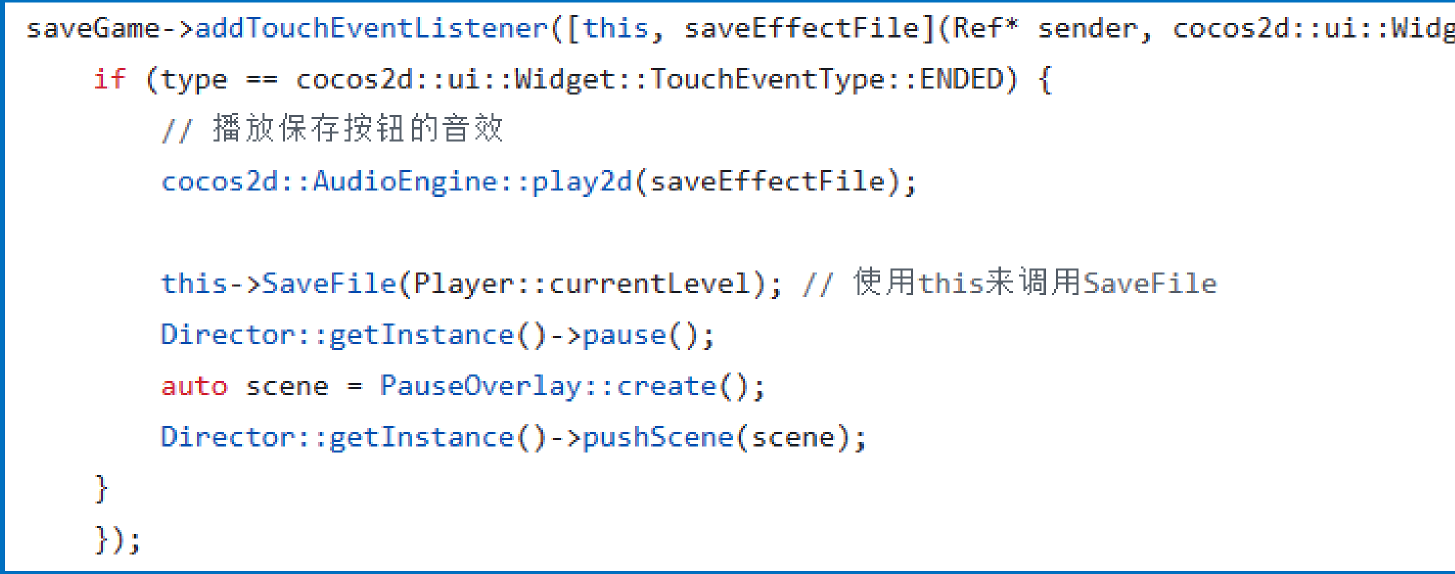
```
saveGame->addTouchEventListener([this, saveEffectFile](Ref* sender, cocos2d::ui::Widget::TouchEventType type) {
    if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
        // 播放保存按钮的音效
        cocos2d::AudioEngine::play2d(saveEffectFile);

        this->SaveFile(Player::currentLevel); // 使用this来调用SaveFile
        Director::getInstance()->pause();
        auto scene = PauseOverlay::create();
        Director::getInstance()->pushScene(scene);
    }
    });
```



Pause Menu(data save)

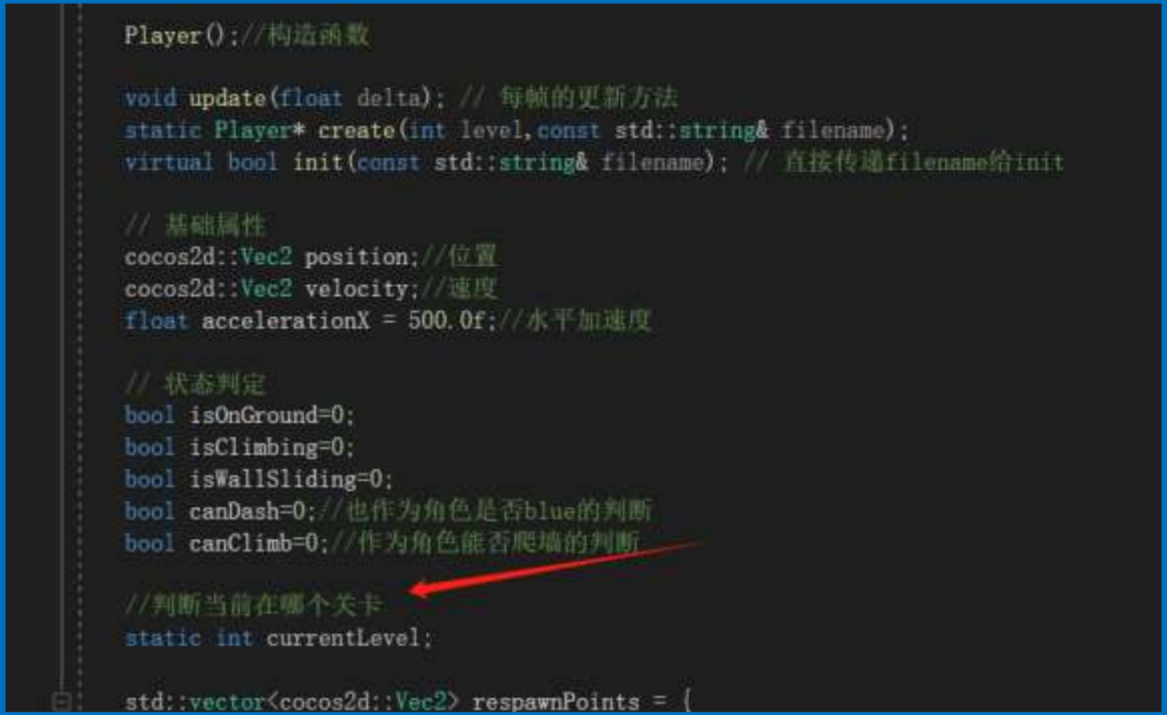


```cpp
void PauseMenu::SaveFile(int n)
{
    std::ofstream file("save.txt");
    if (file.is_open()) {
        file << n;
        file.close();
    }
    else {
        CCLOG("file write errow");
    }
}
```

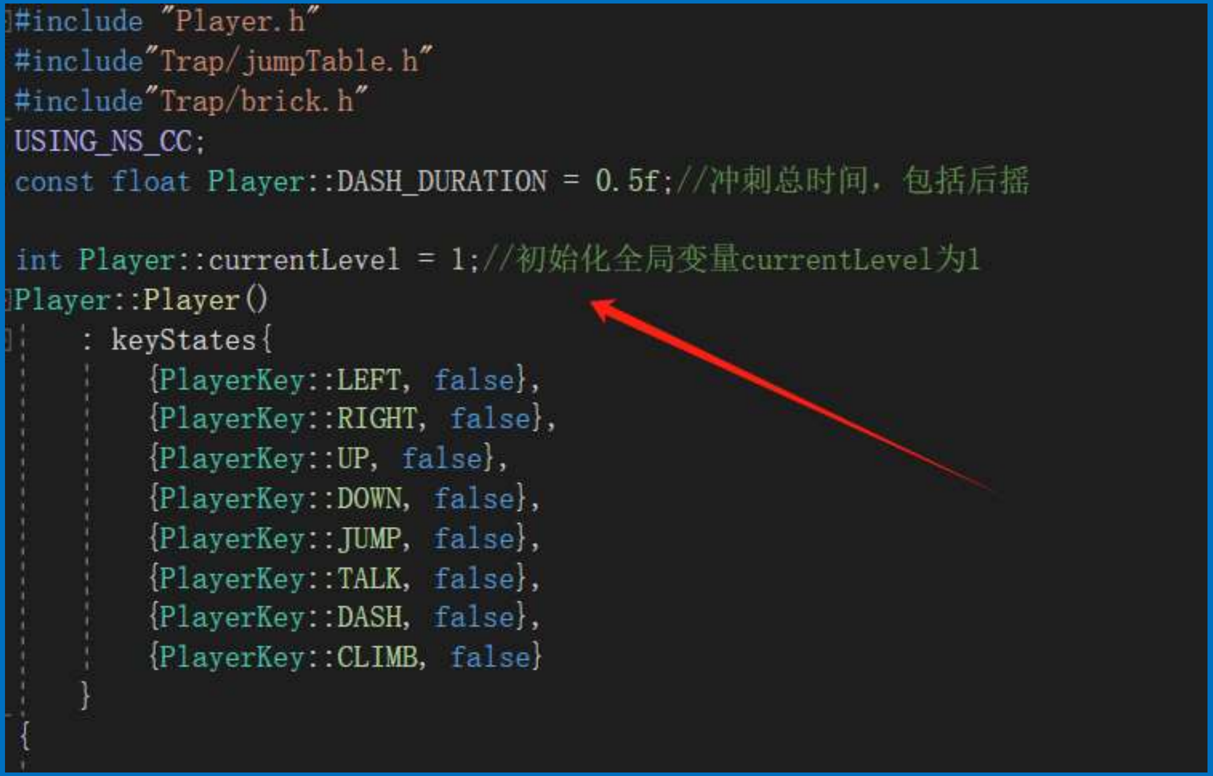


```cpp
saveGame->addTouchEventListener([this, saveEffectFile](Ref* sender, cocos2d::ui::Widg
    if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
        // 播放保存按钮的音效
        cocos2d::AudioEngine::play2d(saveEffectFile);

        this->SaveFile(Player::currentLevel); // 使用this来调用SaveFile
        Director::getInstance()->pause();
        auto scene = PauseOverlay::create();
        Director::getInstance()->pushScene(scene);
    }
    });
```

Pause Menu(data save)

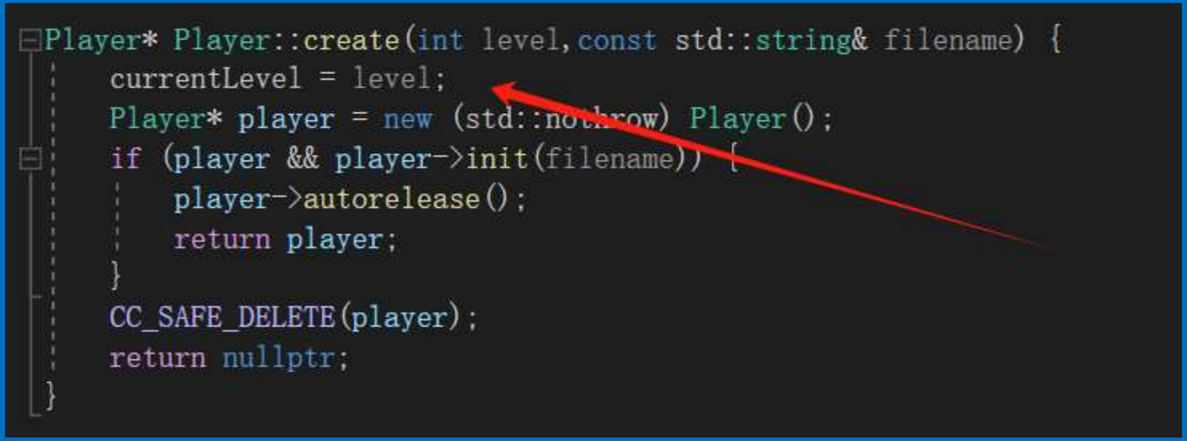


```
Player();//构造函数

void update(float delta); // 每帧的更新方法
static Player* create(int level,const std::string& filename);
virtual bool init(const std::string& filename); // 直接传递filename给init

// 基础属性
cocos2d::Vec2 position;//位置
cocos2d::Vec2 velocity;//速度
float accelerationX = 500.0f;//水平加速度

// 状态判定
bool isOnGround=0;
bool isClimbing=0;
bool isWallSliding=0;
bool canDash=0;//也作为角色是否blue的判断
bool canClimb=0;//作为角色能否爬墙的判断

//判断当前在哪个关卡
static int currentLevel;

std::vector<cocos2d::Vec2> respawnPoints = {
```

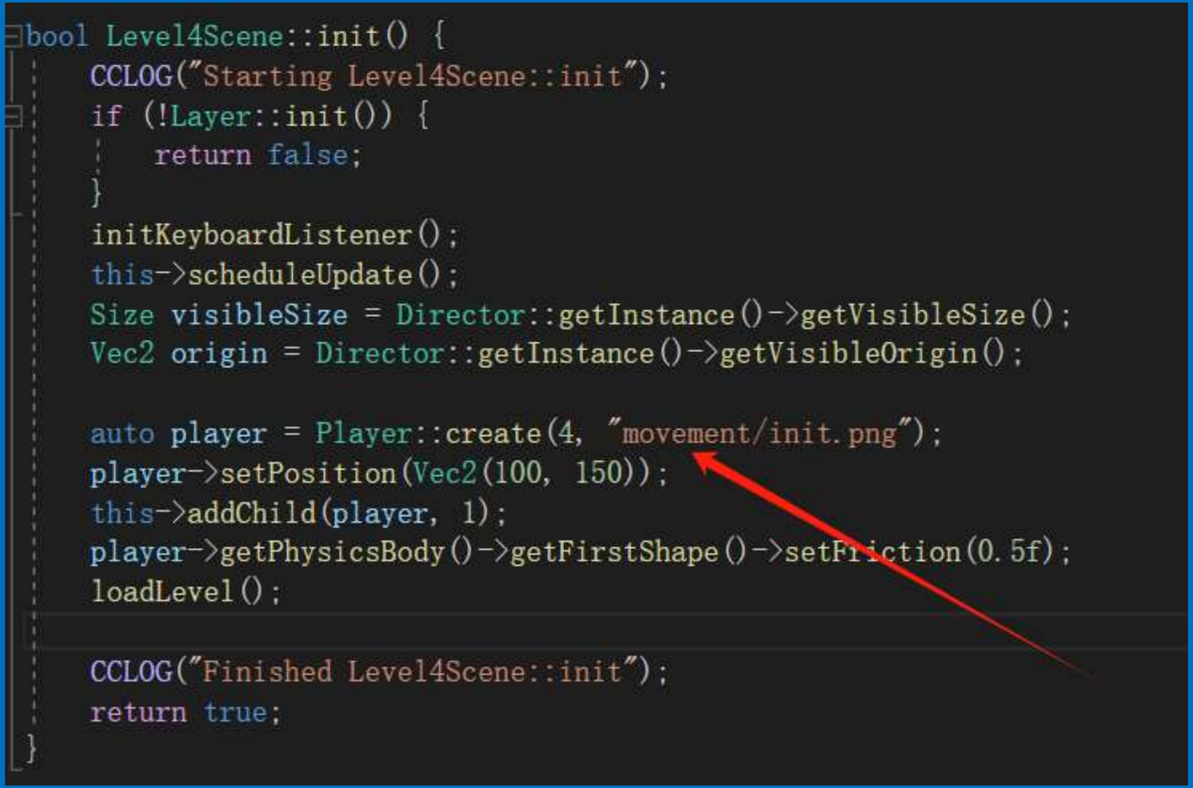


```
#include "Player.h"
#include"Trap/jumpTable.h"
#include"Trap/brick.h"
USING_NS_CC;
const float Player::DASH_DURATION = 0.5f;//冲刺总时间，包括后摇

int Player::currentLevel = 1;//初始化全局变量currentLevel为1
Player::Player()
    : keyStates{
        {PlayerKey::LEFT, false},
        {PlayerKey::RIGHT, false},
        {PlayerKey::UP, false},
        {PlayerKey::DOWN, false},
        {PlayerKey::JUMP, false},
        {PlayerKey::TALK, false},
        {PlayerKey::DASH, false},
        {PlayerKey::CLIMB, false}
    }
{
```

Pause Menu(data save)

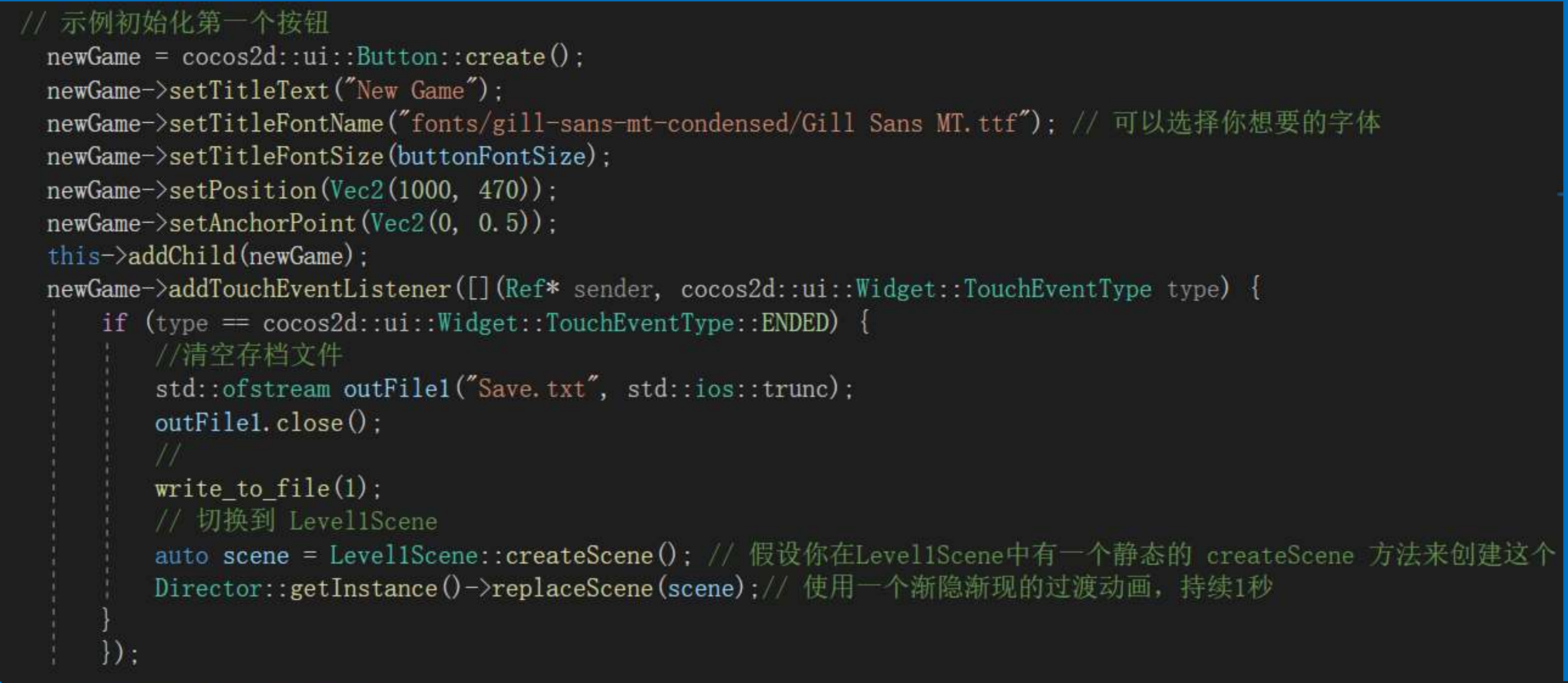


```cpp
Player* Player::create(int level,const std::string& filename) {
    currentLevel = level;
    Player* player = new (std::nothrow) Player();
    if (player && player->init(filename)) {
        player->autorelease();
        return player;
    }
    CC_SAFE_DELETE(player);
    return nullptr;
}
```

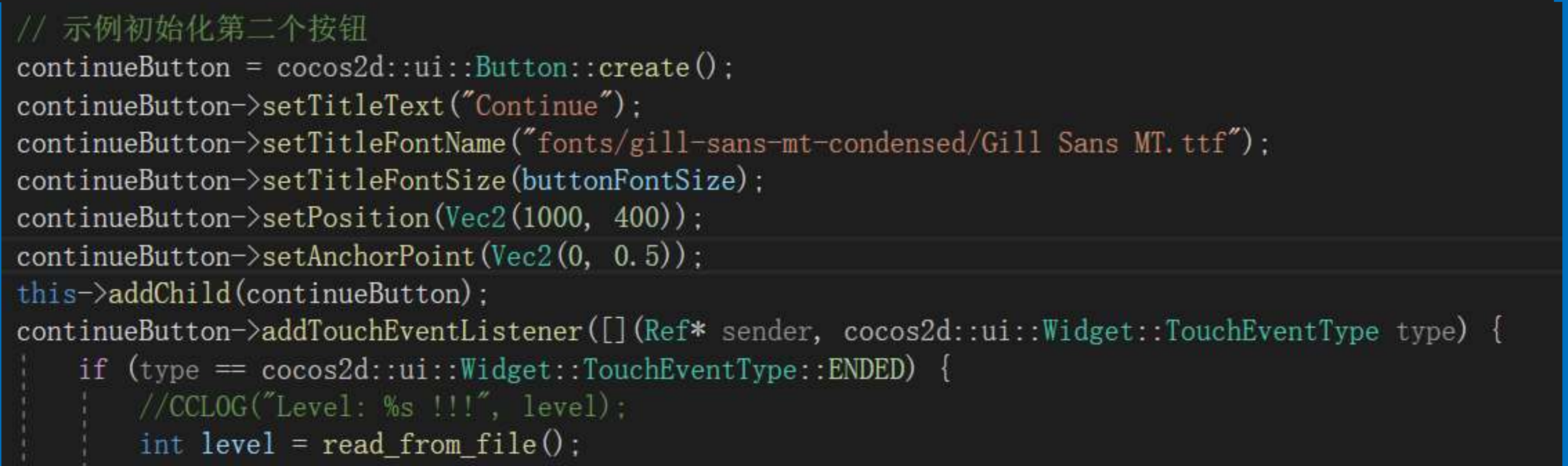


```cpp
bool Level4Scene::init() {
    CCLOG("Starting Level4Scene::init");
    if (!Layer::init()) {
        return false;
    }
    initKeyboardListener();
    this->scheduleUpdate();
    Size visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();

    auto player = Player::create(4, "movement/init.png");
    player->setPosition(Vec2(100, 150));
    this->addChild(player, 1);
    player->getPhysicsBody()->getFirstShape()->setFriction(0.5f);
    loadLevel();

    CCLOG("Finished Level4Scene::init");
    return true;
}
```

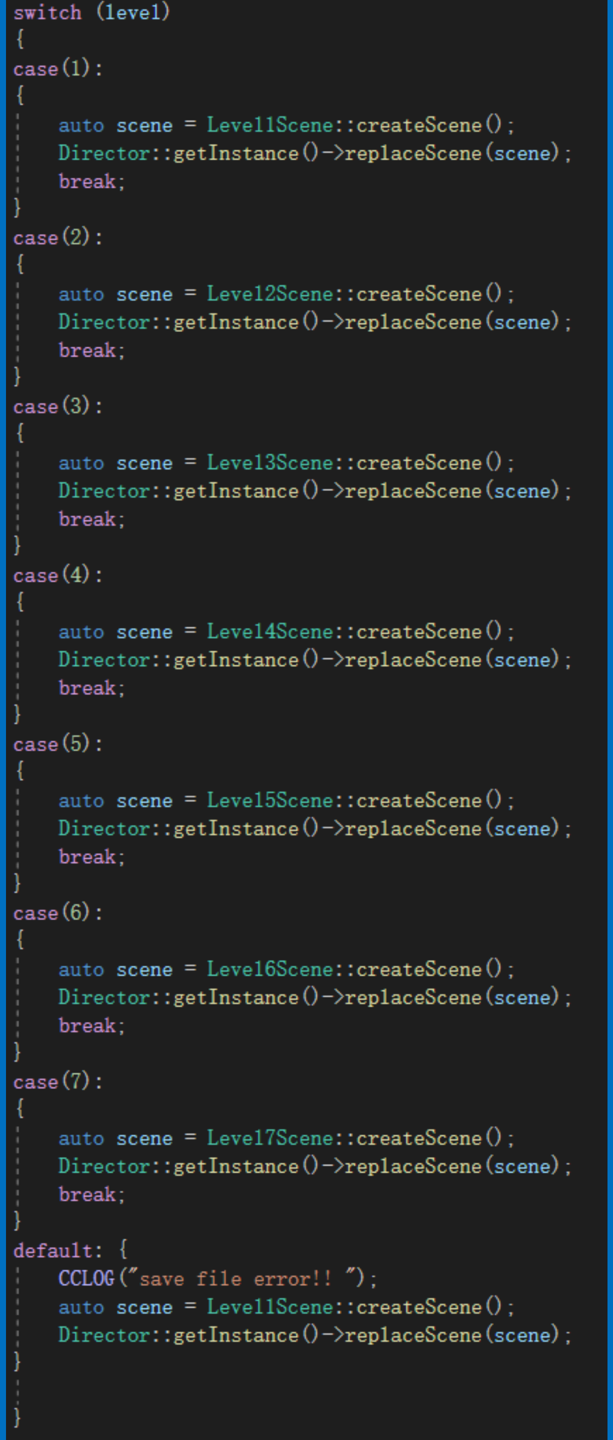
New game continue (Data read)



```
// 示例初始化第一个按钮
  newGame = cocos2d::ui::Button::create();
  newGame->setTitleText("New Game");
  newGame->setTitleFontName("fonts/gill-sans-mt-condensed/Gill Sans MT.ttf"); // 可以选择你想要的字体
  newGame->setTitleFontSize(buttonFontSize);
  newGame->setPosition(Vec2(1000, 470));
  newGame->setAnchorPoint(Vec2(0, 0.5));
  this->addChild(newGame);
  newGame->addTouchEventListener([](Ref* sender, cocos2d::ui::Widget::TouchEventType type) {
      if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
          //清空存档文件
          std::ofstream outFile1("Save.txt", std::ios::trunc);
          outFile1.close();
          //
          write_to_file(1);
          // 切换到 Level1Scene
          auto scene = Level1Scene::createScene(); // 假设你在Level1Scene中有一个静态的 createScene 方法来创建这个
          Director::getInstance()->replaceScene(scene);// 使用一个渐隐渐现的过渡动画，持续1秒
      }
      });
```

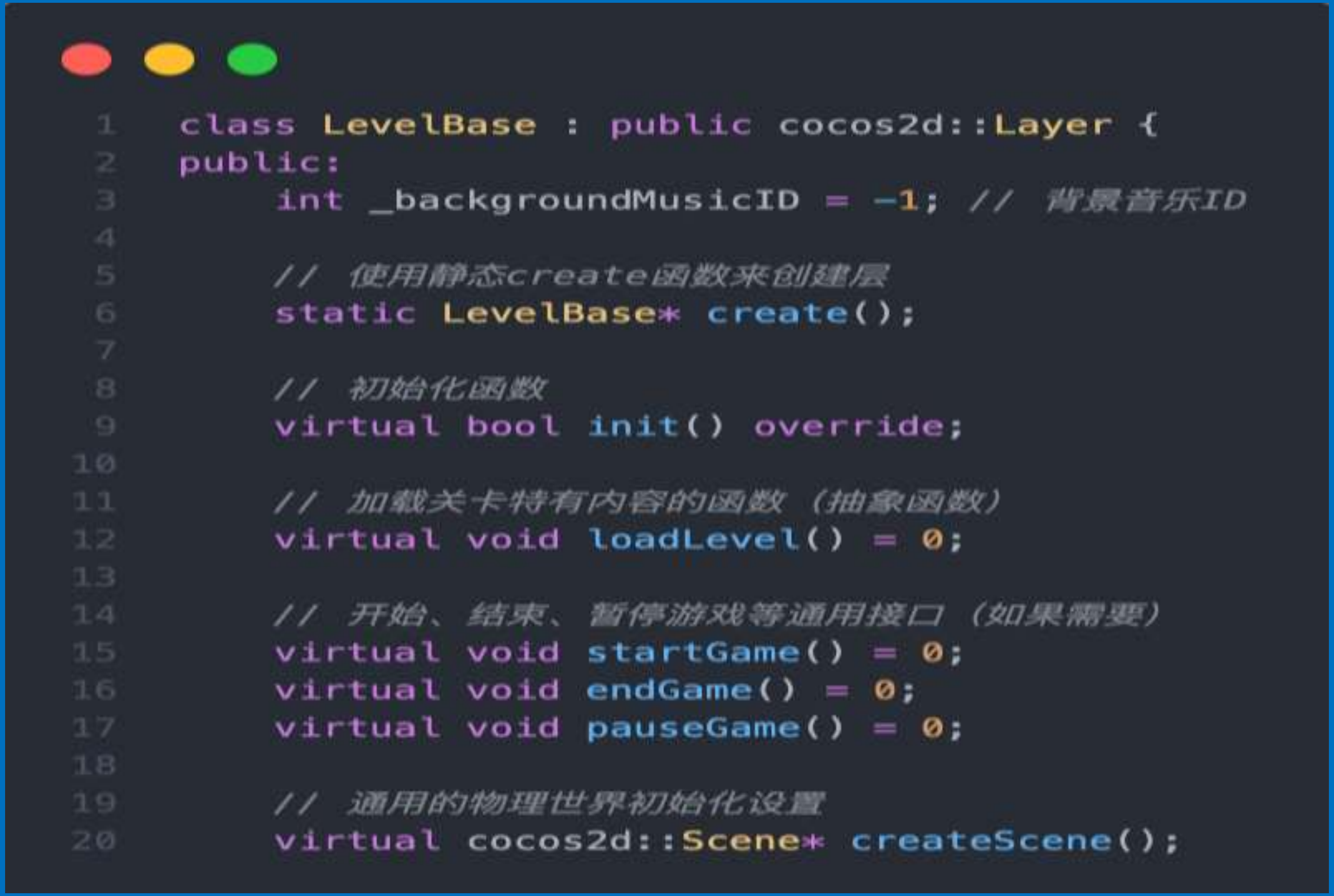


```
// 示例初始化第二个按钮
continueButton = cocos2d::ui::Button::create();
continueButton->setTitleText("Continue");
continueButton->setTitleFontName("fonts/gill-sans-mt-condensed/Gill Sans MT.ttf");
continueButton->setTitleFontSize(buttonFontSize);
continueButton->setPosition(Vec2(1000, 400));
continueButton->setAnchorPoint(Vec2(0, 0.5));
this->addChild(continueButton);
continueButton->addTouchEventListener([](Ref* sender, cocos2d::ui::Widget::TouchEventType type) {
    if (type == cocos2d::ui::Widget::TouchEventType::ENDED) {
        //CCLOG("Level: %s !!!", level);
        int level = read_from_file();
```



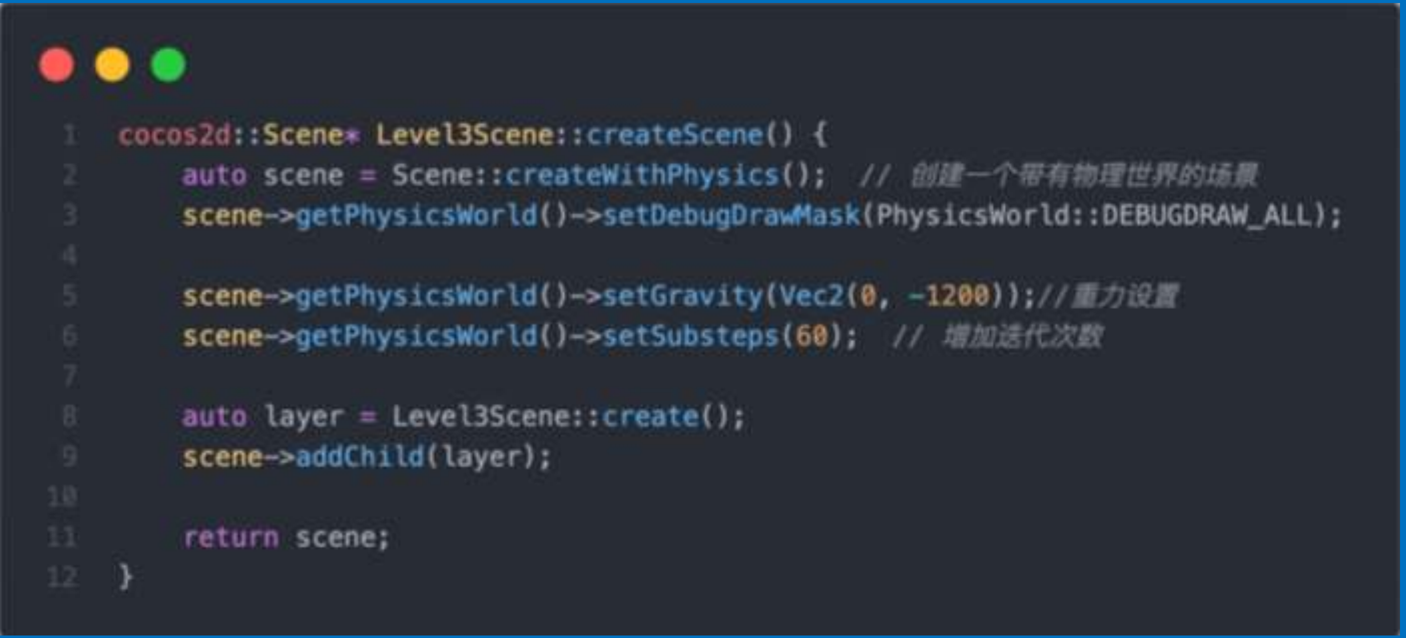
```
switch (level)
{
case(1):
{
    auto scene = Level1Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(2):
{
    auto scene = Level2Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(3):
{
    auto scene = Level3Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(4):
{
    auto scene = Level4Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(5):
{
    auto scene = Level5Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(6):
{
    auto scene = Level6Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
case(7):
{
    auto scene = Level7Scene::createScene();
    Director::getInstance()->replaceScene(scene);
    break;
}
default: {
    CCLOG("save file error!! ");
    auto scene = Level1Scene::createScene();
    Director::getInstance()->replaceScene(scene);
}

}
```

LevelBase Class

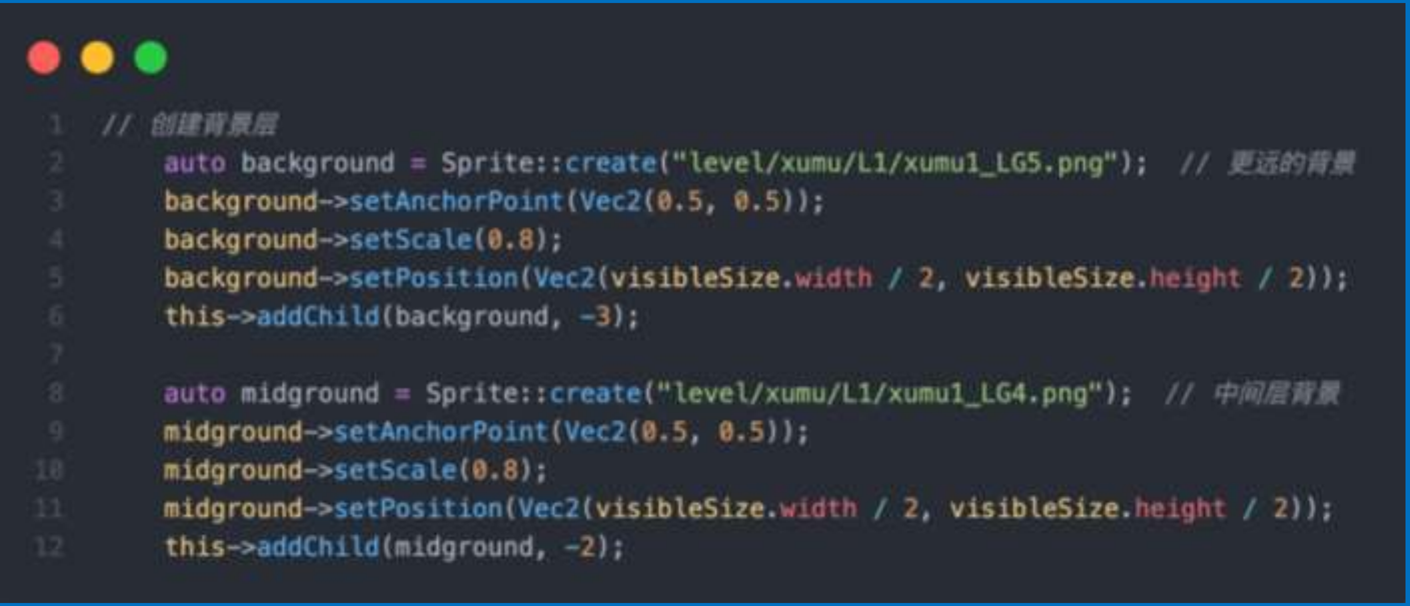


```cpp
class LevelBase : public cocos2d::Layer {
public:
    int _backgroundMusicID = -1; // 背景音乐ID

    // 使用静态create函数来创建层
    static LevelBase* create();

    // 初始化函数
    virtual bool init() override;

    // 加载关卡特有内容的函数（抽象函数）
    virtual void loadLevel() = 0;

    // 开始、结束、暂停游戏等通用接口（如果需要）
    virtual void startGame() = 0;
    virtual void endGame() = 0;
    virtual void pauseGame() = 0;

    // 通用的物理世界初始化设置
    virtual cocos2d::Scene* createScene();
```

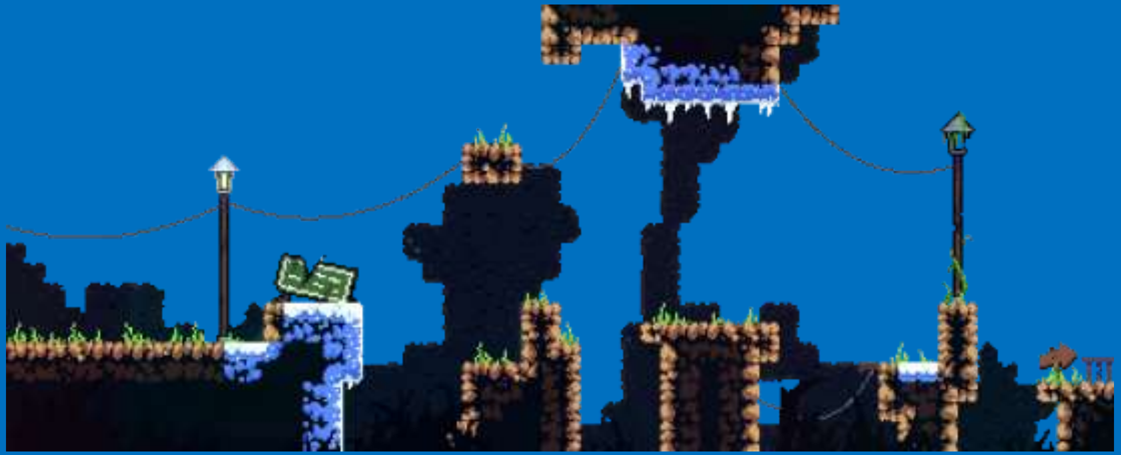
Level Class



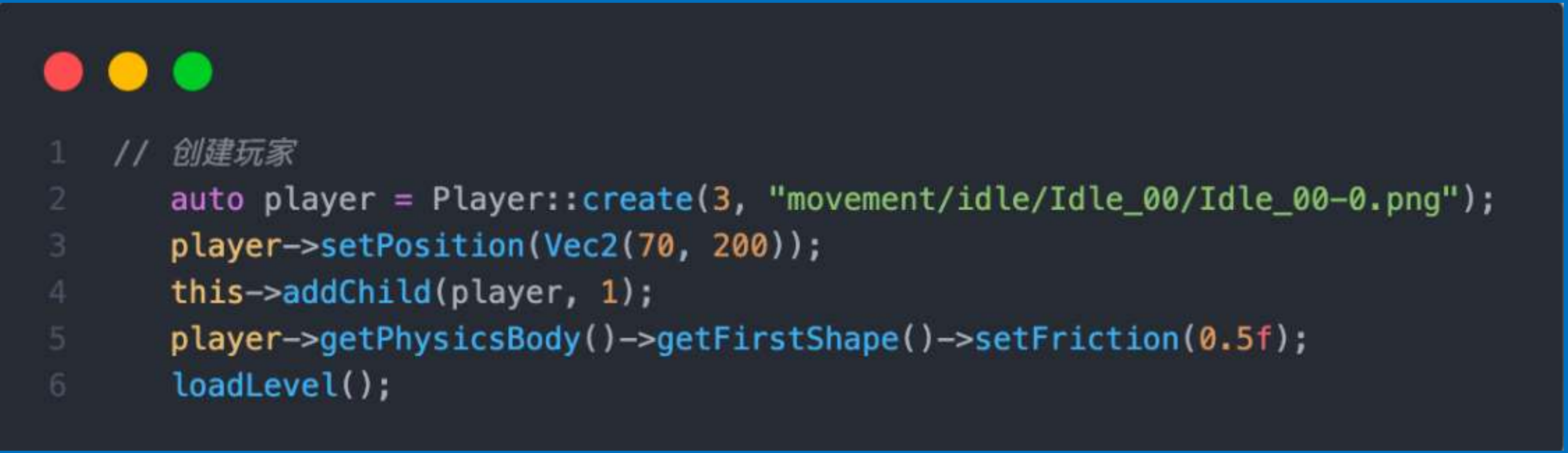
```cpp
cocos2d::Scene* Level3Scene::createScene() {
    auto scene = Scene::createWithPhysics();  // 创建一个带有物理世界的场景
    scene->getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW_ALL);

    scene->getPhysicsWorld()->setGravity(Vec2(0, -1200));//重力设置
    scene->getPhysicsWorld()->setSubsteps(60);  // 增加迭代次数

    auto layer = Level3Scene::create();
    scene->addChild(layer);

    return scene;
}
```



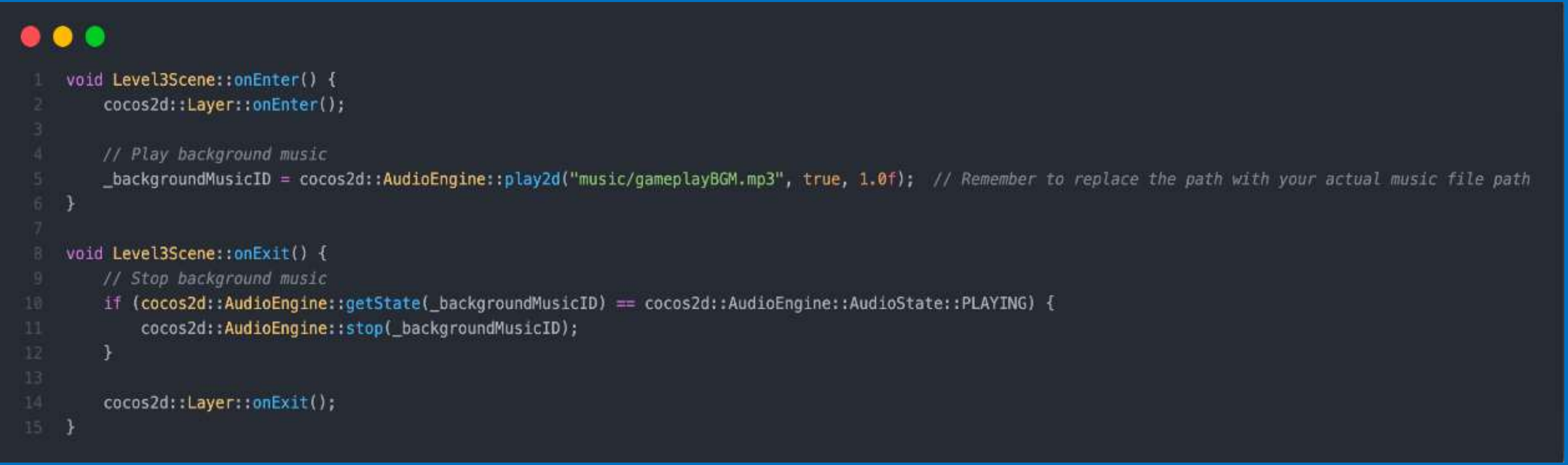
```cpp
// 创建背景层
    auto background = Sprite::create("level/xumu/L1/xumu1_LG5.png");  // 更远的背景
    background->setAnchorPoint(Vec2(0.5, 0.5));
    background->setScale(0.8);
    background->setPosition(Vec2(visibleSize.width / 2, visibleSize.height / 2));
    this->addChild(background, -3);

    auto midground = Sprite::create("level/xumu/L1/xumu1_LG4.png");  // 中间层背景
    midground->setAnchorPoint(Vec2(0.5, 0.5));
    midground->setScale(0.8);
    midground->setPosition(Vec2(visibleSize.width / 2, visibleSize.height / 2));
    this->addChild(midground, -2);
```



Level Class

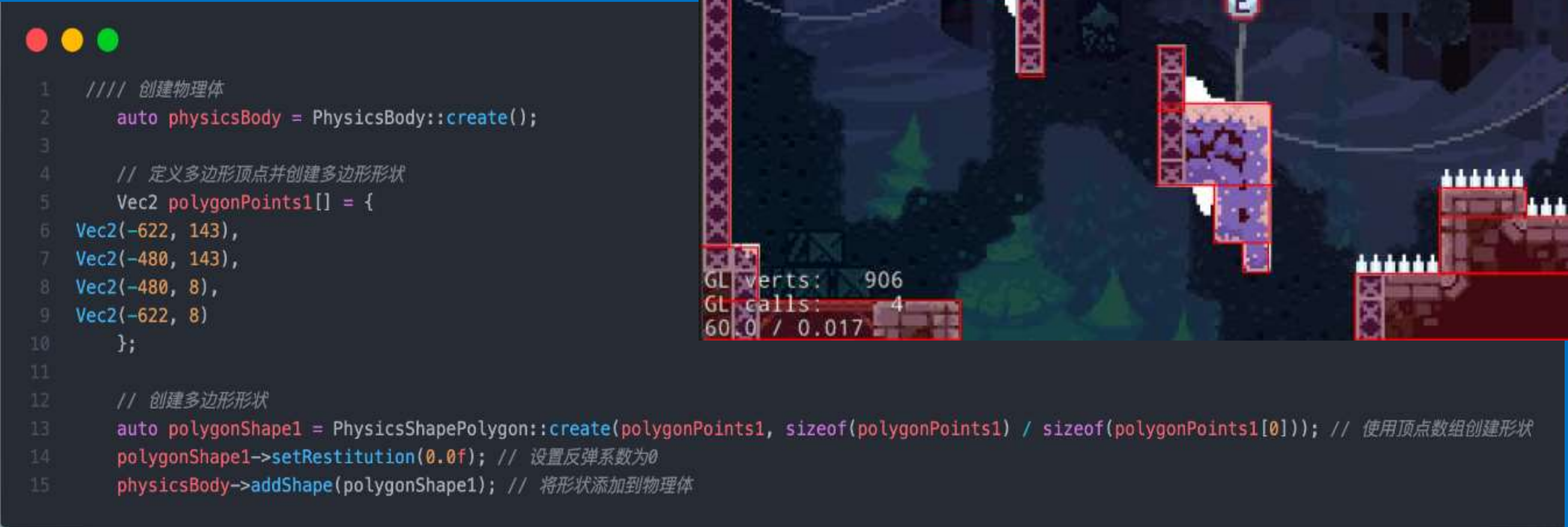


```
// 创建玩家
    auto player = Player::create(3, "movement/idle/Idle_00/Idle_00-0.png");
    player->setPosition(Vec2(70, 200));
    this->addChild(player, 1);
    player->getPhysicsBody()->getFirstShape()->setFriction(0.5f);
    loadLevel();
```

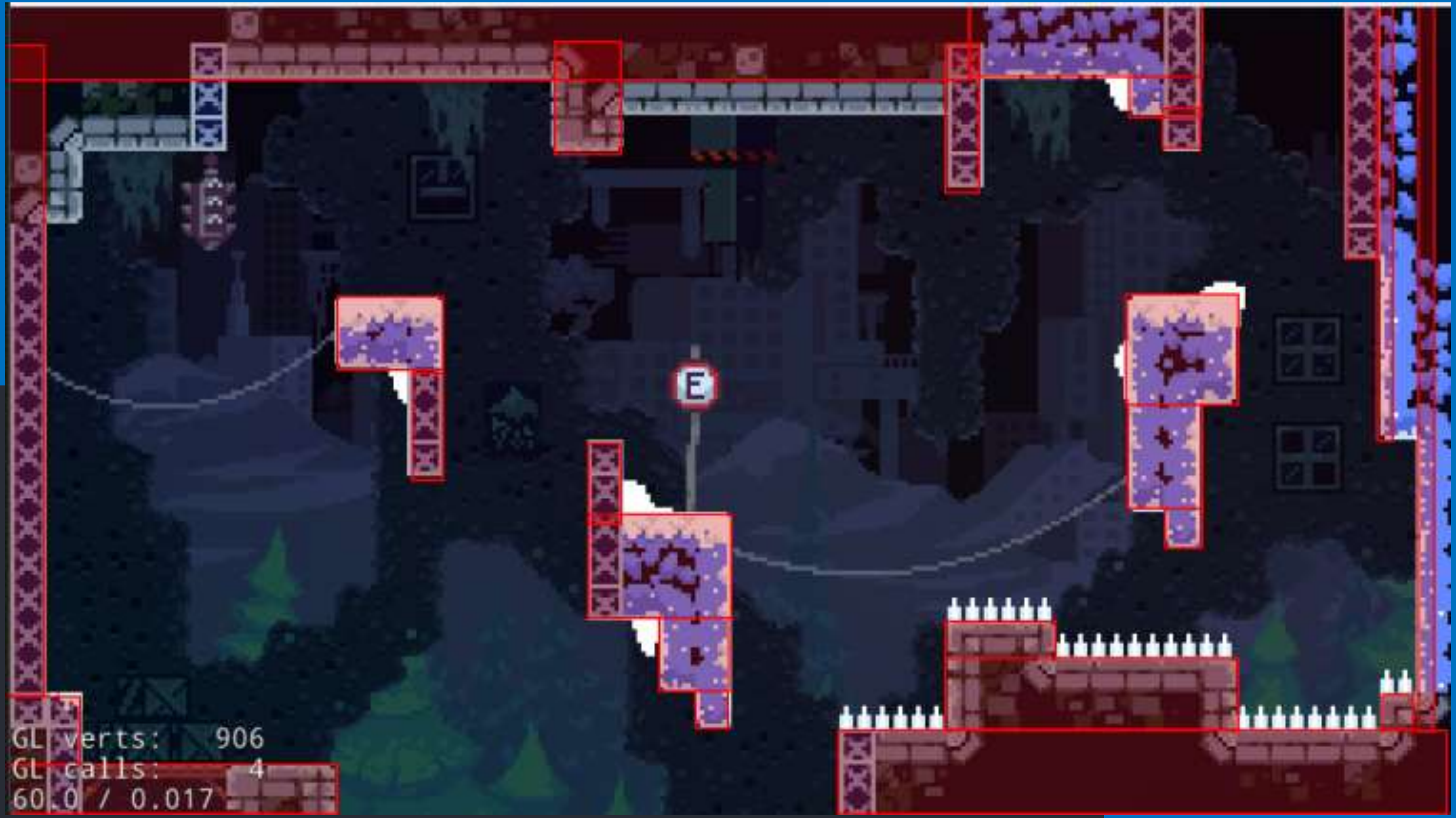


```
void Level3Scene::onEnter() {
    cocos2d::Layer::onEnter();

    // Play background music
    _backgroundMusicID = cocos2d::AudioEngine::play2d("music/gameplayBGM.mp3", true, 1.0f);  // Remember to replace the path with your actual music file path
}

void Level3Scene::onExit() {
    // Stop background music
    if (cocos2d::AudioEngine::getState(_backgroundMusicID) == cocos2d::AudioEngine::AudioState::PLAYING) {
        cocos2d::AudioEngine::stop(_backgroundMusicID);
    }

    cocos2d::Layer::onExit();
}
```

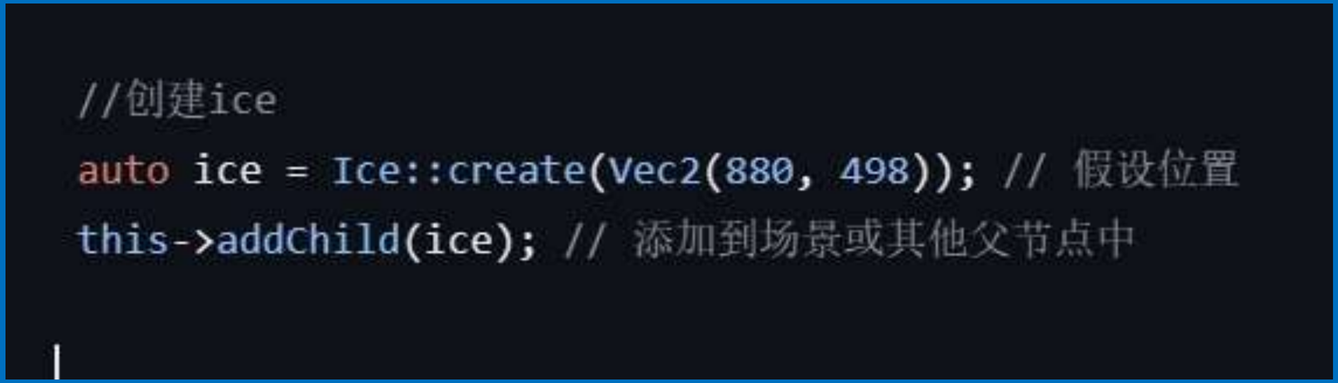
Level Class



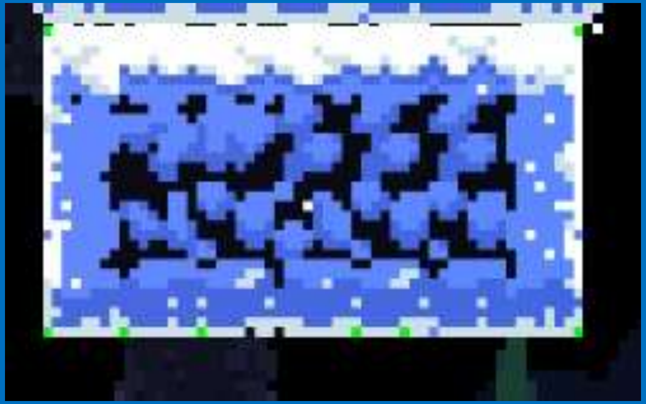
```
//// 创建物理体
    auto physicsBody = PhysicsBody::create();

    // 定义多边形顶点并创建多边形形状
    Vec2 polygonPoints1[] = {
Vec2(-622, 143),
Vec2(-480, 143),
Vec2(-480, 8),
Vec2(-622, 8)
    };

    // 创建多边形形状
    auto polygonShape1 = PhysicsShapePolygon::create(polygonPoints1, sizeof(polygonPoints1) / sizeof(polygonPoints1[0])); // 使用顶点数组创建形状
    polygonShape1->setRestitution(0.0f); // 设置反弹系数为0
    physicsBody->addShape(polygonShape1); // 将形状添加到物理体
```

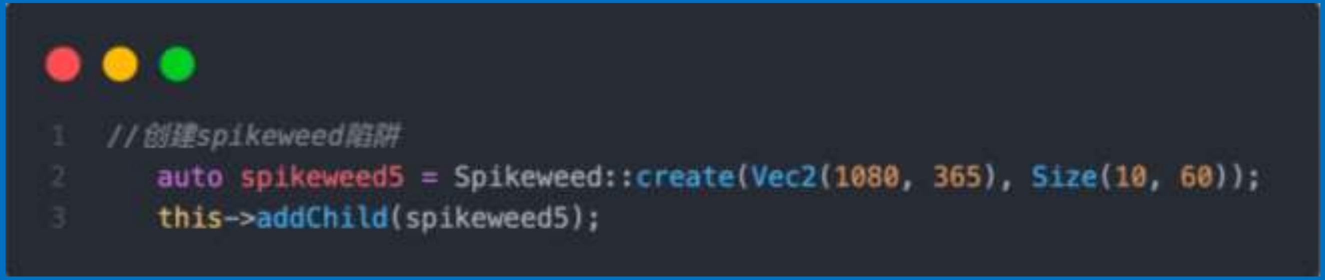


Level Class

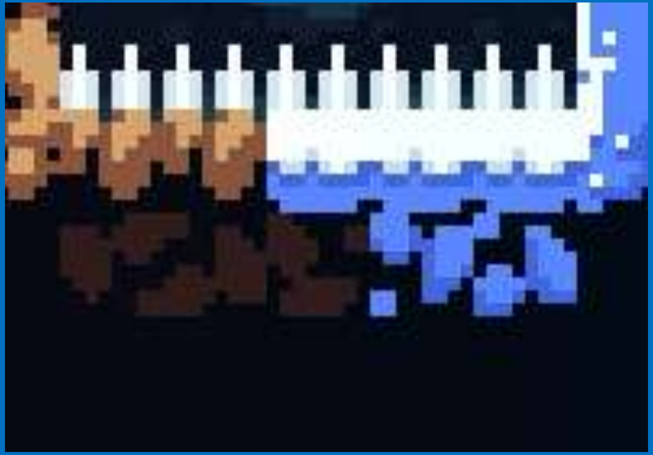


```
//创建ice
auto ice = Ice::create(Vec2(880, 498)); // 假设位置
this->addChild(ice); // 添加到场景或其他父节点中
```

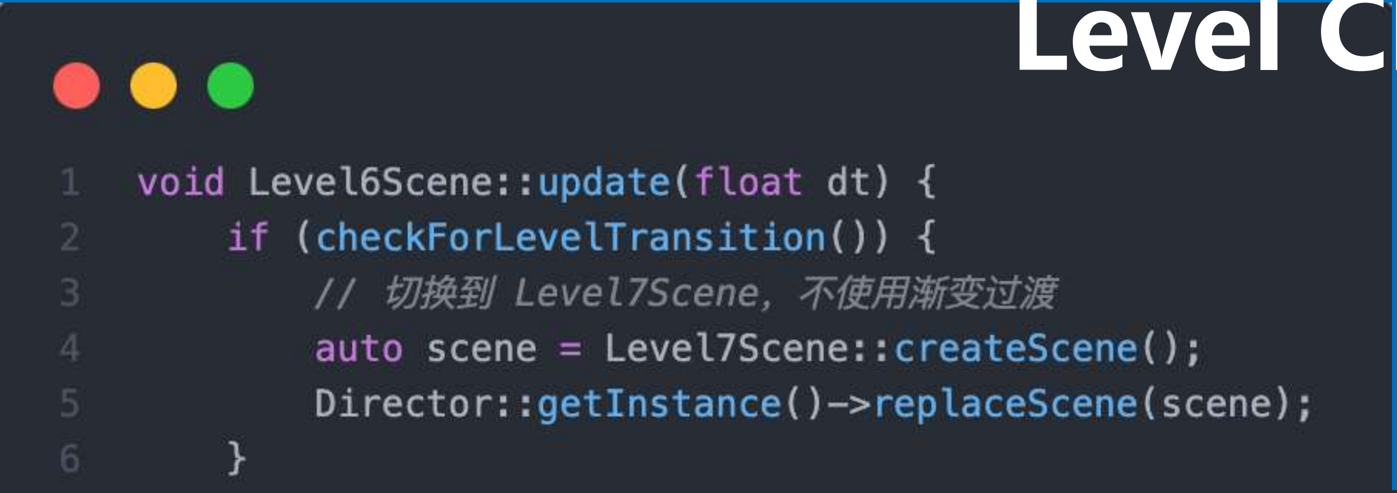




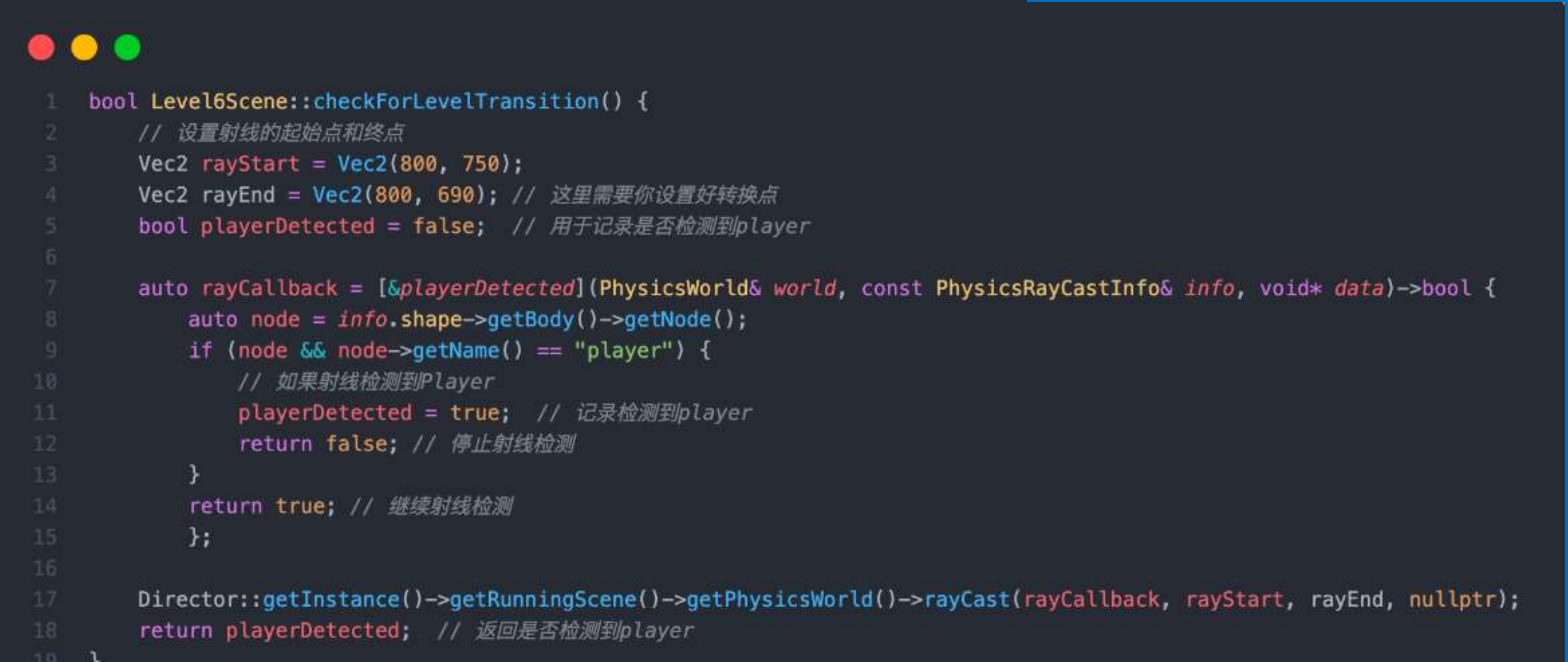
```
//创建spikeweed陷阱
    auto spikeweed5 = Spikeweed::create(Vec2(1080, 365), Size(10, 60));
    this->addChild(spikeweed5);
```



Level Class

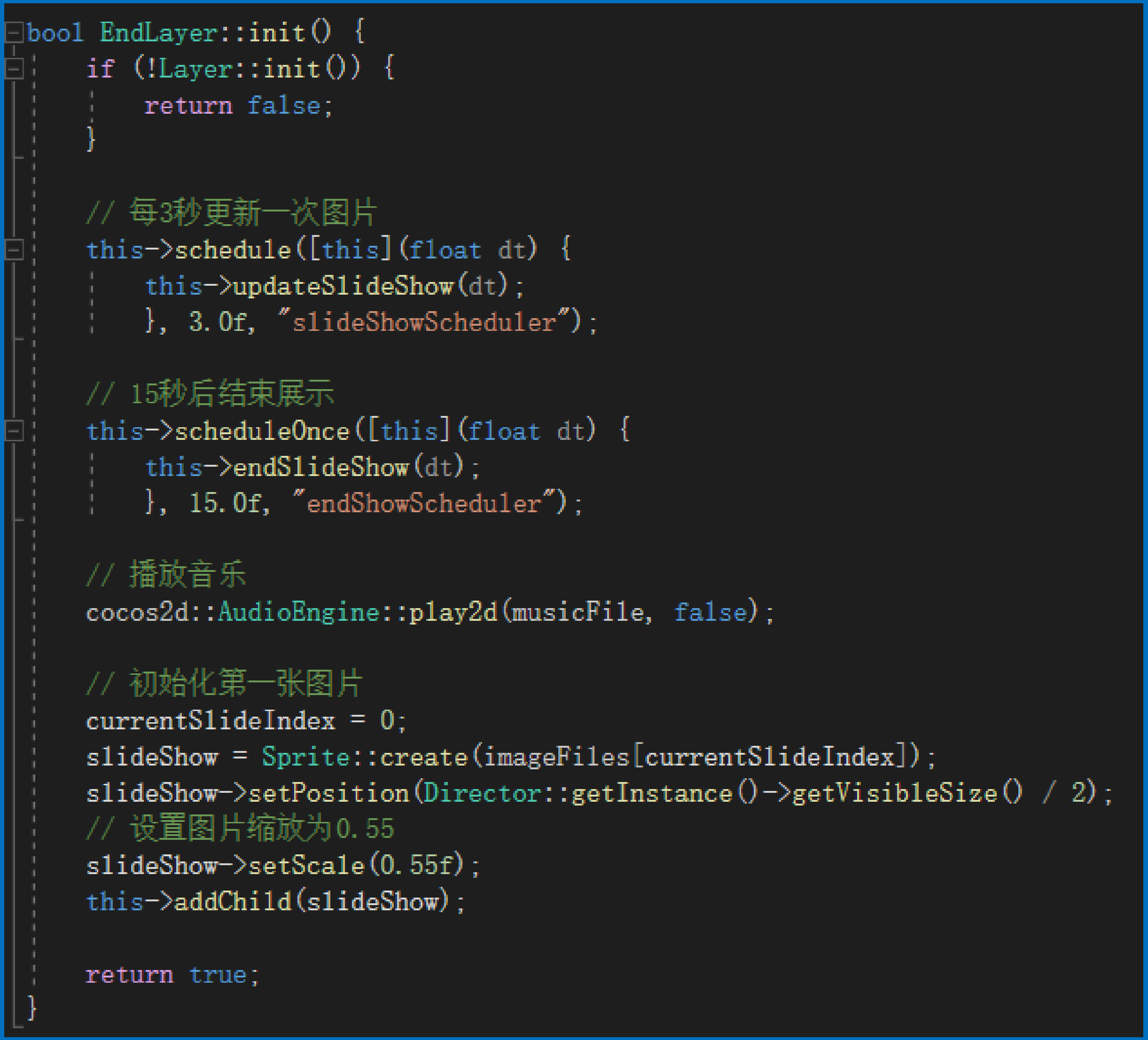


```cpp
void Level6Scene::update(float dt) {
    if (checkForLevelTransition()) {
        // 切换到 Level7Scene, 不使用渐变过渡
        auto scene = Level7Scene::createScene();
        Director::getInstance()->replaceScene(scene);
    }
```



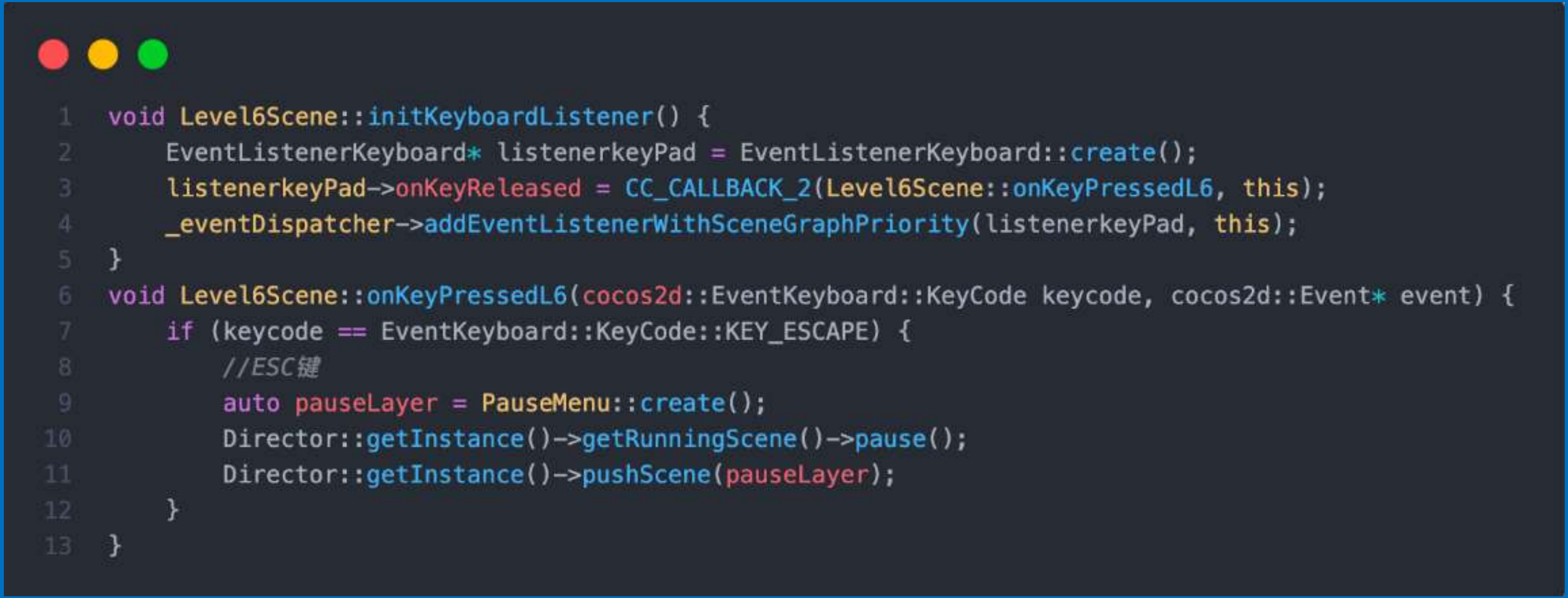
```cpp
bool Level6Scene::checkForLevelTransition() {
    // 设置射线的起始点和终点
    Vec2 rayStart = Vec2(800, 750);
    Vec2 rayEnd = Vec2(800, 690); // 这里需要你设置好转换点
    bool playerDetected = false;  // 用于记录是否检测到player

    auto rayCallback = [&playerDetected](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data)->bool {
        auto node = info.shape->getBody()->getNode();
        if (node && node->getName() == "player") {
            // 如果射线检测到Player
            playerDetected = true;  // 记录检测到player
            return false; // 停止射线检测
        }
        return true; // 继续射线检测
        };

    Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, rayStart, rayEnd, nullptr);
    return playerDetected;  // 返回是否检测到player
}
```

EndLayer Class



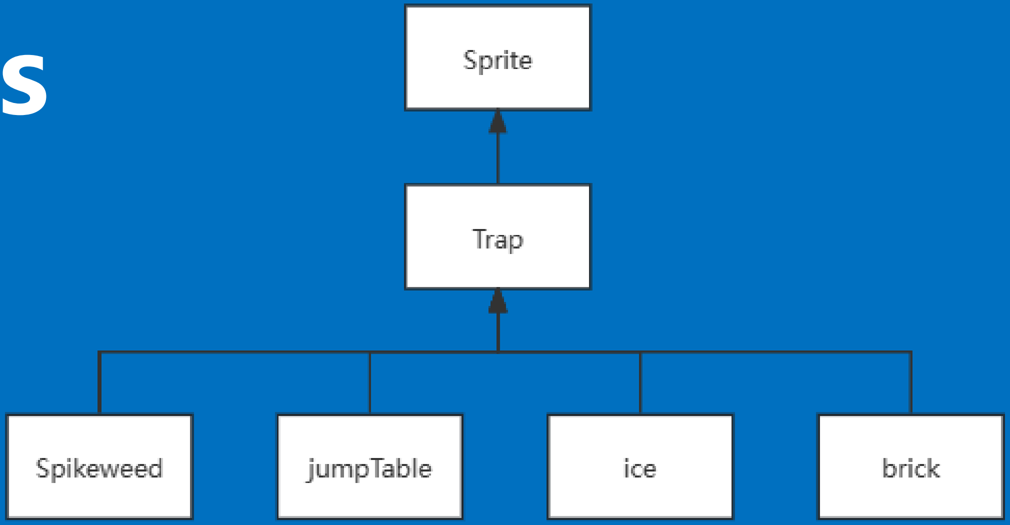
```cpp
bool EndLayer::init() {
    if (!Layer::init()) {
        return false;
    }

    // 每3秒更新一次图片
    this->schedule([this](float dt) {
        this->updateSlideShow(dt);
        }, 3.0f, "slideShowScheduler");

    // 15秒后结束展示
    this->scheduleOnce([this](float dt) {
        this->endSlideShow(dt);
        }, 15.0f, "endShowScheduler");

    // 播放音乐
    cocos2d::AudioEngine::play2d(musicFile, false);

    // 初始化第一张图片
    currentSlideIndex = 0;
    slideShow = Sprite::create(imageFiles[currentSlideIndex]);
    slideShow->setPosition(Director::getInstance()->getVisibleSize() / 2);
    // 设置图片缩放为0.55
    slideShow->setScale(0.55f);
    this->addChild(slideShow);

    return true;
}
```

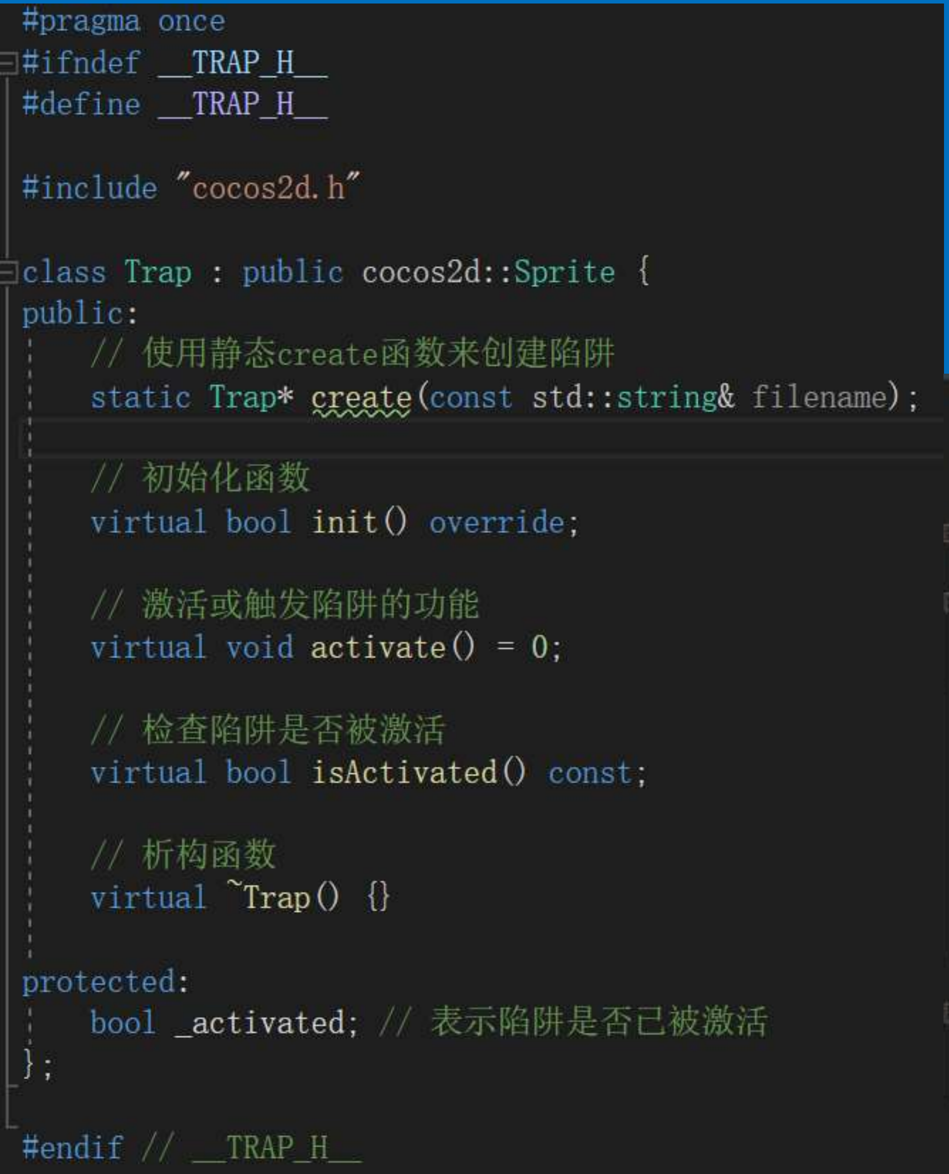
Pause Menu Scene in Level



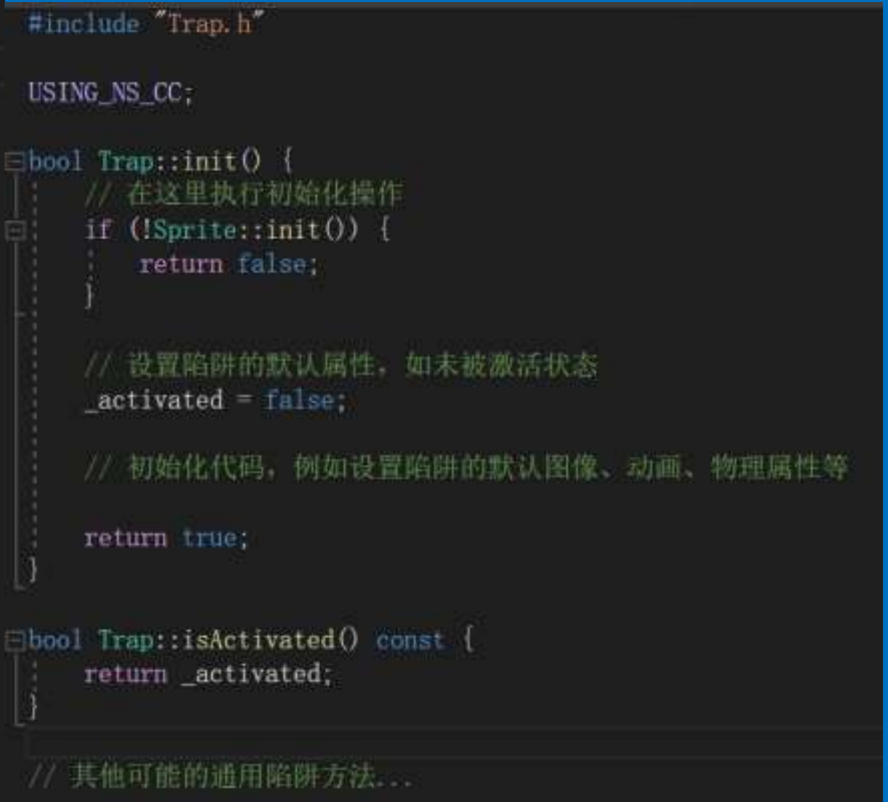
```cpp
void Level6Scene::initKeyboardListener() {
    EventListenerKeyboard* listenerkeyPad = EventListenerKeyboard::create();
    listenerkeyPad->onKeyReleased = CC_CALLBACK_2(Level6Scene::onKeyPressedL6, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(listenerkeyPad, this);
}
void Level6Scene::onKeyPressedL6(cocos2d::EventKeyboard::KeyCode keycode, cocos2d::Event* event) {
    if (keycode == EventKeyboard::KeyCode::KEY_ESCAPE) {
        //ESC鍵
        auto pauseLayer = PauseMenu::create();
        Director::getInstance()->getRunningScene()->pause();
        Director::getInstance()->pushScene(pauseLayer);
    }
}
```

Trap class



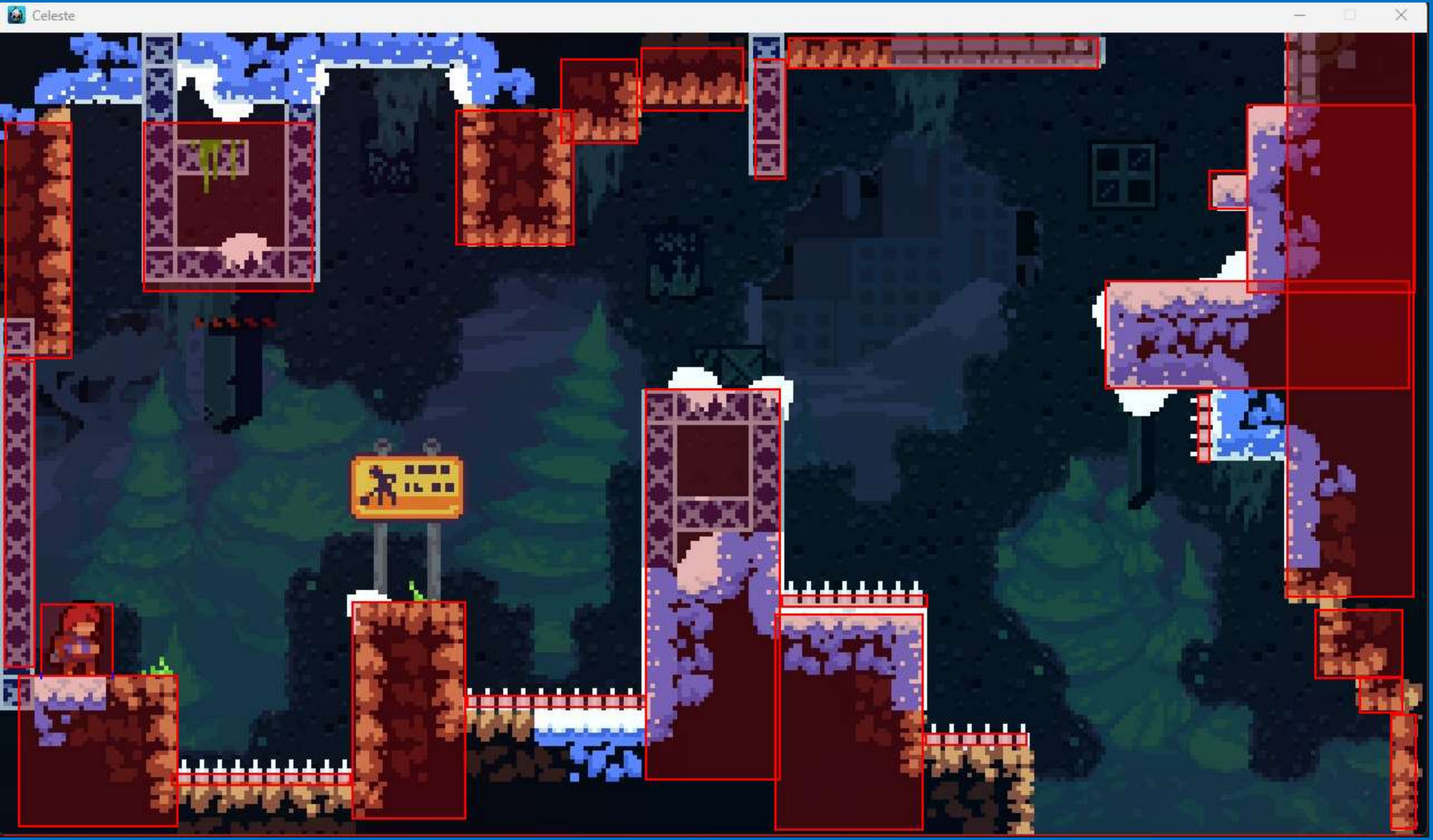


```
#pragma once
#ifndef __TRAP_H__
#define __TRAP_H__

#include "cocos2d.h"

class Trap : public cocos2d::Sprite {
public:
    // 使用静态create函数来创建陷阱
    static Trap* create(const std::string& filename);

    // 初始化函数
    virtual bool init() override;

    // 激活或触发陷阱的功能
    virtual void activate() = 0;

    // 检查陷阱是否被激活
    virtual bool isActivated() const;

    // 析构函数
    virtual ~Trap() {}

protected:
    bool _activated; // 表示陷阱是否已被激活
};

#endif // __TRAP_H__
```

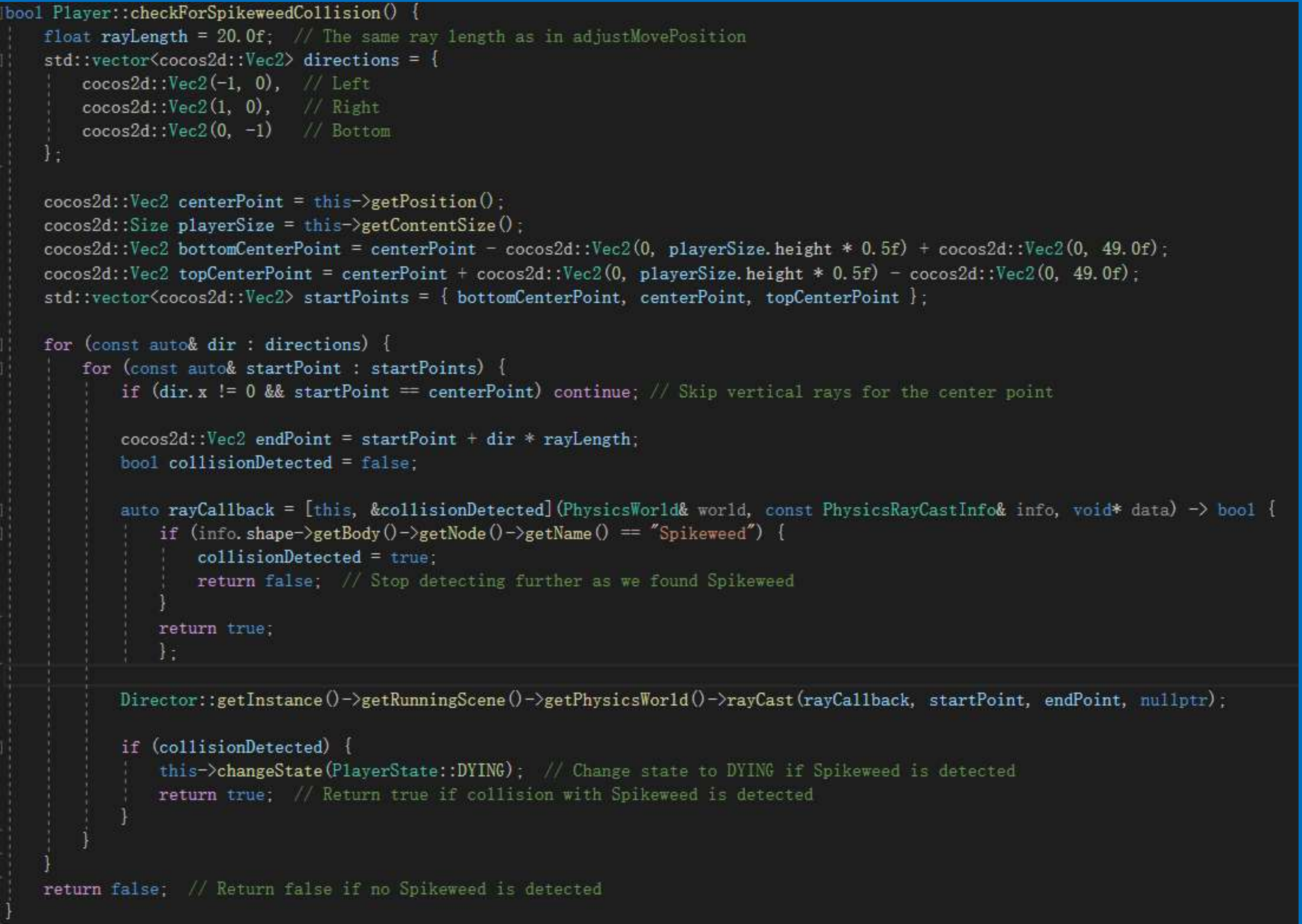


```
#include "Trap.h"

USING_NS_CC;

bool Trap::init() {
    // 在这里执行初始化操作
    if (!Sprite::init()) {
        return false;
    }

    // 设置陷阱的默认属性，如未被激活状态
    _activated = false;

    // 初始化代码，例如设置陷阱的默认图像、动画、物理属性等

    return true;
}

bool Trap::isActivated() const {
    return _activated;
}

// 其他可能的通用陷阱方法...
```

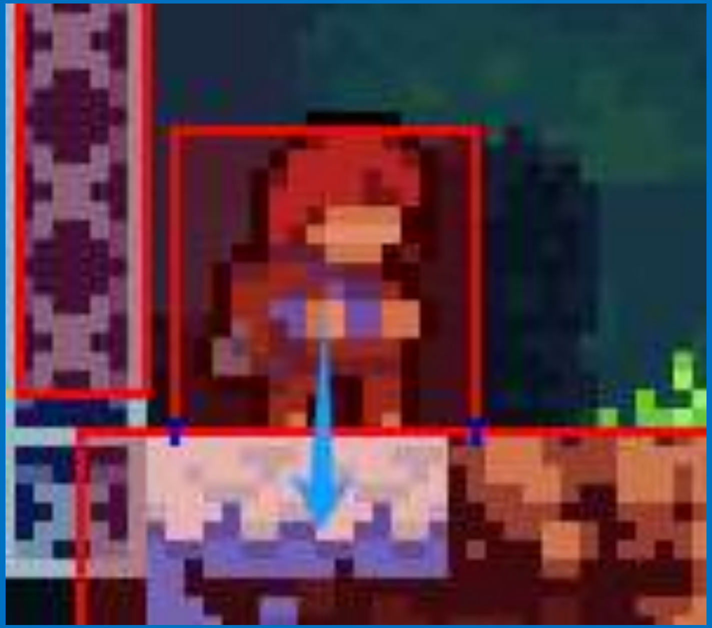
Spikeweed Class



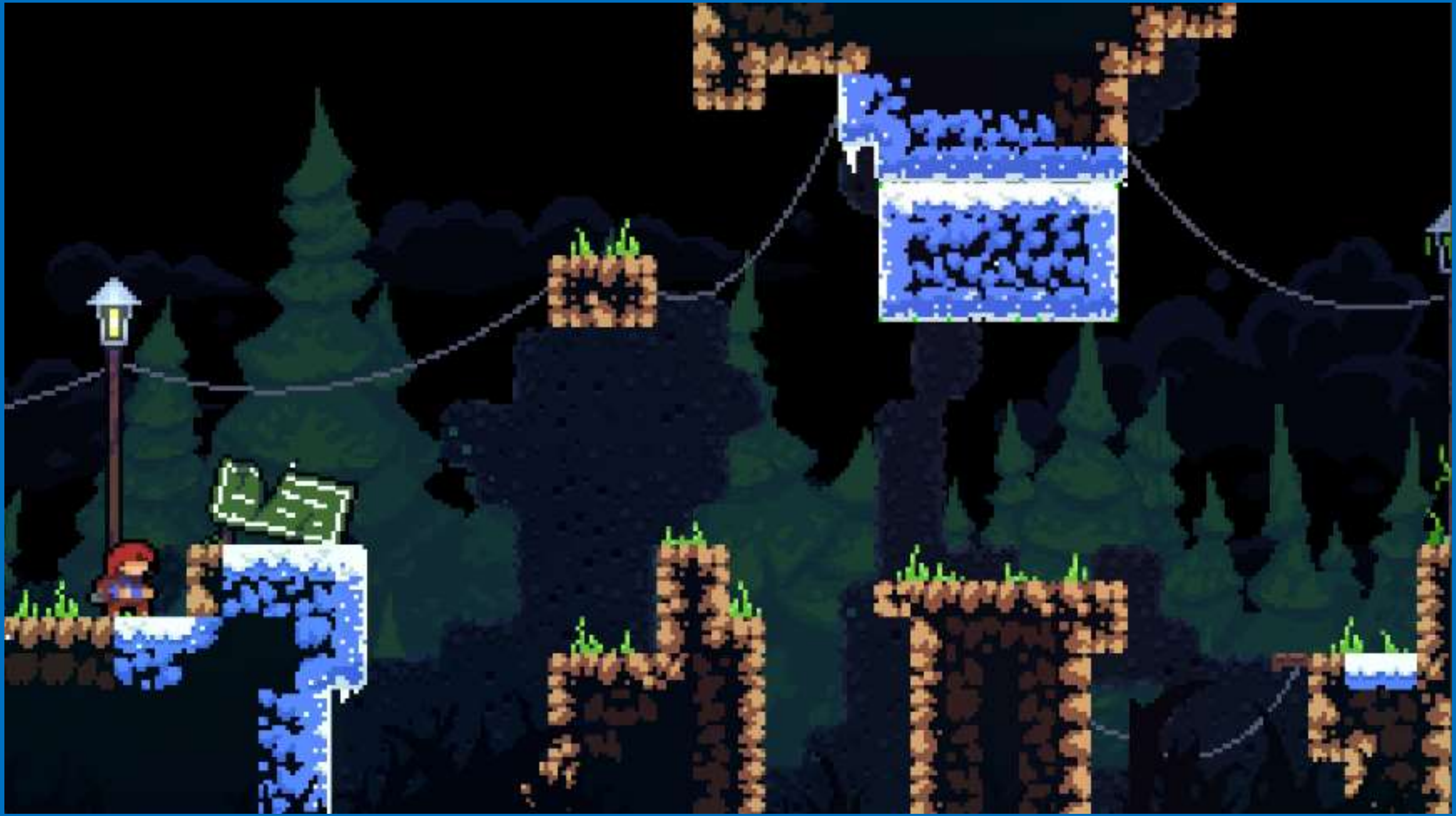
Spikeweed Class



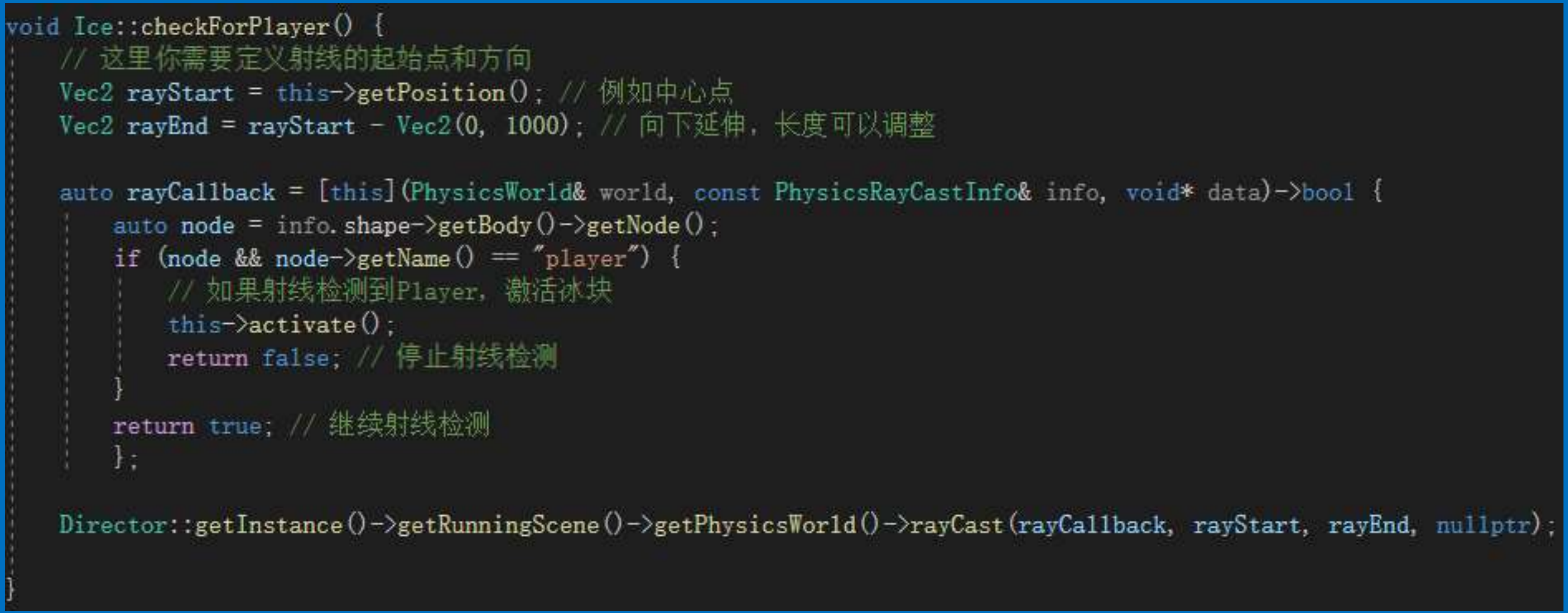
```
bool Player::checkForSpikeweedCollision() {
    float rayLength = 20.0f;  // The same ray length as in adjustMovePosition
    std::vector<cocos2d::Vec2> directions = {
        cocos2d::Vec2(-1, 0),   // Left
        cocos2d::Vec2(1, 0),    // Right
        cocos2d::Vec2(0, -1)    // Bottom
    };

    cocos2d::Vec2 centerPoint = this->getPosition();
    cocos2d::Size playerSize = this->getContentSize();
    cocos2d::Vec2 bottomCenterPoint = centerPoint - cocos2d::Vec2(0, playerSize.height * 0.5f) + cocos2d::Vec2(0, 49.0f);
    cocos2d::Vec2 topCenterPoint = centerPoint + cocos2d::Vec2(0, playerSize.height * 0.5f) - cocos2d::Vec2(0, 49.0f);
    std::vector<cocos2d::Vec2> startPoints = { bottomCenterPoint, centerPoint, topCenterPoint };

    for (const auto& dir : directions) {
        for (const auto& startPoint : startPoints) {
            if (dir.x != 0 && startPoint == centerPoint) continue; // Skip vertical rays for the center point

            cocos2d::Vec2 endPoint = startPoint + dir * rayLength;
            bool collisionDetected = false;

            auto rayCallback = [this, &collisionDetected](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data) -> bool {
                if (info.shape->getBody()->getNode()->getName() == "Spikeweed") {
                    collisionDetected = true;
                    return false;  // Stop detecting further as we found Spikeweed
                }
                return true;
                };

            Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, startPoint, endPoint, nullptr);

            if (collisionDetected) {
                this->changeState(PlayerState::DYING);  // Change state to DYING if Spikeweed is detected
                return true;  // Return true if collision with Spikeweed is detected
            }
        }
    }
    return false;  // Return false if no Spikeweed is detected
}
```



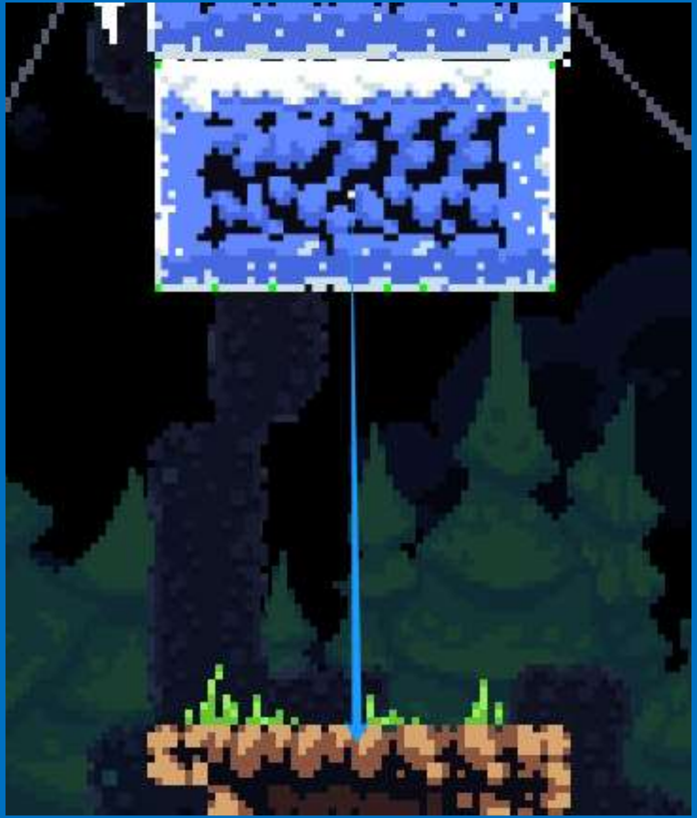
ice Class



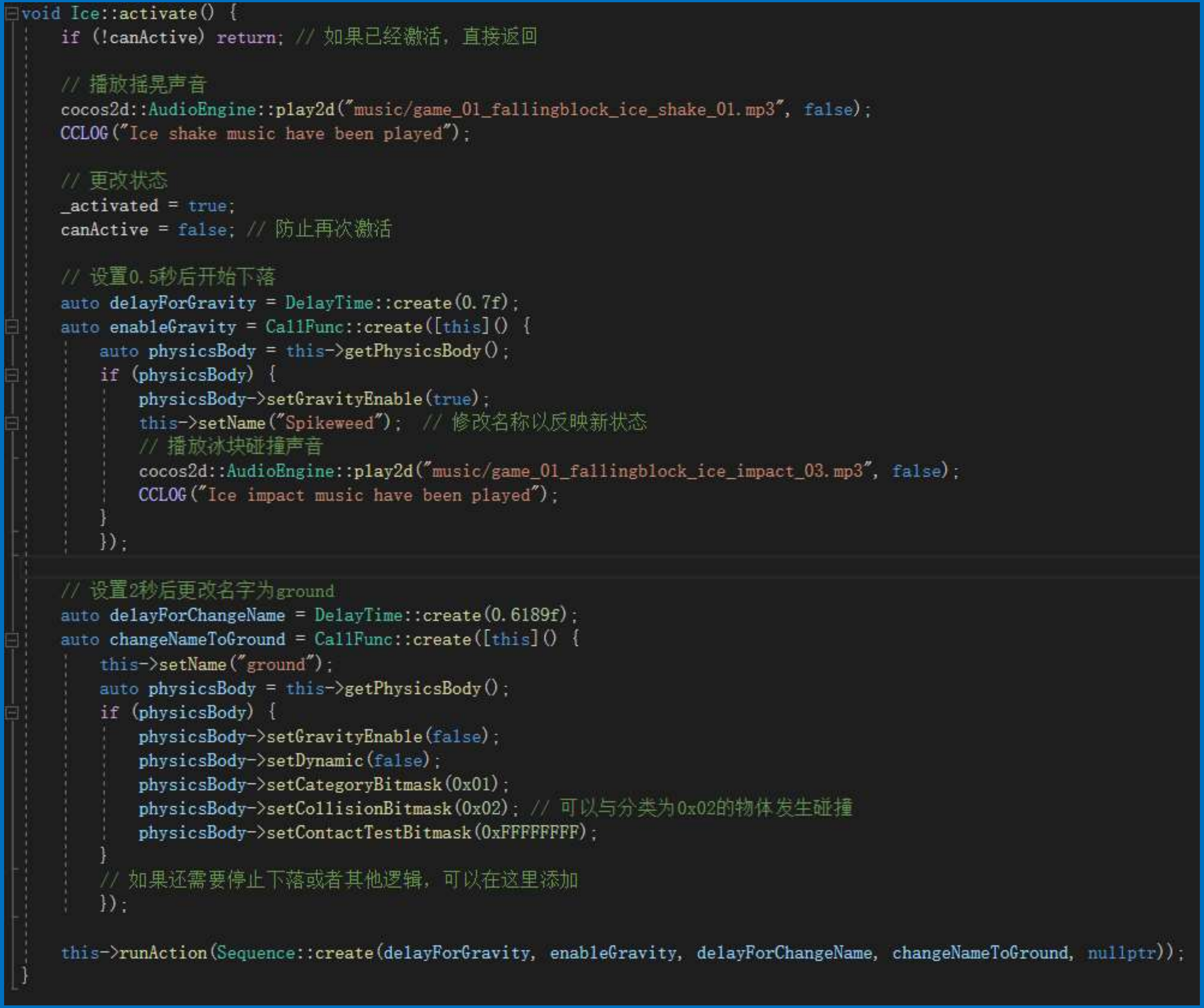
ice Class



```
void Ice::checkForPlayer() {
    // 这里你需要定义射线的起始点和方向
    Vec2 rayStart = this->getPosition(); // 例如中心点
    Vec2 rayEnd = rayStart - Vec2(0, 1000); // 向下延伸，长度可以调整

    auto rayCallback = [this](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data)->bool {
        auto node = info.shape->getBody()->getNode();
        if (node && node->getName() == "player") {
            // 如果射线检测到Player，激活冰块
            this->activate();
            return false; // 停止射线检测
        }
        return true; // 继续射线检测
    };

    Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, rayStart, rayEnd, nullptr);

}
```

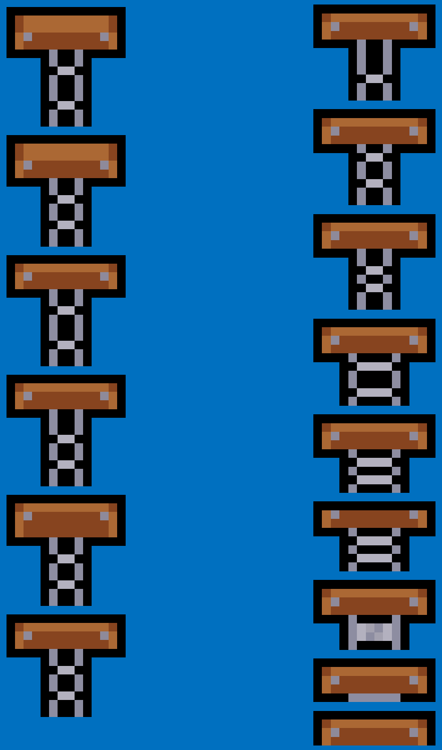


ice Class

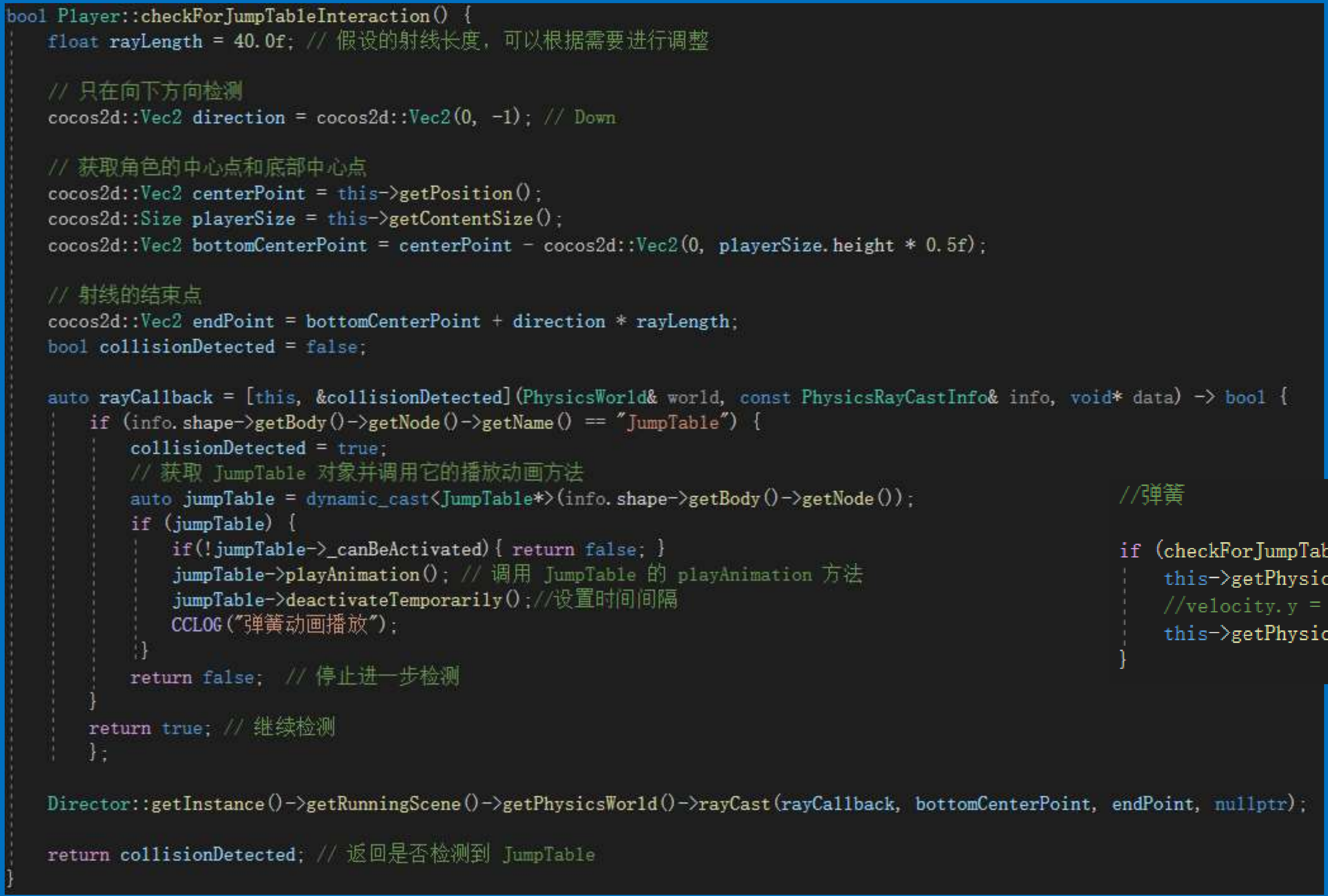


```cpp
void Ice::activate() {
    if (!canActive) return; // 如果已经激活，直接返回

    // 播放摇晃声音
    cocos2d::AudioEngine::play2d("music/game_01_fallingblock_ice_shake_01.mp3", false);
    CCLOG("Ice shake music have been played");

    // 更改状态
    _activated = true;
    canActive = false; // 防止再次激活

    // 设置0.5秒后开始下落
    auto delayForGravity = DelayTime::create(0.7f);
    auto enableGravity = CallFunc::create([this]() {
        auto physicsBody = this->getPhysicsBody();
        if (physicsBody) {
            physicsBody->setGravityEnable(true);
            this->setName("Spikeweed");  // 修改名称以反映新状态
            // 播放冰块碰撞声音
            cocos2d::AudioEngine::play2d("music/game_01_fallingblock_ice_impact_03.mp3", false);
            CCLOG("Ice impact music have been played");
        }
        });

    // 设置2秒后更改名字为ground
    auto delayForChangeName = DelayTime::create(0.6189f);
    auto changeNameToGround = CallFunc::create([this]() {
        this->setName("ground");
        auto physicsBody = this->getPhysicsBody();
        if (physicsBody) {
            physicsBody->setGravityEnable(false);
            physicsBody->setDynamic(false);
            physicsBody->setCategoryBitmask(0x01);
            physicsBody->setCollisionBitmask(0x02); // 可以与分类为0x02的物体发生碰撞
            physicsBody->setContactTestBitmask(0xFFFFFFFF);
        }
        // 如果还需要停止下落或者其他逻辑，可以在这里添加
        });

    this->runAction(Sequence::create(delayForGravity, enableGravity, delayForChangeName, changeNameToGround, nullptr));
}
```

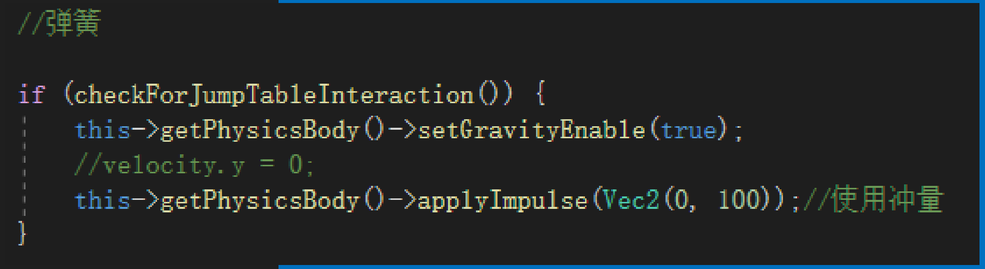
jumpTable Class



jumpTable Class

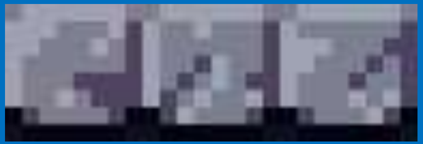


```cpp
bool Player::checkForJumpTableInteraction() {
    float rayLength = 40.0f; // 假设的射线长度，可以根据需要进行调整

    // 只在向下方向检测
    cocos2d::Vec2 direction = cocos2d::Vec2(0, -1); // Down

    // 获取角色的中心点和底部中心点
    cocos2d::Vec2 centerPoint = this->getPosition();
    cocos2d::Size playerSize = this->getContentSize();
    cocos2d::Vec2 bottomCenterPoint = centerPoint - cocos2d::Vec2(0, playerSize.height * 0.5f);

    // 射线的结束点
    cocos2d::Vec2 endPoint = bottomCenterPoint + direction * rayLength;
    bool collisionDetected = false;

    auto rayCallback = [this, &collisionDetected](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data) -> bool {
        if (info.shape->getBody()->getNode()->getName() == "JumpTable") {
            collisionDetected = true;
            // 获取 JumpTable 对象并调用它的播放动画方法
            auto jumpTable = dynamic_cast<JumpTable*>(info.shape->getBody()->getNode());
            if (jumpTable) {
                if(!jumpTable->_canBeActivated){ return false; }
                jumpTable->playAnimation(); // 调用 JumpTable 的 playAnimation 方法
                jumpTable->deactivateTemporarily();//设置时间间隔
                CCLOG("弹簧动画播放");
            }
            return false;  // 停止进一步检测
        }
        return true; // 继续检测
    };

    Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, bottomCenterPoint, endPoint, nullptr);

    return collisionDetected; // 返回是否检测到 JumpTable
}
```

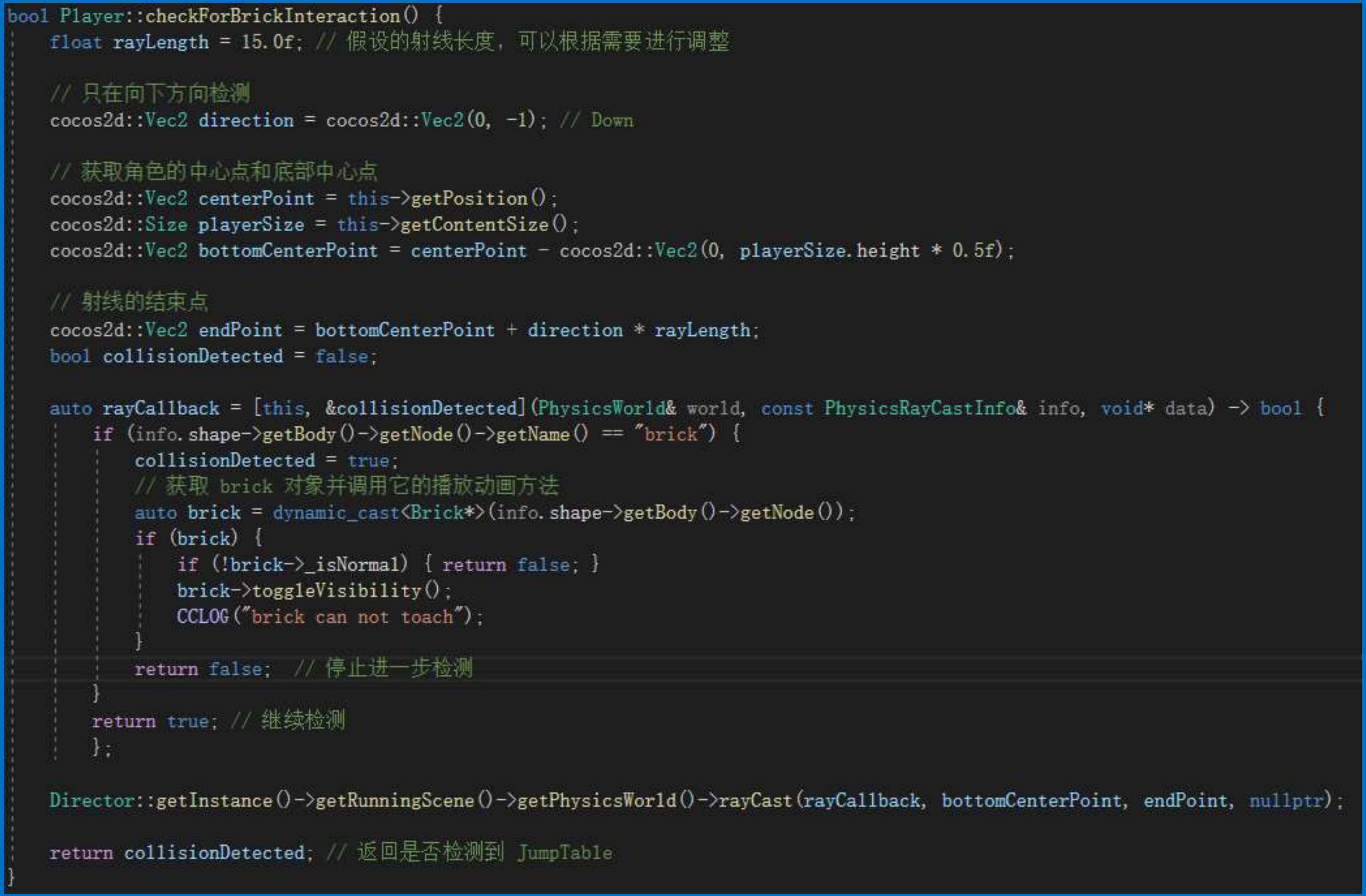


```cpp
//弹簧

if (checkForJumpTableInteraction()) {
    this->getPhysicsBody()->setGravityEnable(true);
    //velocity.y = 0;
    this->getPhysicsBody()->applyImpulse(Vec2(0, 100));//使用冲量
}
```

brick Class

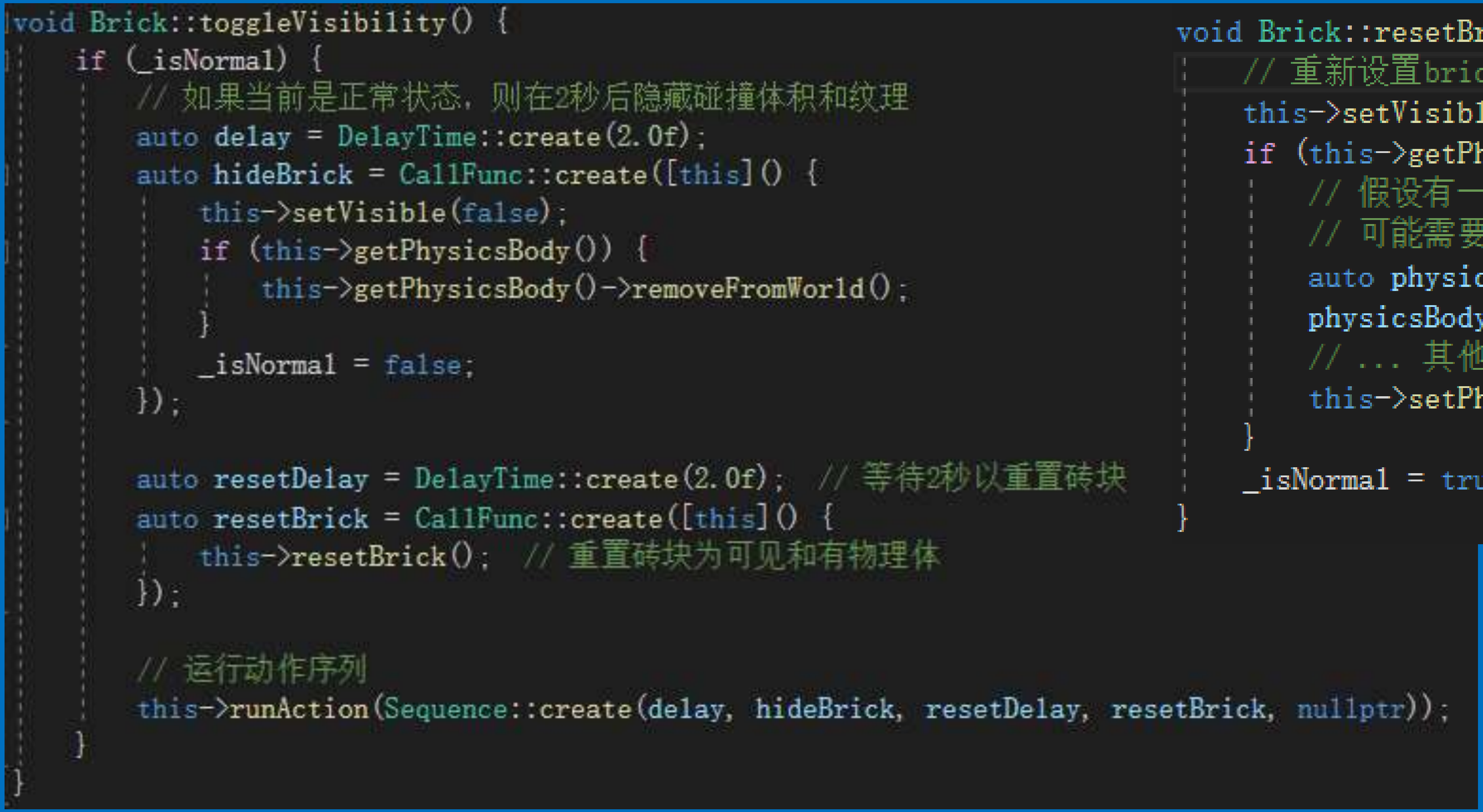


brick Class

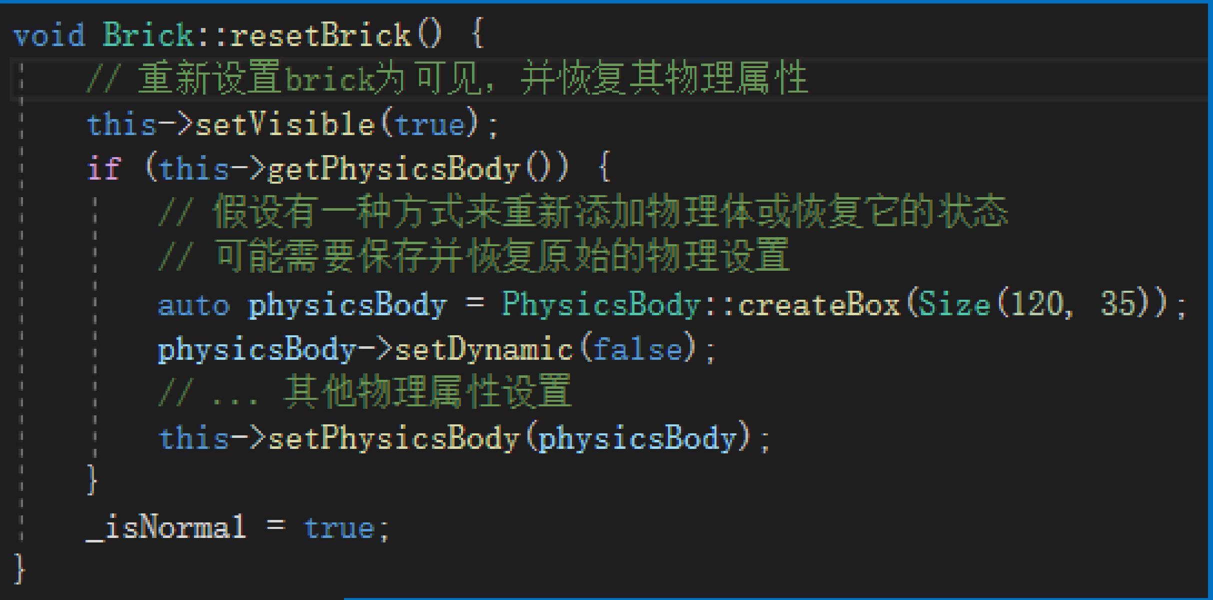


```cpp
bool Player::checkForBrickInteraction() {
    float rayLength = 15.0f; // 假设的射线长度, 可以根据需要进行调整

    // 只在向下方向检测
    cocos2d::Vec2 direction = cocos2d::Vec2(0, -1); // Down

    // 获取角色的中心点和底部中心点
    cocos2d::Vec2 centerPoint = this->getPosition();
    cocos2d::Size playerSize = this->getContentSize();
    cocos2d::Vec2 bottomCenterPoint = centerPoint - cocos2d::Vec2(0, playerSize.height * 0.5f);

    // 射线的结束点
    cocos2d::Vec2 endPoint = bottomCenterPoint + direction * rayLength;
    bool collisionDetected = false;

    auto rayCallback = [this, &collisionDetected](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data) -> bool {
        if (info.shape->getBody()->getNode()->getName() == "brick") {
            collisionDetected = true;
            // 获取 brick 对象并调用它的播放动画方法
            auto brick = dynamic_cast<Brick*>(info.shape->getBody()->getNode());
            if (brick) {
                if (!brick->_isNormal) { return false; }
                brick->toggleVisibility();
                CCLOG("brick can not toach");
            }
            return false;  // 停止进一步检测
        }
        return true; // 继续检测
        };

    Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, bottomCenterPoint, endPoint, nullptr);

    return collisionDetected; // 返回是否检测到 JumpTable
}
```

brick Class

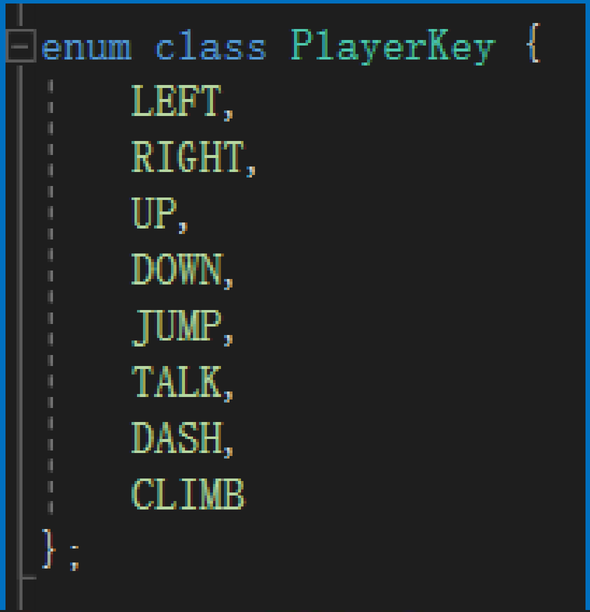


```
void Brick::toggleVisibility() {
    if (_isNormal) {
        // 如果当前是正常状态，则在2秒后隐藏碰撞体积和纹理
        auto delay = DelayTime::create(2.0f);
        auto hideBrick = CallFunc::create([this]() {
            this->setVisible(false);
            if (this->getPhysicsBody()) {
                this->getPhysicsBody()->removeFromWorld();
            }
            _isNormal = false;
        });

        auto resetDelay = DelayTime::create(2.0f);  // 等待2秒以重置砖块
        auto resetBrick = CallFunc::create([this]() {
            this->resetBrick();  // 重置砖块为可见和有物理体
        });

        // 运行动作序列
        this->runAction(Sequence::create(delay, hideBrick, resetDelay, resetBrick, nullptr));
    }
}
```

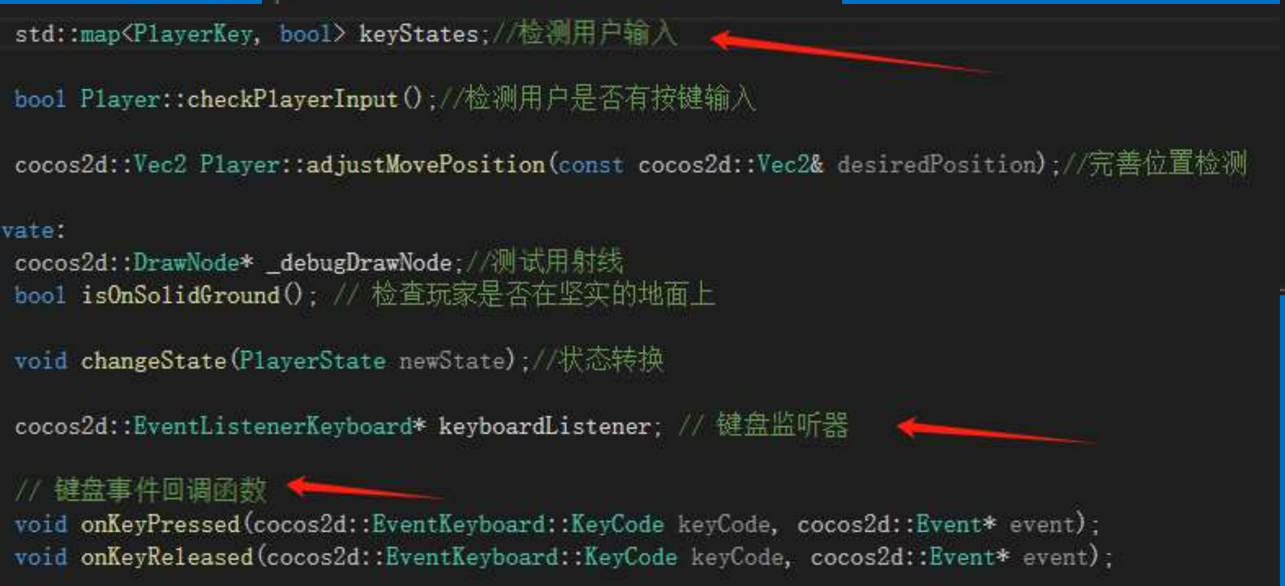


```
void Brick::resetBrick() {
    // 重新设置brick为可见，并恢复其物理属性
    this->setVisible(true);
    if (this->getPhysicsBody()) {
        // 假设有一种方式来重新添加物理体或恢复它的状态
        // 可能需要保存并恢复原始的物理设置
        auto physicsBody = PhysicsBody::createBox(Size(120, 35));
        physicsBody->setDynamic(false);
        // ... 其他物理属性设置
        this->setPhysicsBody(physicsBody);
    }
    _isNormal = true;
}
```

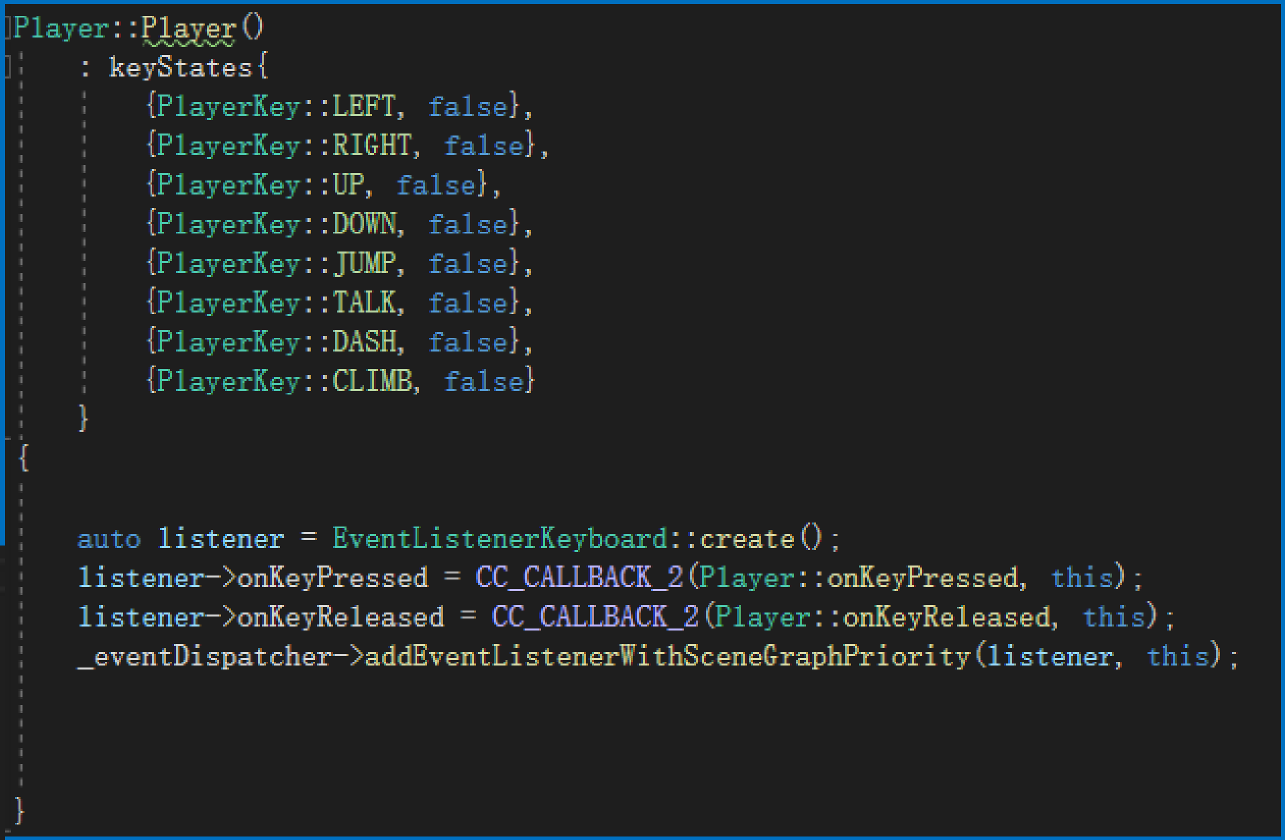
Player Class (Keyboard Monitoring)



```cpp
enum class PlayerKey {
    LEFT,
    RIGHT,
    UP,
    DOWN,
    JUMP,
    TALK,
    DASH,
    CLIMB
};
```

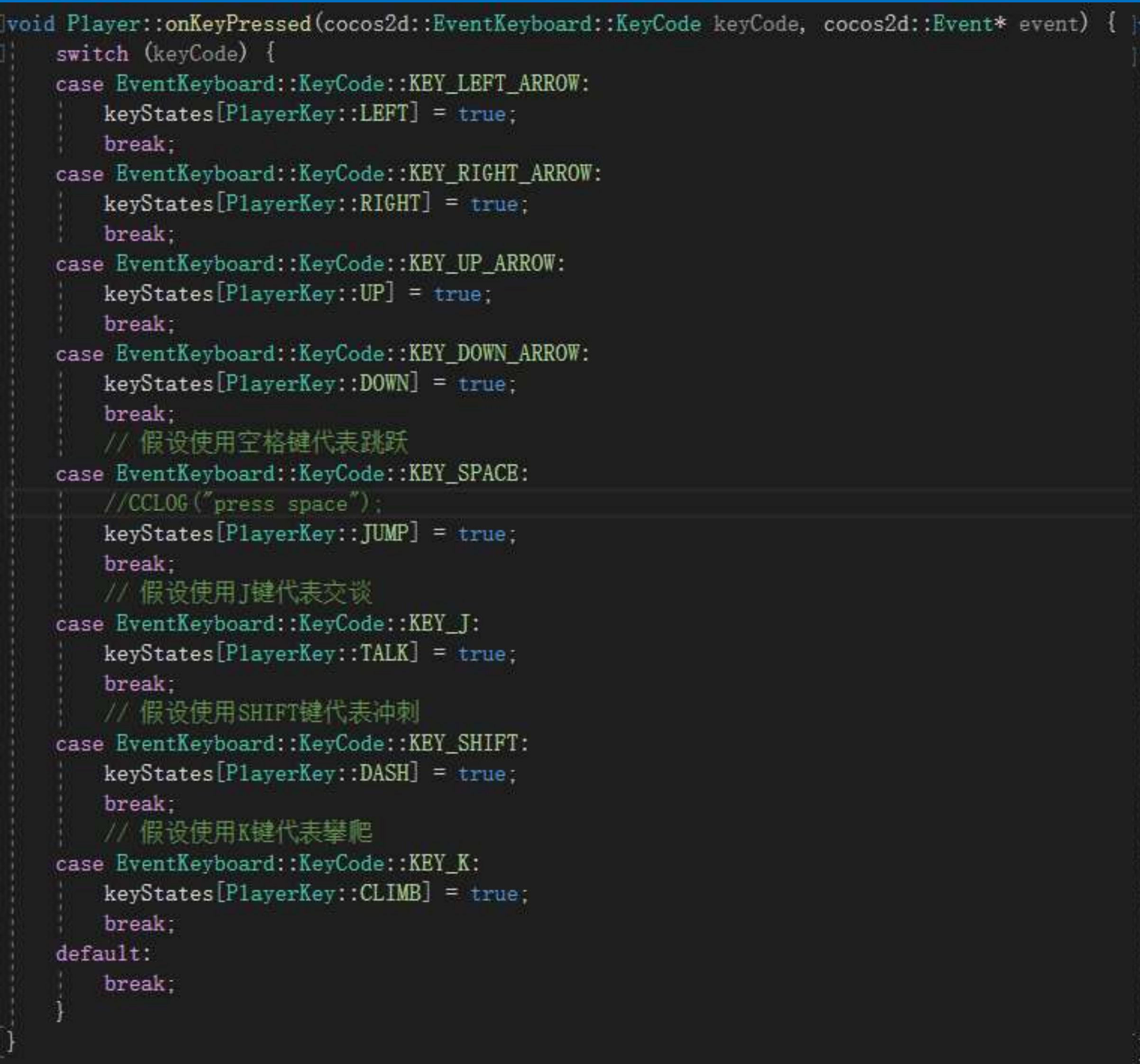


```cpp
std::map<PlayerKey, bool> keyStates;//检测用户输入

bool Player::checkPlayerInput();//检测用户是否有按键输入

cocos2d::Vec2 Player::adjustMovePosition(const cocos2d::Vec2& desiredPosition);//完善位置检测

vate:
cocos2d::DrawNode* _debugDrawNode;//测试用射线
bool isOnSolidGround(); // 检查玩家是否在坚实的地面上

void changeState(PlayerState newState);//状态转换

cocos2d::EventListenerKeyboard* keyboardListener; // 键盘监听器

// 键盘事件回调函数
void onKeyPressed(cocos2d::EventKeyboard::KeyCode keyCode, cocos2d::Event* event);
void onKeyReleased(cocos2d::EventKeyboard::KeyCode keyCode, cocos2d::Event* event);
```

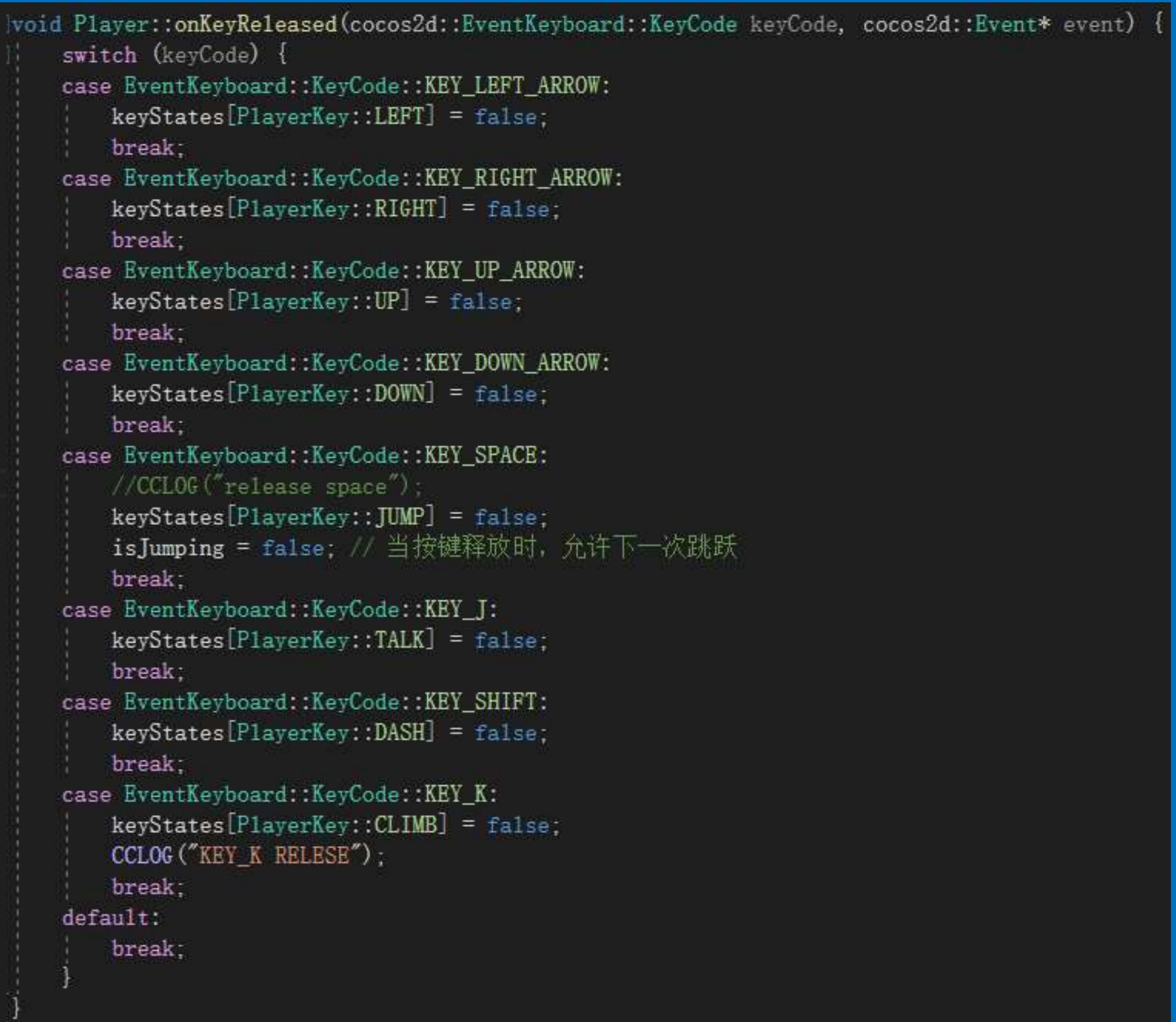


```cpp
Player::Player()
    : keyStates{
        {PlayerKey::LEFT, false},
        {PlayerKey::RIGHT, false},
        {PlayerKey::UP, false},
        {PlayerKey::DOWN, false},
        {PlayerKey::JUMP, false},
        {PlayerKey::TALK, false},
        {PlayerKey::DASH, false},
        {PlayerKey::CLIMB, false}
    }
{

    auto listener = EventListenerKeyboard::create();
    listener->onKeyPressed = CC_CALLBACK_2(Player::onKeyPressed, this);
    listener->onKeyReleased = CC_CALLBACK_2(Player::onKeyReleased, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(listener, this);


}
```

Player Class (Keyboard Monitoring)

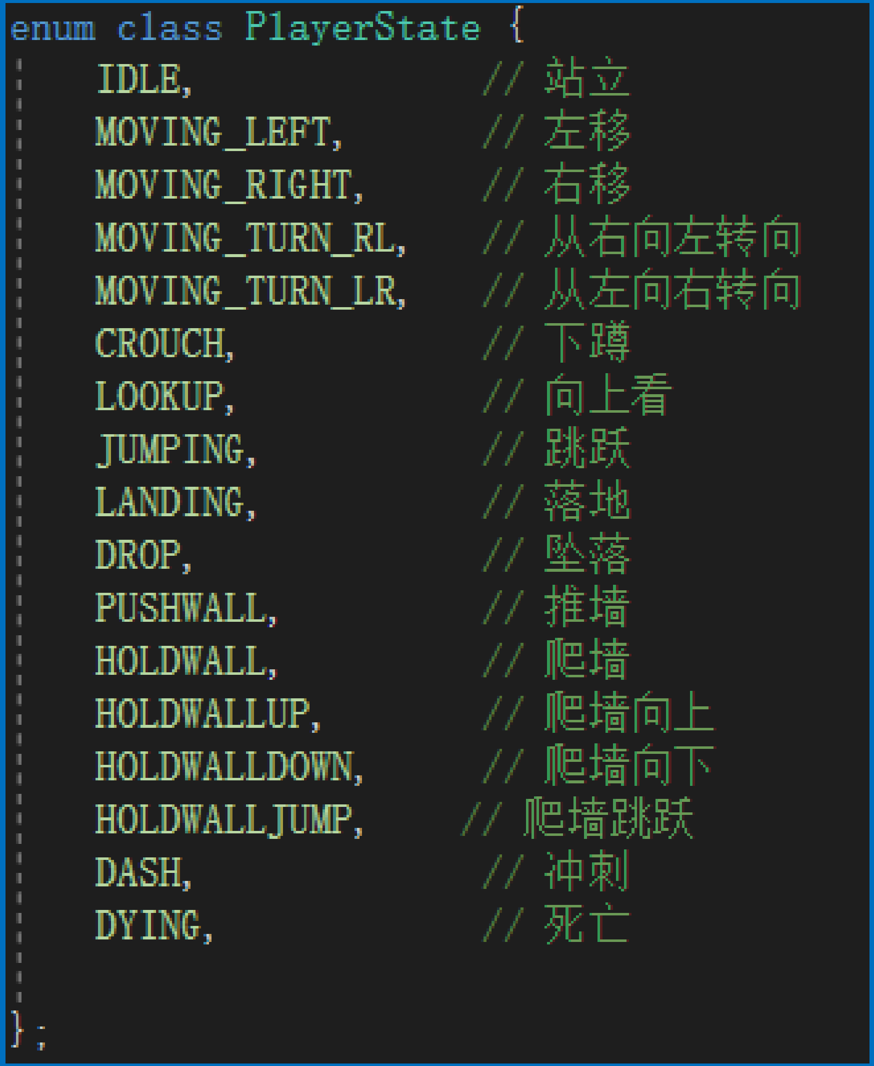


```cpp
void Player::onKeyPressed(cocos2d::EventKeyboard::KeyCode keyCode, cocos2d::Event* event) {
    switch (keyCode) {
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        keyStates[PlayerKey::LEFT] = true;
        break;
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        keyStates[PlayerKey::RIGHT] = true;
        break;
    case EventKeyboard::KeyCode::KEY_UP_ARROW:
        keyStates[PlayerKey::UP] = true;
        break;
    case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        keyStates[PlayerKey::DOWN] = true;
        break;
        // 假设使用空格键代表跳跃
    case EventKeyboard::KeyCode::KEY_SPACE:
        //CCLOG("press space");
        keyStates[PlayerKey::JUMP] = true;
        break;
        // 假设使用J键代表交谈
    case EventKeyboard::KeyCode::KEY_J:
        keyStates[PlayerKey::TALK] = true;
        break;
        // 假设使用SHIFT键代表冲刺
    case EventKeyboard::KeyCode::KEY_SHIFT:
        keyStates[PlayerKey::DASH] = true;
        break;
        // 假设使用K键代表攀爬
    case EventKeyboard::KeyCode::KEY_K:
        keyStates[PlayerKey::CLIMB] = true;
        break;
    default:
        break;
    }
}
```



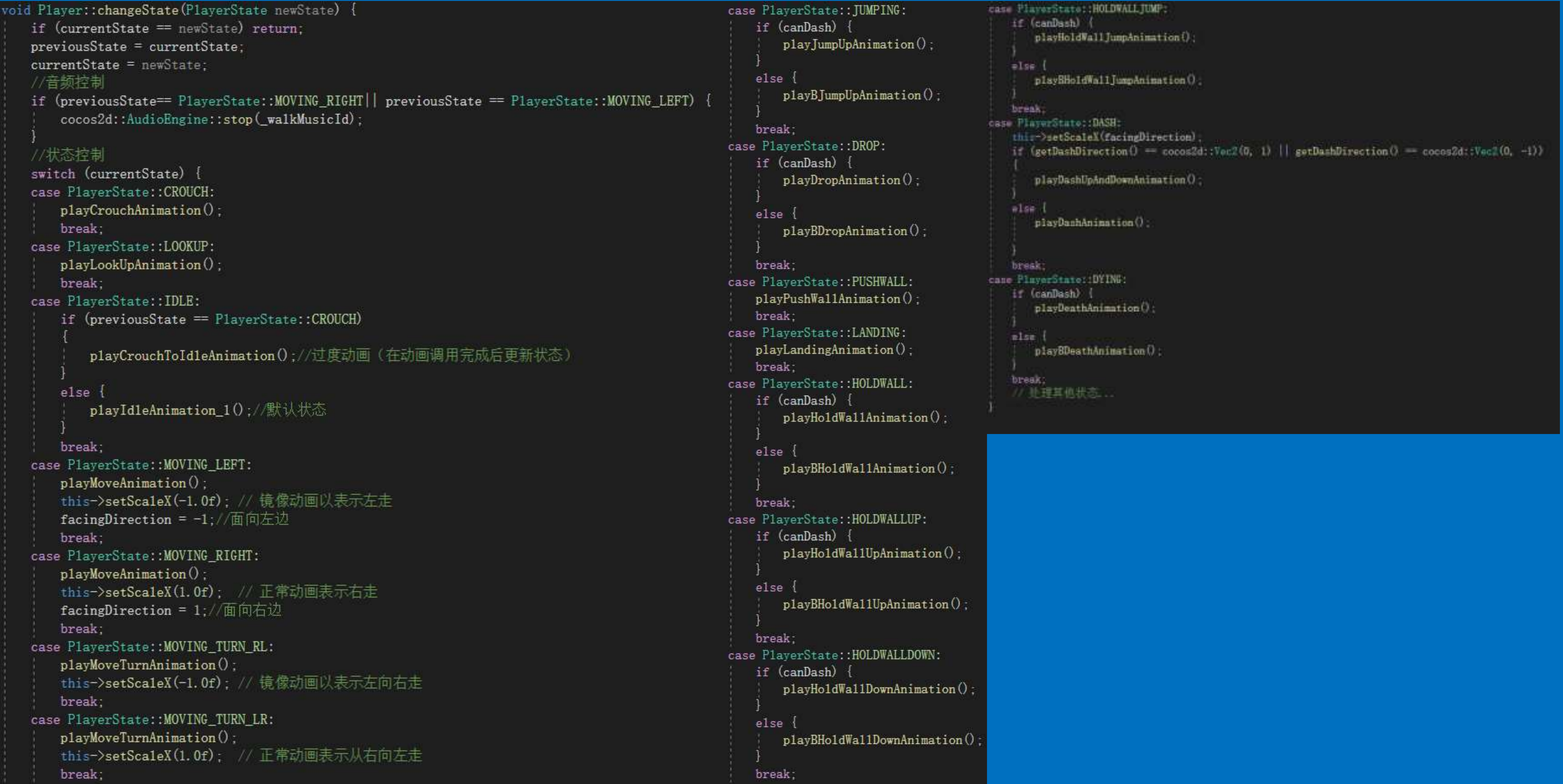
```cpp
void Player::onKeyReleased(cocos2d::EventKeyboard::KeyCode keyCode, cocos2d::Event* event) {
    switch (keyCode) {
    case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        keyStates[PlayerKey::LEFT] = false;
        break;
    case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        keyStates[PlayerKey::RIGHT] = false;
        break;
    case EventKeyboard::KeyCode::KEY_UP_ARROW:
        keyStates[PlayerKey::UP] = false;
        break;
    case EventKeyboard::KeyCode::KEY_DOWN_ARROW:
        keyStates[PlayerKey::DOWN] = false;
        break;
    case EventKeyboard::KeyCode::KEY_SPACE:
        //CCLOG("release space");
        keyStates[PlayerKey::JUMP] = false;
        isJumping = false; // 当按键释放时，允许下一次跳跃
        break;
    case EventKeyboard::KeyCode::KEY_J:
        keyStates[PlayerKey::TALK] = false;
        break;
    case EventKeyboard::KeyCode::KEY_SHIFT:
        keyStates[PlayerKey::DASH] = false;
        break;
    case EventKeyboard::KeyCode::KEY_K:
        keyStates[PlayerKey::CLIMB] = false;
        CCLOG("KEY_K RELESE");
        break;
    default:
        break;
    }
}
```

Player Class(State Machine)



```
enum class PlayerState {
    IDLE,                   // 站立
    MOVING_LEFT,            // 左移
    MOVING_RIGHT,           // 右移
    MOVING_TURN_RL,         // 从右向左转向
    MOVING_TURN_LR,         // 从左向右转向
    CROUCH,                 // 下蹲
    LOOKUP,                 // 向上看
    JUMPING,                // 跳跃
    LANDING,                // 落地
    DROP,                   // 坠落
    PUSHWALL,               // 推墙
    HOLDWALL,               // 爬墙
    HOLDWALLUP,             // 爬墙向上
    HOLDWALLDOWN,           // 爬墙向下
    HOLDWALLJUMP,          // 爬墙跳跃
    DASH,                   // 冲刺
    DYING,                  // 死亡

};
```

Player Class (State Machine)

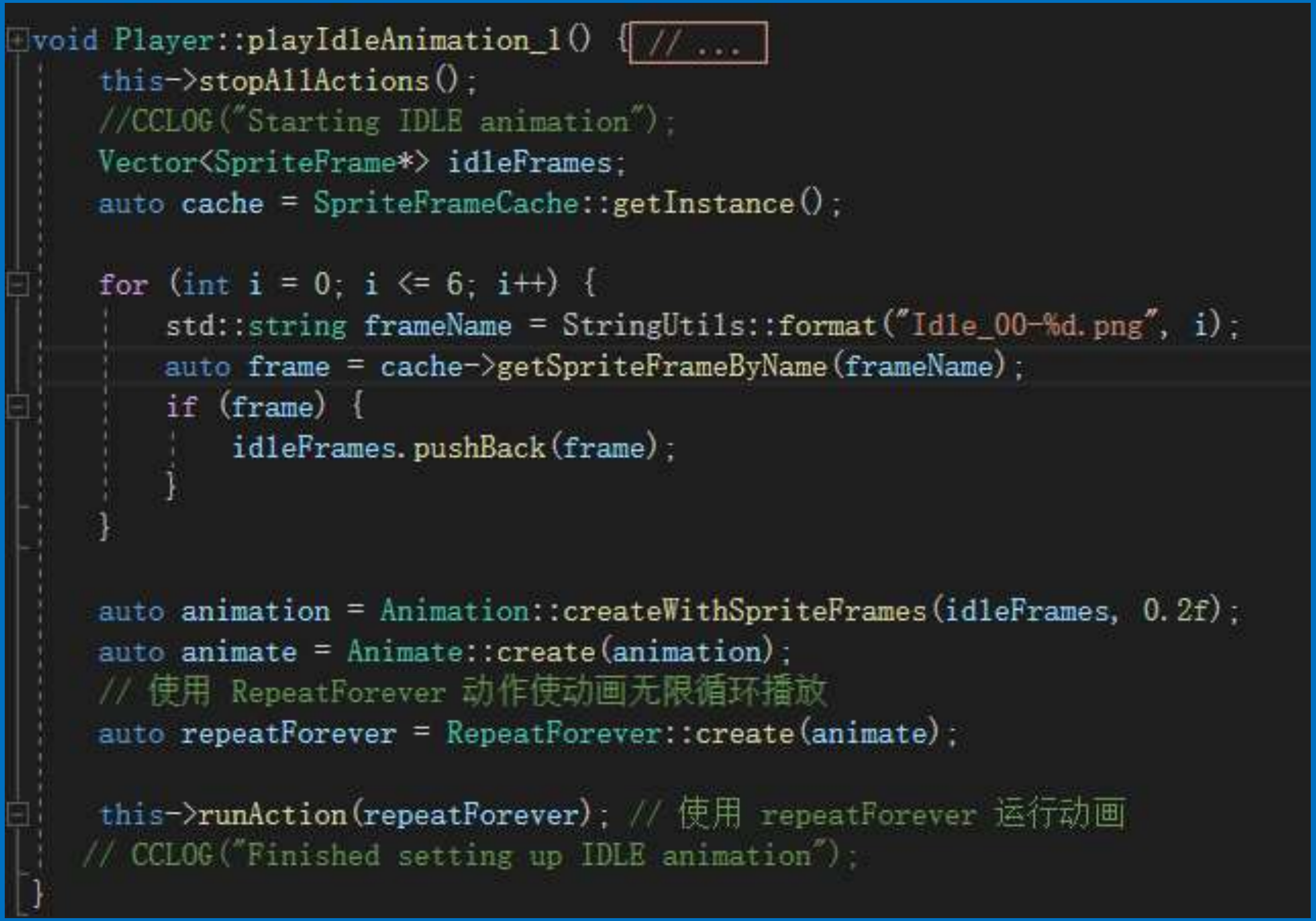


```cpp
void Player::changeState(PlayerState newState) {
    if (currentState == newState) return;
    previousState = currentState;
    currentState = newState;
    //音频控制
    if (previousState== PlayerState::MOVING_RIGHT|| previousState == PlayerState::MOVING_LEFT) {
        cocos2d::AudioEngine::stop(_walkMusicId);
    }
    //状态控制
    switch (currentState) {
    case PlayerState::CROUCH:
        playCrouchAnimation();
        break;
    case PlayerState::LOOKUP:
        playLookUpAnimation();
        break;
    case PlayerState::IDLE:
        if (previousState == PlayerState::CROUCH)
        {
            playCrouchToIdleAnimation();//过度动画（在动画调用完成后更新状态）
        }
        else {
            playIdleAnimation_1();//默认状态
        }
        break;
    case PlayerState::MOVING_LEFT:
        playMoveAnimation();
        this->setScaleX(-1.0f); // 镜像动画以表示左走
        facingDirection = -1;//面向左边
        break;
    case PlayerState::MOVING_RIGHT:
        playMoveAnimation();
        this->setScaleX(1.0f);  // 正常动画表示右走
        facingDirection = 1;//面向右边
        break;
    case PlayerState::MOVING_TURN_RL:
        playMoveTurnAnimation();
        this->setScaleX(-1.0f); // 镜像动画以表示左向右走
        break;
    case PlayerState::MOVING_TURN_LR:
        playMoveTurnAnimation();
        this->setScaleX(1.0f);  // 正常动画表示从右向左走
        break;
```

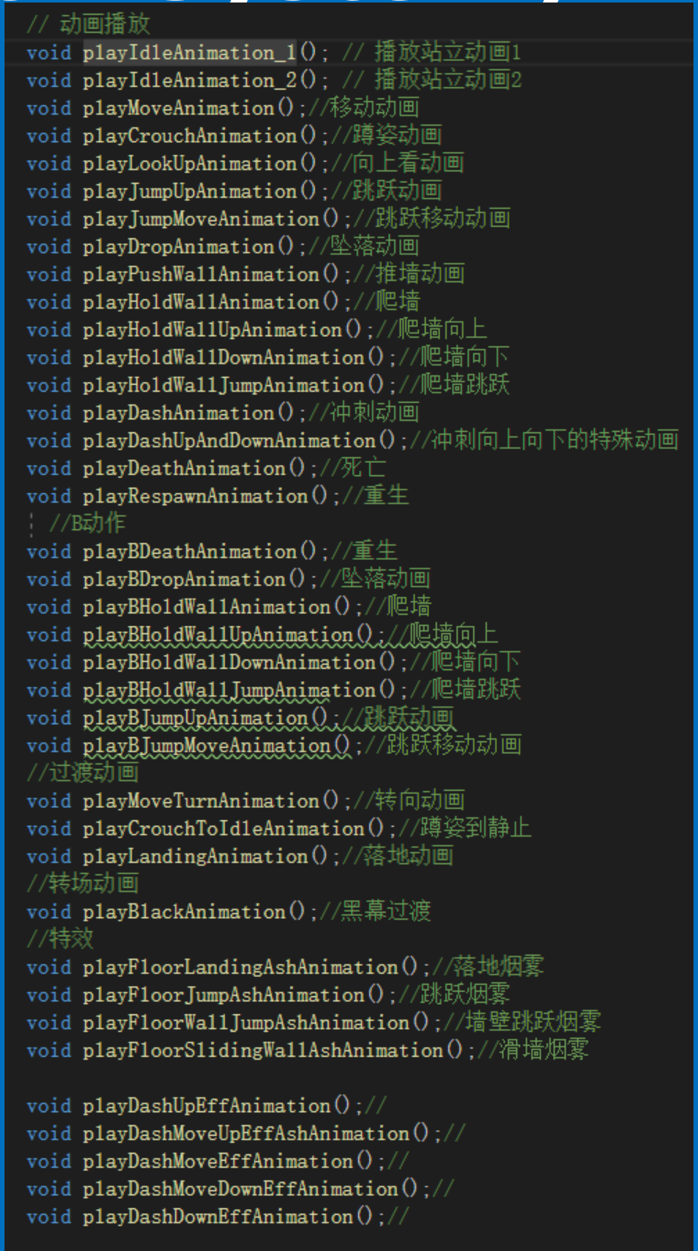
```cpp
    case PlayerState::JUMPING:
        if (canDash) {
            playJumpUpAnimation();
        }
        else {
            playBJumpUpAnimation();
        }
        break;
    case PlayerState::DROP:
        if (canDash) {
            playDropAnimation();
        }
        else {
            playBDropAnimation();
        }
        break;
    case PlayerState::PUSHWALL:
        playPushWallAnimation();
        break;
    case PlayerState::LANDING:
        playLandingAnimation();
        break;
    case PlayerState::HOLDWALL:
        if (canDash) {
            playHoldWallAnimation();
        }
        else {
            playBHoldWallAnimation();
        }
        break;
    case PlayerState::HOLDWALLUP:
        if (canDash) {
            playHoldWallUpAnimation();
        }
        else {
            playBHoldWallUpAnimation();
        }
        break;
    case PlayerState::HOLDWALLDOWN:
        if (canDash) {
            playHoldWallDownAnimation();
        }
        else {
            playBHoldWallDownAnimation();
        }
        break;
```

```cpp
    case PlayerState::HOLDWALLJUMP:
        if (canDash) {
            playHoldWallJumpAnimation();
        }
        else {
            playBHoldWallJumpAnimation();
        }
        break;
    case PlayerState::DASH:
        this->setScaleX(facingDirection);
        if (getDashDirection() == cocos2d::Vec2(0, 1) || getDashDirection() == cocos2d::Vec2(0, -1))
        {
            playDashUpAndDownAnimation();
        }
        else {
            playDashAnimation();

        }
        break;
    case PlayerState::DYING:
        if (canDash) {
            playDeathAnimation();
        }
        else {
            playBDeathAnimation();
        }
        break;
        // 处理其他状态...
    }
```

Player Class(Animation System)

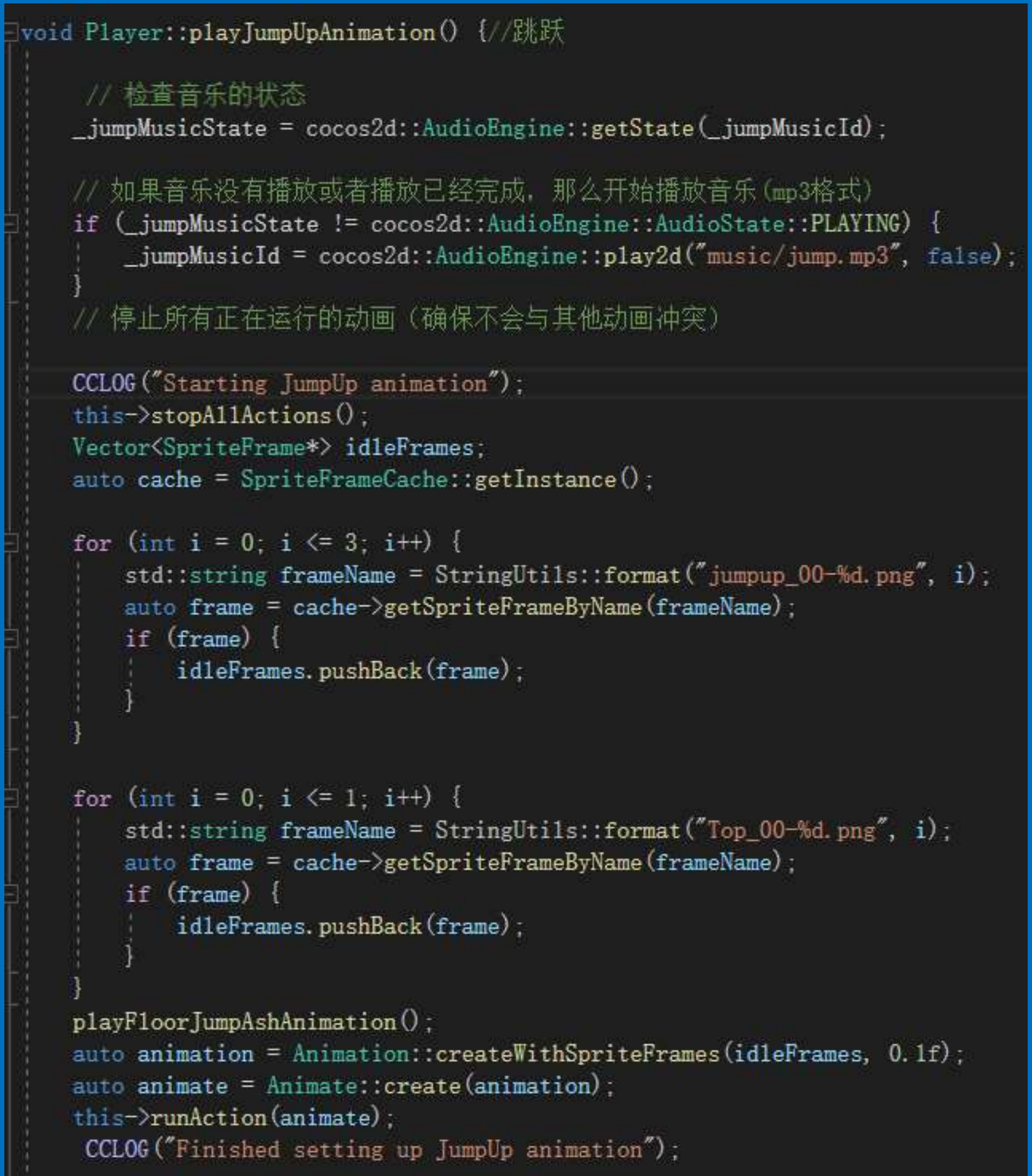


```cpp
void Player::playIdleAnimation_1() {// ...
    this->stopAllActions();
    //CCLOG("Starting IDLE animation");
    Vector<SpriteFrame*> idleFrames;
    auto cache = SpriteFrameCache::getInstance();

    for (int i = 0; i <= 6; i++) {
        std::string frameName = StringUtils::format("Idle_00-%d.png", i);
        auto frame = cache->getSpriteFrameByName(frameName);
        if (frame) {
            idleFrames.pushBack(frame);
        }
    }

    auto animation = Animation::createWithSpriteFrames(idleFrames, 0.2f);
    auto animate = Animate::create(animation);
    // 使用 RepeatForever 动作使动画无限循环播放
    auto repeatForever = RepeatForever::create(animate);

    this->runAction(repeatForever); // 使用 repeatForever 运行动画
   // CCLOG("Finished setting up IDLE animation");
}
```

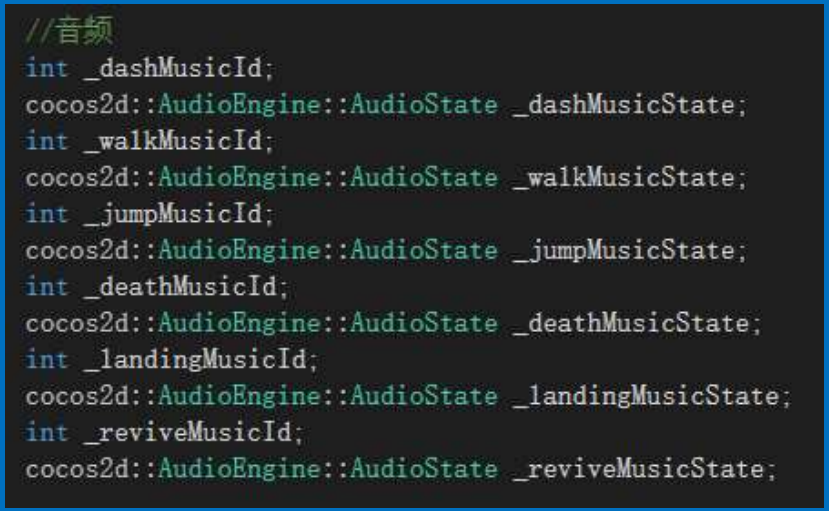


```cpp
// 动画播放
void playIdleAnimation_1(); // 播放站立动画1
void playIdleAnimation_2(); // 播放站立动画2
void playMoveAnimation();//移动动画
void playCrouchAnimation();//蹲姿动画
void playLookUpAnimation();//向上看动画
void playJumpUpAnimation();//跳跃动画
void playJumpMoveAnimation();//跳跃移动动画
void playDropAnimation();//坠落动画
void playPushWallAnimation();//推墙动画
void playHoldWallAnimation();//爬墙
void playHoldWallUpAnimation();//爬墙向上
void playHoldWallDownAnimation();//爬墙向下
void playHoldWallJumpAnimation();//爬墙跳跃
void playDashAnimation();//冲刺动画
void playDashUpAndDownAnimation();//冲刺向上向下的特殊动画
void playDeathAnimation();//死亡
void playRespawnAnimation();//重生
 //B动作
void playBDeathAnimation();//重生
void playBDropAnimation();//坠落动画
void playBHoldWallAnimation();//爬墙
void playBHoldWallUpAnimation();//爬墙向上
void playBHoldWallDownAnimation();//爬墙向下
void playBHoldWallJumpAnimation();//爬墙跳跃
void playBJumpUpAnimation();//跳跃动画
void playBJumpMoveAnimation();//跳跃移动动画
//过渡动画
void playMoveTurnAnimation();//转向动画
void playCrouchToIdleAnimation();//蹲姿到静止
void playLandingAnimation();//落地动画
//转场动画
void playBlackAnimation();//黑幕过渡
//特效
void playFloorLandingAshAnimation();//落地烟雾
void playFloorJumpAshAnimation();//跳跃烟雾
void playFloorWallJumpAshAnimation();//墙壁跳跃烟雾
void playFloorSlidingWallAshAnimation();//滑墙烟雾

void playDashUpEffAnimation();//
void playDashMoveUpEffAshAnimation();//
void playDashMoveEffAnimation();//
void playDashMoveDownEffAnimation();//
void playDashDownEffAnimation();//
```

Player Class(Audio System)

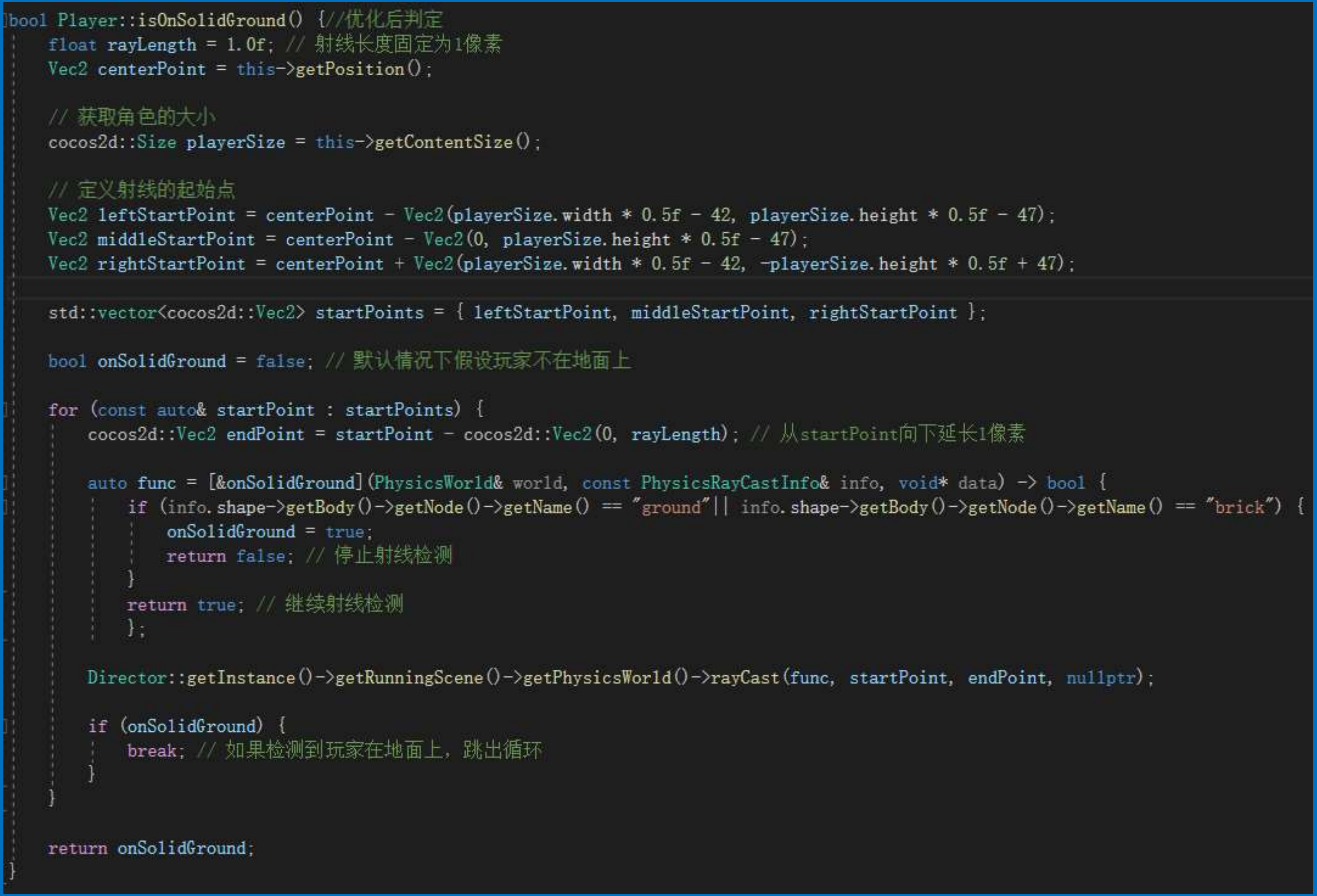


```
void Player::playJumpUpAnimation() {//跳跃

     // 检查音乐的状态
    _jumpMusicState = cocos2d::AudioEngine::getState(_jumpMusicId);

    // 如果音乐没有播放或者播放已经完成，那么开始播放音乐(mp3格式)
    if (_jumpMusicState != cocos2d::AudioEngine::AudioState::PLAYING) {
        _jumpMusicId = cocos2d::AudioEngine::play2d("music/jump.mp3", false);
    }
    // 停止所有正在运行的动画（确保不会与其他动画冲突）

    CCLOG("Starting JumpUp animation");
    this->stopAllActions();
    Vector<SpriteFrame*> idleFrames;
    auto cache = SpriteFrameCache::getInstance();

    for (int i = 0; i <= 3; i++) {
        std::string frameName = StringUtils::format("jumpup_00-%d.png", i);
        auto frame = cache->getSpriteFrameByName(frameName);
        if (frame) {
            idleFrames.pushBack(frame);
        }
    }

    for (int i = 0; i <= 1; i++) {
        std::string frameName = StringUtils::format("Top_00-%d.png", i);
        auto frame = cache->getSpriteFrameByName(frameName);
        if (frame) {
            idleFrames.pushBack(frame);
        }
    }
    playFloorJumpAshAnimation();
    auto animation = Animation::createWithSpriteFrames(idleFrames, 0.1f);
    auto animate = Animate::create(animation);
    this->runAction(animate);
     CCLOG("Finished setting up JumpUp animation");
```

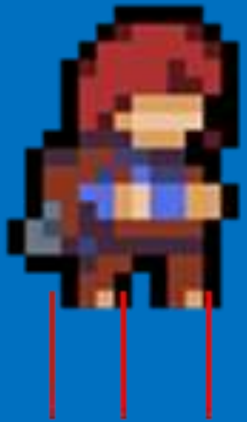


```
//音频
int _dashMusicId;
cocos2d::AudioEngine::AudioState _dashMusicState;
int _walkMusicId;
cocos2d::AudioEngine::AudioState _walkMusicState;
int _jumpMusicId;
cocos2d::AudioEngine::AudioState _jumpMusicState;
int _deathMusicId;
cocos2d::AudioEngine::AudioState _deathMusicState;
int _landingMusicId;
cocos2d::AudioEngine::AudioState _landingMusicState;
int _reviveMusicId;
cocos2d::AudioEngine::AudioState _reviveMusicState;
```

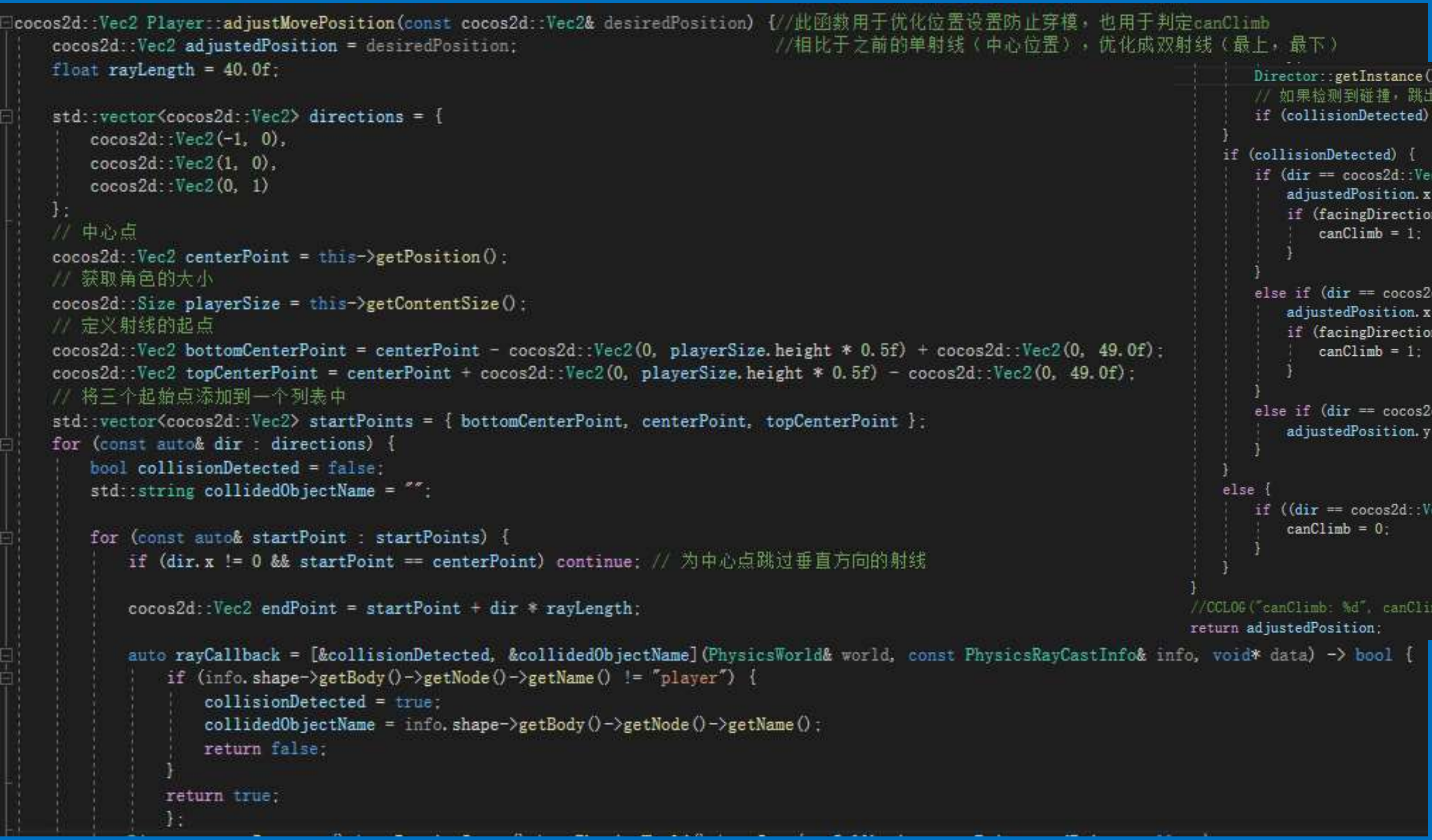
isOnSolidGround()



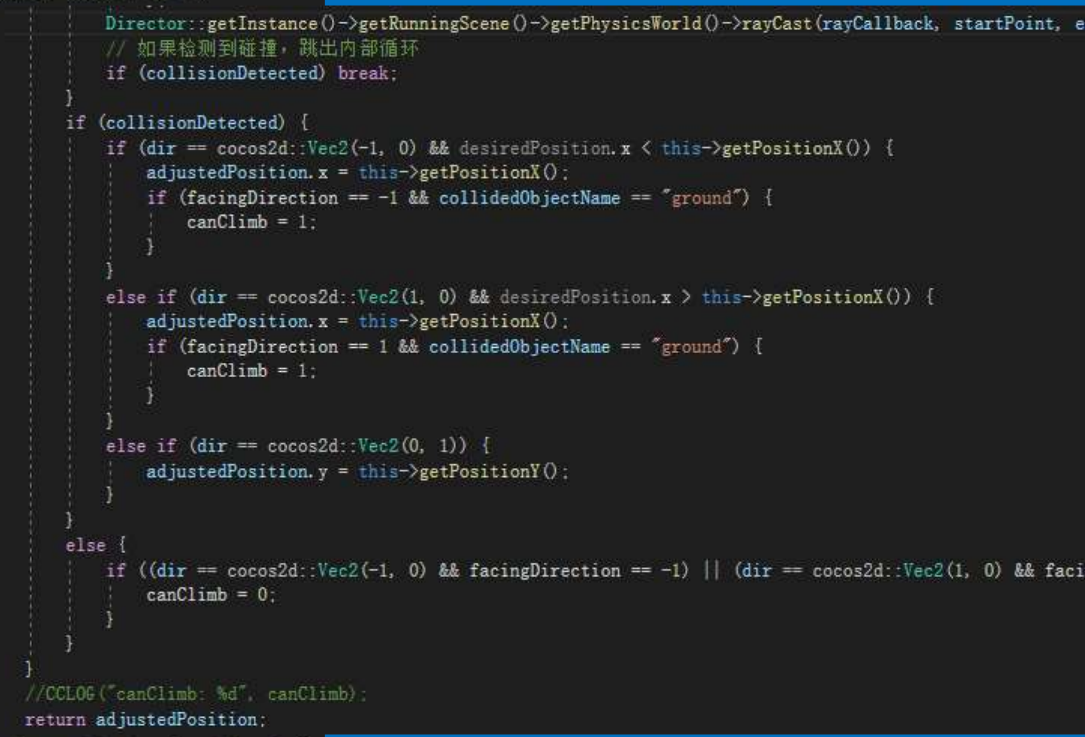
```cpp
bool Player::isOnSolidGround() {//优化后判定
    float rayLength = 1.0f; // 射线长度固定为1像素
    Vec2 centerPoint = this->getPosition();

    // 获取角色的大小
    cocos2d::Size playerSize = this->getContentSize();

    // 定义射线的起始点
    Vec2 leftStartPoint = centerPoint - Vec2(playerSize.width * 0.5f - 42, playerSize.height * 0.5f - 47);
    Vec2 middleStartPoint = centerPoint - Vec2(0, playerSize.height * 0.5f - 47);
    Vec2 rightStartPoint = centerPoint + Vec2(playerSize.width * 0.5f - 42, -playerSize.height * 0.5f + 47);

    std::vector<cocos2d::Vec2> startPoints = { leftStartPoint, middleStartPoint, rightStartPoint };

    bool onSolidGround = false; // 默认情况下假设玩家不在地面上

    for (const auto& startPoint : startPoints) {
        cocos2d::Vec2 endPoint = startPoint - cocos2d::Vec2(0, rayLength); // 从startPoint向下延长1像素

        auto func = [&onSolidGround](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data) -> bool {
            if (info.shape->getBody()->getNode()->getName() == "ground"|| info.shape->getBody()->getNode()->getName() == "brick") {
                onSolidGround = true;
                return false; // 停止射线检测
            }
            return true; // 继续射线检测
            };

        Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(func, startPoint, endPoint, nullptr);

        if (onSolidGround) {
            break; // 如果检测到玩家在地面上，跳出循环
        }
    }

    return onSolidGround;
}
```



adjustMovePosition ()

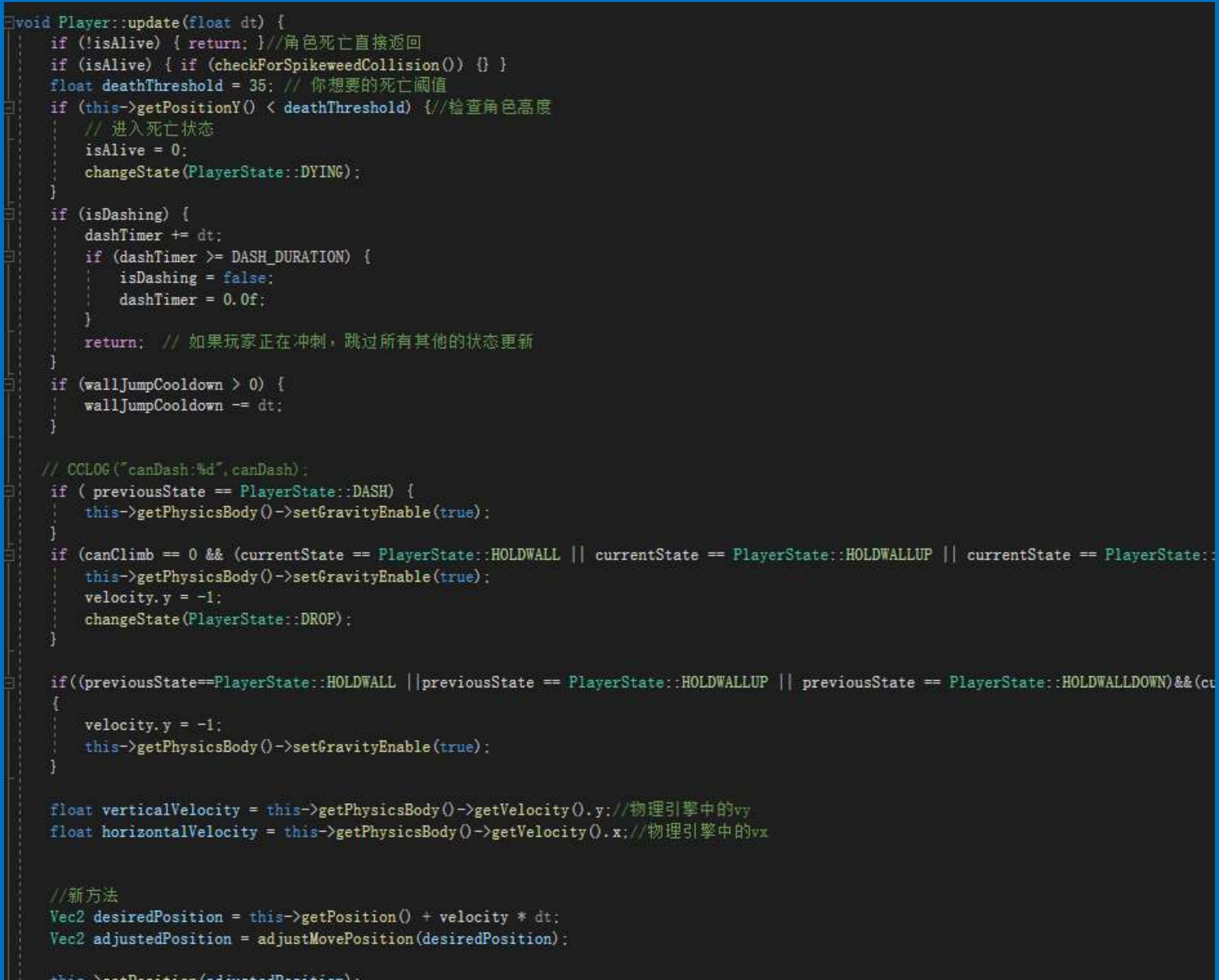


```cpp
cocos2d::Vec2 Player::adjustMovePosition(const cocos2d::Vec2& desiredPosition) {//此函数用于优化位置设置防止穿模，也用于判定canClimb
    cocos2d::Vec2 adjustedPosition = desiredPosition;                          //相比于之前的单射线（中心位置），优化成双射线（最上，最下）
    float rayLength = 40.0f;

    std::vector<cocos2d::Vec2> directions = {
        cocos2d::Vec2(-1, 0),
        cocos2d::Vec2(1, 0),
        cocos2d::Vec2(0, 1)
    };
    // 中心点
    cocos2d::Vec2 centerPoint = this->getPosition();
    // 获取角色的大小
    cocos2d::Size playerSize = this->getContentSize();
    // 定义射线的起点
    cocos2d::Vec2 bottomCenterPoint = centerPoint - cocos2d::Vec2(0, playerSize.height * 0.5f) + cocos2d::Vec2(0, 49.0f);
    cocos2d::Vec2 topCenterPoint = centerPoint + cocos2d::Vec2(0, playerSize.height * 0.5f) - cocos2d::Vec2(0, 49.0f);
    // 将三个起始点添加到一个列表中
    std::vector<cocos2d::Vec2> startPoints = { bottomCenterPoint, centerPoint, topCenterPoint };
    for (const auto& dir : directions) {
        bool collisionDetected = false;
        std::string collidedObjectName = "";

        for (const auto& startPoint : startPoints) {
            if (dir.x != 0 && startPoint == centerPoint) continue; // 为中心点跳过垂直方向的射线

            cocos2d::Vec2 endPoint = startPoint + dir * rayLength;

            auto rayCallback = [&collisionDetected, &collidedObjectName](PhysicsWorld& world, const PhysicsRayCastInfo& info, void* data) -> bool {
                if (info.shape->getBody()->getNode()->getName() != "player") {
                    collisionDetected = true;
                    collidedObjectName = info.shape->getBody()->getNode()->getName();
                    return false;
                }
                return true;
            };
```

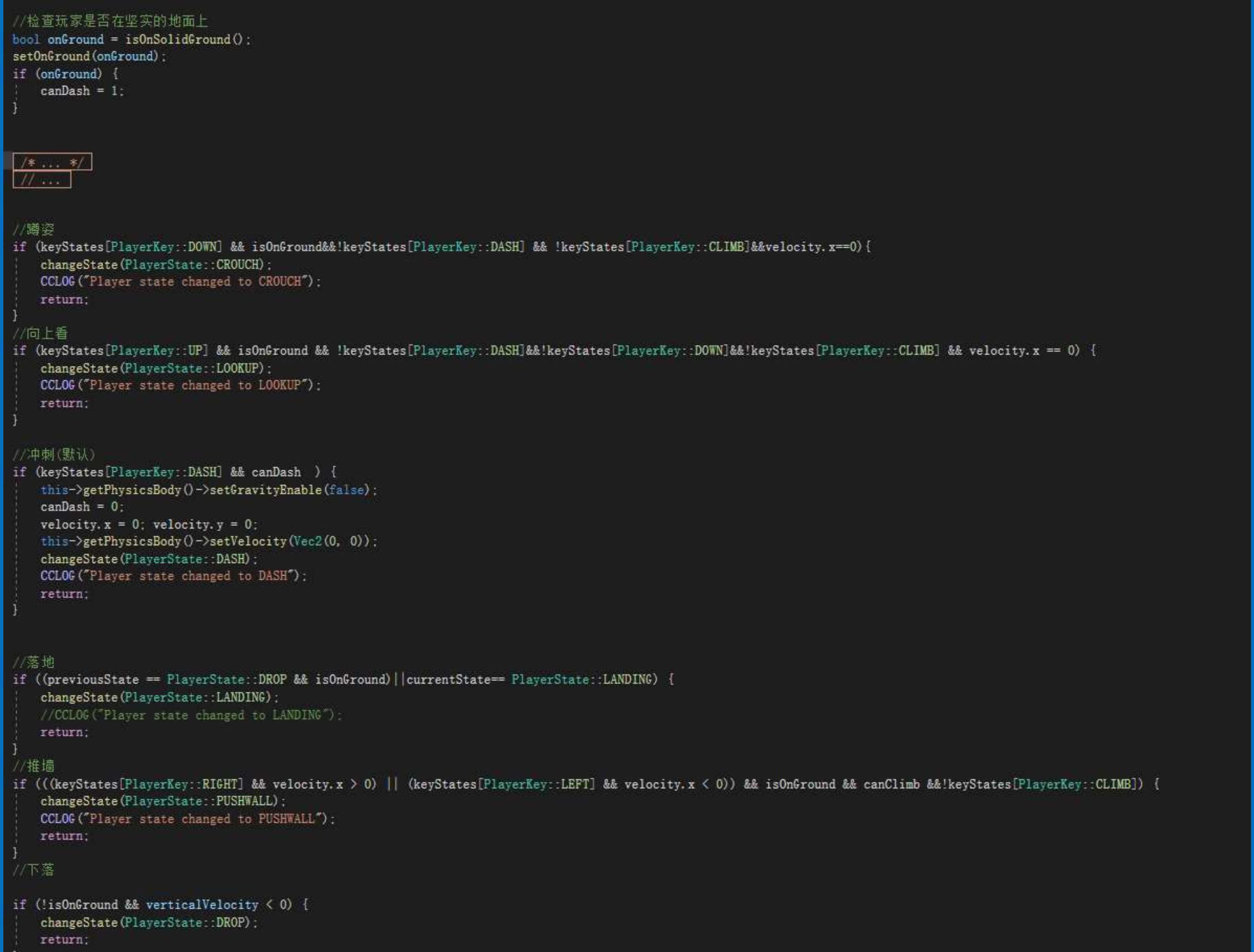


```cpp
            Director::getInstance()->getRunningScene()->getPhysicsWorld()->rayCast(rayCallback, startPoint, e
            // 如果检测到碰撞，跳出内部循环
            if (collisionDetected) break;
        }
        if (collisionDetected) {
            if (dir == cocos2d::Vec2(-1, 0) && desiredPosition.x < this->getPositionX()) {
                adjustedPosition.x = this->getPositionX();
                if (facingDirection == -1 && collidedObjectName == "ground") {
                    canClimb = 1;
                }
            }
            else if (dir == cocos2d::Vec2(1, 0) && desiredPosition.x > this->getPositionX()) {
                adjustedPosition.x = this->getPositionX();
                if (facingDirection == 1 && collidedObjectName == "ground") {
                    canClimb = 1;
                }
            }
            else if (dir == cocos2d::Vec2(0, 1)) {
                adjustedPosition.y = this->getPositionY();
            }
        }
        else {
            if ((dir == cocos2d::Vec2(-1, 0) && facingDirection == -1) || (dir == cocos2d::Vec2(1, 0) && faci
                canClimb = 0;
            }
        }
    }
    //CCLOG("canClimb: %d", canClimb);
    return adjustedPosition;
```

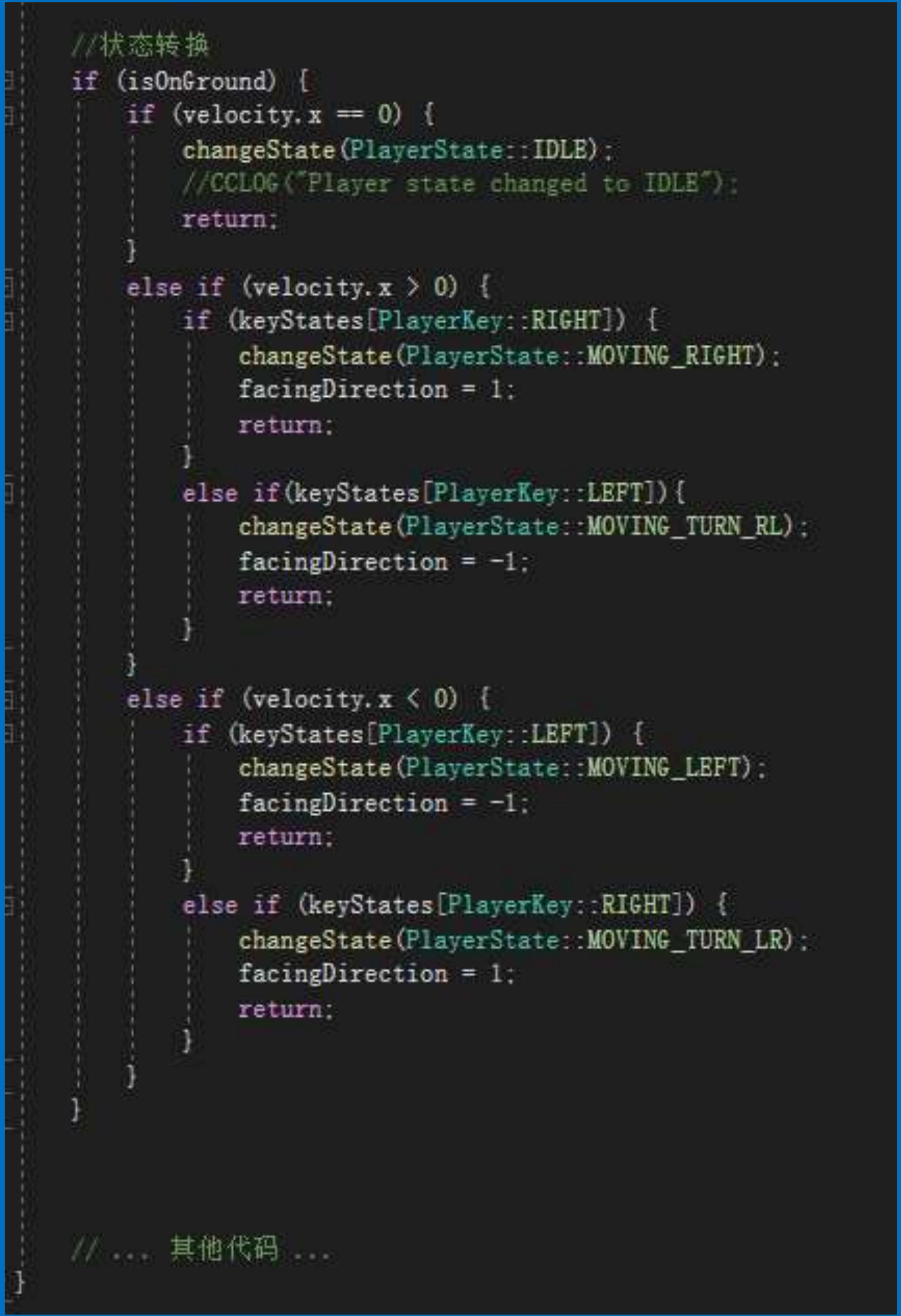
update()



```cpp
void Player::update(float dt) {
    if (!isAlive) { return; }//角色死亡直接返回
    if (isAlive) { if (checkForSpikeweedCollision()) {} }
    float deathThreshold = 35; // 你想要的死亡阈值
    if (this->getPositionY() < deathThreshold) {//检查角色高度
        // 进入死亡状态
        isAlive = 0;
        changeState(PlayerState::DYING);
    }
    if (isDashing) {
        dashTimer += dt;
        if (dashTimer >= DASH_DURATION) {
            isDashing = false;
            dashTimer = 0.0f;
        }
        return;  // 如果玩家正在冲刺，跳过所有其他的状态更新
    }
    if (wallJumpCooldown > 0) {
        wallJumpCooldown -= dt;
    }

   // CCLOG("canDash:%d",canDash);
    if ( previousState == PlayerState::DASH) {
        this->getPhysicsBody()->setGravityEnable(true);
    }
    if (canClimb == 0 && (currentState == PlayerState::HOLDWALL || currentState == PlayerState::HOLDWALLUP || currentState == PlayerState::
        this->getPhysicsBody()->setGravityEnable(true);
        velocity.y = -1;
        changeState(PlayerState::DROP);
    }

    if((previousState==PlayerState::HOLDWALL ||previousState == PlayerState::HOLDWALLUP || previousState == PlayerState::HOLDWALLDOWN)&&(cu
    {
        velocity.y = -1;
        this->getPhysicsBody()->setGravityEnable(true);
    }

    float verticalVelocity = this->getPhysicsBody()->getVelocity().y;//物理引擎中的vy
    float horizontalVelocity = this->getPhysicsBody()->getVelocity().x;//物理引擎中的vx


    //新方法
    Vec2 desiredPosition = this->getPosition() + velocity * dt;
    Vec2 adjustedPosition = adjustMovePosition(desiredPosition);

    this->setPosition(adjustedPosition);
```



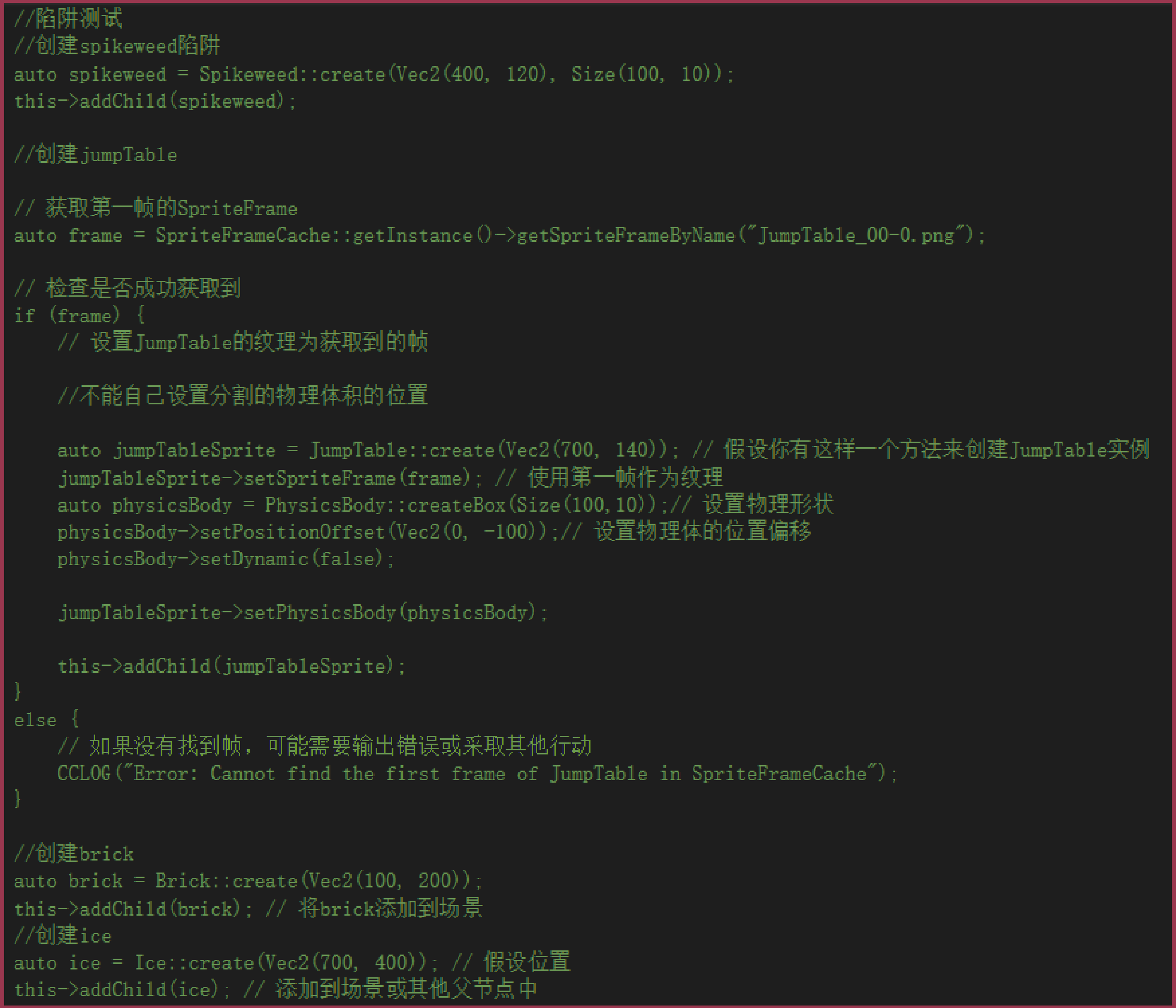
```
//检查玩家是否在坚实的地面上
bool onGround = isOnSolidGround();
setOnGround(onGround);
if (onGround) {
    canDash = 1;
}


/* ... */
// ...


//蹲姿
if (keyStates[PlayerKey::DOWN] && isOnGround&&!keyStates[PlayerKey::DASH] && !keyStates[PlayerKey::CLIMB]&&velocity.x==0){
    changeState(PlayerState::CROUCH);
    CCLOG("Player state changed to CROUCH");
    return;
}
//向上看
if (keyStates[PlayerKey::UP] && isOnGround && !keyStates[PlayerKey::DASH]&&!keyStates[PlayerKey::DOWN]&&!keyStates[PlayerKey::CLIMB] && velocity.x == 0) {
    changeState(PlayerState::LOOKUP);
    CCLOG("Player state changed to LOOKUP");
    return;
}

//冲刺(默认)
if (keyStates[PlayerKey::DASH] && canDash  ) {
    this->getPhysicsBody()->setGravityEnable(false);
    canDash = 0;
    velocity.x = 0; velocity.y = 0;
    this->getPhysicsBody()->setVelocity(Vec2(0, 0));
    changeState(PlayerState::DASH);
    CCLOG("Player state changed to DASH");
    return;
}


//落地
if ((previousState == PlayerState::DROP && isOnGround)||currentState== PlayerState::LANDING) {
    changeState(PlayerState::LANDING);
    //CCLOG("Player state changed to LANDING");
    return;
}
//推墙
if (((keyStates[PlayerKey::RIGHT] && velocity.x > 0) || (keyStates[PlayerKey::LEFT] && velocity.x < 0)) && isOnGround && canClimb &&!keyStates[PlayerKey::CLIMB]) {
    changeState(PlayerState::PUSHWALL);
    CCLOG("Player state changed to PUSHWALL");
    return;
}
//下落

if (!isOnGround && verticalVelocity < 0) {
    changeState(PlayerState::DROP);
    return;
```



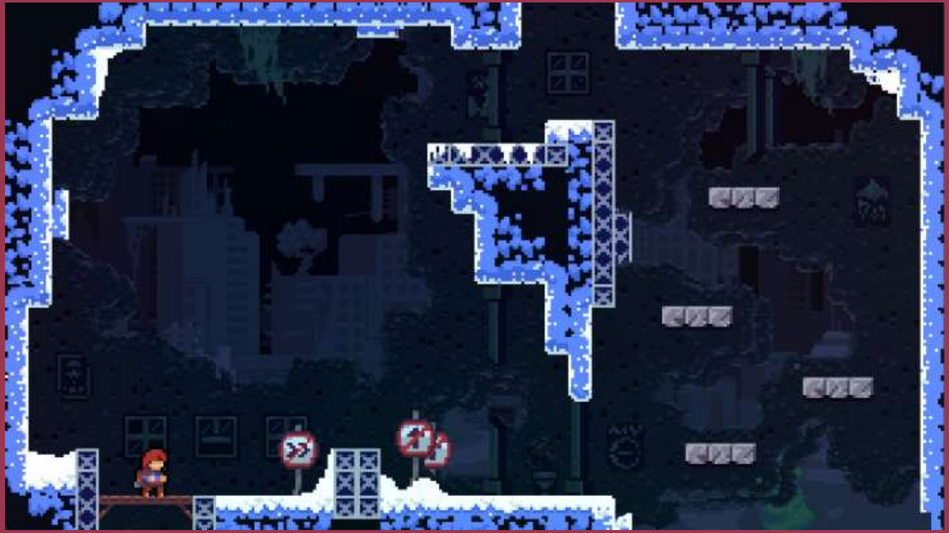
```
    //状态转换
    if (isOnGround) {
        if (velocity.x == 0) {
            changeState(PlayerState::IDLE);
            //CCLOG("Player state changed to IDLE");
            return;
        }
        else if (velocity.x > 0) {
            if (keyStates[PlayerKey::RIGHT]) {
                changeState(PlayerState::MOVING_RIGHT);
                facingDirection = 1;
                return;
            }
            else if(keyStates[PlayerKey::LEFT]){
                changeState(PlayerState::MOVING_TURN_RL);
                facingDirection = -1;
                return;
            }
        }
        else if (velocity.x < 0) {
            if (keyStates[PlayerKey::LEFT]) {
                changeState(PlayerState::MOVING_LEFT);
                facingDirection = -1;
                return;
            }
            else if (keyStates[PlayerKey::RIGHT]) {
                changeState(PlayerState::MOVING_TURN_LR);
                facingDirection = 1;
                return;
            }
        }
    }



    // ... 其他代码 ...
}
```

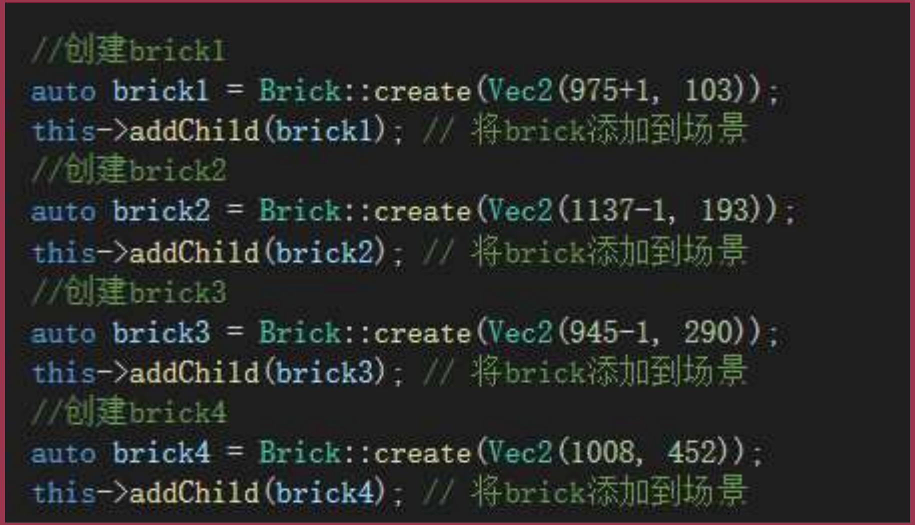
面向对象编程
OOP

OOP



```
//陷阱测试
//创建spikeweed陷阱
auto spikeweed = Spikeweed::create(Vec2(400, 120), Size(100, 10));
this->addChild(spikeweed);

//创建jumpTable

// 获取第一帧的SpriteFrame
auto frame = SpriteFrameCache::getInstance()->getSpriteFrameByName("JumpTable_00-0.png");

// 检查是否成功获取到
if (frame) {
    // 设置JumpTable的纹理为获取到的帧

    //不能自己设置分割的物理体积的位置

    auto jumpTableSprite = JumpTable::create(Vec2(700, 140)); // 假设你有这样一个方法来创建JumpTable实例
    jumpTableSprite->setSpriteFrame(frame); // 使用第一帧作为纹理
    auto physicsBody = PhysicsBody::createBox(Size(100,10));// 设置物理形状
    physicsBody->setPositionOffset(Vec2(0, -100));// 设置物理体的位置偏移
    physicsBody->setDynamic(false);

    jumpTableSprite->setPhysicsBody(physicsBody);

    this->addChild(jumpTableSprite);
}
else {
    // 如果没有找到帧，可能需要输出错误或采取其他行动
    CCLOG("Error: Cannot find the first frame of JumpTable in SpriteFrameCache");
}

//创建brick
auto brick = Brick::create(Vec2(100, 200));
this->addChild(brick); // 将brick添加到场景
//创建ice
auto ice = Ice::create(Vec2(700, 400)); // 假设位置
this->addChild(ice); // 添加到场景或其他父节点中
```





```
//创建brick1
auto brick1 = Brick::create(Vec2(975+1, 103));
this->addChild(brick1); // 将brick添加到场景
//创建brick2
auto brick2 = Brick::create(Vec2(1137-1, 193));
this->addChild(brick2); // 将brick添加到场景
//创建brick3
auto brick3 = Brick::create(Vec2(945-1, 290));
this->addChild(brick3); // 将brick添加到场景
//创建brick4
auto brick4 = Brick::create(Vec2(1008, 452));
this->addChild(brick4); // 将brick添加到场景
```

Special Thanks:

Senior Cai Zhengte, for his explanation of the Cocos engine and advice on the initial framework setup of the project.

Bilibili UP, kimdor, for the inspiration his videos provided for this project and his selfless sharing of materials.

Disclaimer: