**Folder .**

**File CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.20)  
  
  
  
# ================== Project meta ==================  
  
set(APP\_NAME Faterial)  
  
project(${APP\_NAME} C CXX)  
  
  
  
# ================== MSVC runtime & common defs ==================  
  
# Ép dùng /MD (MultiThreadedDLL) và /MDd ở Debug để đồng bộ CRT  
  
if (MSVC)  
  
 # VS 2019+ khuyến nghị dùng biến này thay vì chỉnh flags thủ công  
  
 set(CMAKE\_MSVC\_RUNTIME\_LIBRARY "MultiThreaded$<$<CONFIG:Debug>:Debug>DLL" CACHE STRING "" FORCE)  
  
  
  
 # Một số thư viện cũ gọi symbol stdio cũ => cần legacy\_stdio\_definitions khi link  
  
 # Tránh các warning về "unsafe" CRT  
  
 add\_compile\_definitions(\_CRT\_SECURE\_NO\_WARNINGS)  
  
  
  
 # Tránh macro min/max của Windows.h chồng lấn std::min/std::max  
  
 # (Engine có thể include Windows.h; định nghĩa này ở mức build là an toàn)  
  
 add\_compile\_definitions(NOMINMAX)  
  
  
  
 # Tùy chọn: bật tối ưu compile song song khi dùng MSBuild từ cmake --build  
  
 add\_compile\_options(/MP)  
  
endif()  
  
  
  
# ================== Cocos2d-x engine ==================  
  
# Cấu hình đường dẫn engine tích hợp trong repo  
  
if(NOT DEFINED BUILD\_ENGINE\_DONE)  
  
 set(COCOS2DX\_ROOT\_PATH ${CMAKE\_CURRENT\_SOURCE\_DIR}/cocos2d)  
  
 set(CMAKE\_MODULE\_PATH ${COCOS2DX\_ROOT\_PATH}/cmake/Modules/)  
  
 include(CocosBuildSet)  
  
 # Build engine thành static lib theo preset của Cocos  
  
 add\_subdirectory(${COCOS2DX\_ROOT\_PATH}/cocos ${ENGINE\_BINARY\_PATH}/cocos/core)  
  
endif()  
  
  
  
# ================== Game headers & sources ==================  
  
# Giữ cấu trúc Classes theo bạn đang dùng; có thể chia subfolders sau  
  
set(GAME\_HEADER  
  
 Classes/AppDelegate.h  
  
 Classes/MenuScene.h  
  
 Classes/GameScene.h  
  
  
  
 # Physics (unifier)  
  
 Classes/PhysicsCategories.h  
  
 Classes/PhysicsTags.h  
  
  
  
 # Entities  
  
 Classes/Player.h  
  
 Classes/Enemy.h  
  
 Classes/Goomba.h  
  
 Classes/Spiker.h  
  
 Classes/BossGolem.h  
  
 Classes/Coin.h  
  
 Classes/Crate.h  
  
 Classes/Star.h  
  
 Classes/Upgrade.h  
  
 Classes/Gate.h  
  
 Classes/PressurePlate.h  
  
)  
  
  
  
set(GAME\_SRC  
  
 Classes/AppDelegate.cpp  
  
 Classes/MenuScene.cpp  
  
 Classes/GameScene.cpp  
  
  
  
 # Player  
  
 Classes/Player.cpp  
  
  
  
 # Entities impl  
  
 Classes/Enemy.cpp  
  
 Classes/Goomba.cpp  
  
 Classes/Spiker.cpp  
  
 Classes/BossGolem.cpp  
  
 Classes/Coin.cpp  
  
 Classes/Crate.cpp  
  
 Classes/Star.cpp  
  
 Classes/Upgrade.cpp  
  
 Classes/Gate.cpp  
  
 Classes/PressurePlate.cpp  
  
)  
  
  
  
# ================== Resources ==================  
  
set(GAME\_RES\_FOLDER "${CMAKE\_CURRENT\_SOURCE\_DIR}/Resources")  
  
if(APPLE OR WINDOWS)  
  
 cocos\_mark\_multi\_resources(common\_res\_files RES\_TO "Resources" FOLDERS ${GAME\_RES\_FOLDER})  
  
endif()  
  
  
  
# ================== Platform glue ==================  
  
if(ANDROID)  
  
 list(APPEND GAME\_SRC proj.android/app/jni/hellocpp/main.cpp)  
  
elseif(LINUX)  
  
 list(APPEND GAME\_SRC proj.linux/main.cpp)  
  
elseif(WINDOWS)  
  
 list(APPEND GAME\_HEADER  
  
 proj.win32/main.h  
  
 proj.win32/resource.h  
  
 proj.win32/CrashGuard.h # header-only util (nếu có)  
  
 )  
  
 list(APPEND GAME\_SRC  
  
 proj.win32/main.cpp  
  
 proj.win32/game.rc  
  
 ${common\_res\_files}  
  
 )  
  
elseif(APPLE)  
  
 if(IOS)  
  
 list(APPEND GAME\_HEADER  
  
 proj.ios\_mac/ios/AppController.h  
  
 proj.ios\_mac/ios/RootViewController.h  
  
 )  
  
 set(APP\_UI\_RES  
  
 proj.ios\_mac/ios/LaunchScreen.storyboard  
  
 proj.ios\_mac/ios/LaunchScreenBackground.png  
  
 proj.ios\_mac/ios/Images.xcassets  
  
 )  
  
 list(APPEND GAME\_SRC  
  
 proj.ios\_mac/ios/main.m  
  
 proj.ios\_mac/ios/AppController.mm  
  
 proj.ios\_mac/ios/RootViewController.mm  
  
 proj.ios\_mac/ios/Prefix.pch  
  
 ${APP\_UI\_RES}  
  
 )  
  
 elseif(MACOSX)  
  
 set(APP\_UI\_RES  
  
 proj.ios\_mac/mac/Icon.icns  
  
 proj.ios\_mac/mac/Info.plist  
  
 )  
  
 list(APPEND GAME\_SRC  
  
 proj.ios\_mac/mac/main.cpp  
  
 proj.ios\_mac/mac/Prefix.pch  
  
 ${APP\_UI\_RES}  
  
 )  
  
 endif()  
  
 list(APPEND GAME\_SRC ${common\_res\_files})  
  
endif()  
  
  
  
# ================== Target ==================  
  
set(all\_code\_files ${GAME\_HEADER} ${GAME\_SRC})  
  
  
  
if(NOT ANDROID)  
  
 # Dự án desktop/mobile bình thường => exe  
  
 add\_executable(${APP\_NAME} ${all\_code\_files})  
  
else()  
  
 # Android cần shared lib  
  
 add\_library(${APP\_NAME} SHARED ${all\_code\_files})  
  
 add\_subdirectory(${COCOS2DX\_ROOT\_PATH}/cocos/platform/android ${ENGINE\_BINARY\_PATH}/cocos/platform)  
  
 target\_link\_libraries(${APP\_NAME} PRIVATE -Wl,--whole-archive cpp\_android\_spec -Wl,--no-whole-archive)  
  
endif()  
  
  
  
# Link engine  
  
target\_link\_libraries(${APP\_NAME} PRIVATE cocos2d)  
  
  
  
# Include paths game  
  
target\_include\_directories(${APP\_NAME}  
  
 PRIVATE Classes  
  
 PRIVATE ${COCOS2DX\_ROOT\_PATH}/cocos/audio/include/  
  
)  
  
  
  
# ================== Windows: CRT & system libs ==================  
  
if (MSVC)  
  
 # Tránh kéo LIBCMT (CRT tĩnh) khi ta đã dùng /MD cho toàn bộ app  
  
 target\_link\_options(${APP\_NAME} PRIVATE /NODEFAULTLIB:LIBCMT)  
  
  
  
 # Bổ sung các import-lib CRT hiện đại để resolve \_\_imp\_\_xxx và handler nội bộ  
  
 target\_link\_libraries(${APP\_NAME} PRIVATE  
  
 DbgHelp  
  
 ucrt  
  
 vcruntime  
  
 msvcrt  
  
 legacy\_stdio\_definitions  
  
 # Quan trọng: cho timeBeginPeriod/timeEndPeriod (engine gọi trong CCApplication-win32)  
  
 winmm  
  
 )  
  
  
  
 # Giảm ồn cảnh báo LNK4098 nếu toolchain in ra (mismatch defaultlib)  
  
 target\_link\_options(${APP\_NAME} PRIVATE /IGNORE:4098)  
  
  
  
 # Nếu bạn đang build Win32, có thể bật /SAFESEH:NO khi có lib bên thứ ba thiếu SEH table  
  
 # target\_link\_options(${APP\_NAME} PRIVATE /SAFESEH:NO)  
  
endif()  
  
  
  
# ================== App config & copy resources ==================  
  
setup\_cocos\_app\_config(${APP\_NAME})  
  
  
  
if(APPLE)  
  
 set\_target\_properties(${APP\_NAME} PROPERTIES RESOURCE "${APP\_UI\_RES}")  
  
 if(MACOSX)  
  
 set\_xcode\_property(${APP\_NAME} INFOPLIST\_FILE "${CMAKE\_CURRENT\_SOURCE\_DIR}/proj.ios\_mac/mac/Info.plist")  
  
 elseif(IOS)  
  
 set\_xcode\_property(${APP\_NAME} INFOPLIST\_FILE "${CMAKE\_CURRENT\_SOURCE\_DIR}/proj.ios\_mac/ios/Info.plist")  
  
 set\_xcode\_property(${APP\_NAME} ASSETCATALOG\_COMPILER\_APPICON\_NAME "AppIcon")  
  
 endif()  
  
elseif(WINDOWS)  
  
 cocos\_copy\_target\_dll(${APP\_NAME})  
  
endif()  
  
  
  
if(LINUX OR WINDOWS)  
  
 cocos\_get\_resource\_path(APP\_RES\_DIR ${APP\_NAME})  
  
 cocos\_copy\_target\_res(${APP\_NAME} LINK\_TO ${APP\_RES\_DIR} FOLDERS ${GAME\_RES\_FOLDER})  
  
endif()  
  
  
  
# ================== Quality-of-life (tùy chọn) ==================  
  
# Cờ kiểm tra C++ tiêu chuẩn (nếu bạn muốn ép C++17)  
  
# set\_property(TARGET ${APP\_NAME} PROPERTY CXX\_STANDARD 17)  
  
# set\_property(TARGET ${APP\_NAME} PROPERTY CXX\_STANDARD\_REQUIRED ON)

**Folder Classes**

**File main.cpp**

#include "core/AppDelegate.h"  
  
#include "cocos2d.h"  
  
  
  
USING\_NS\_CC;  
  
  
  
int main(int argc, char\*\* argv) {  
  
 AppDelegate app;  
  
 return Application::getInstance()->run();  
  
}

**Folder Classes/bosses**

**File BossGolem.cpp**

// BossGolem.cpp  
  
#include "game/bosses/BossGolem.h"  
  
USING\_NS\_CC;  
  
BossGolem\* BossGolem::create(){ auto p=new(std::nothrow) BossGolem(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool BossGolem::init(){  
  
 if(!Enemy::init()) return false;  
  
 setTextureRect(Rect(0,0,72,72));  
  
 setColor(Color3B(90,70,50));  
  
 \_hp = 20; \_speed = 60.f;  
  
 return true;  
  
}  
  
void BossGolem::takeHit(int dmg){  
  
 \_hp -= dmg;  
  
 setColor((\_hp%2==0)?Color3B(120,90,60):Color3B(90,70,50));  
  
 if(\_hp<=0) removeFromParent();  
  
}

**File BossGolem.h**

// BossGolem.h  
  
#pragma once  
  
#include "game/Enemy.h"  
  
class BossGolem : public Enemy {  
  
public:  
  
 static BossGolem\* create();  
  
 bool init() override;  
  
 void takeHit(int dmg) override; // trụ hơn  
  
};

**Folder Classes/core**

**File AppDelegate.cpp**

// AppDelegate.cpp  
  
#include "core/AppDelegate.h"  
  
#include "base/CCDirector.h"  
  
#include "platform/CCGLView.h"  
  
#include "scenes/MenuScene.h"  
  
USING\_NS\_CC;  
  
  
  
AppDelegate::AppDelegate() {}  
  
AppDelegate::~AppDelegate() {}  
  
  
  
void AppDelegate::initGLContextAttrs(){  
  
 GLContextAttrs a{8,8,8,8,24,8};  
  
 GLView::setGLContextAttrs(a);  
  
}  
  
  
  
bool AppDelegate::applicationDidFinishLaunching(){  
  
 auto\* director = Director::getInstance();  
  
 auto\* glview = director->getOpenGLView();  
  
 if(!glview){  
  
#if (CC\_TARGET\_PLATFORM == CC\_PLATFORM\_WIN32)  
  
 glview = GLViewImpl::createWithRect("MyGame", Rect(0,0,1280,720));  
  
#else  
  
 glview = GLViewImpl::create("MyGame");  
  
#endif  
  
 director->setOpenGLView(glview);  
  
 }  
  
 director->setAnimationInterval(1.0f/60.0f);  
  
#if COCOS2D\_DEBUG  
  
 director->setDisplayStats(true);  
  
#endif  
  
 glview->setDesignResolutionSize(1280, 720, ResolutionPolicy::NO\_BORDER);  
  
 director->runWithScene(MenuScene::createScene());  
  
 return true;  
  
}  
  
void AppDelegate::applicationDidEnterBackground(){ Director::getInstance()->stopAnimation(); }  
  
void AppDelegate::applicationWillEnterForeground(){ Director::getInstance()->startAnimation(); }

**File AppDelegate.h**

// AppDelegate.h  
  
#pragma once  
  
#include "cocos2d.h"  
  
class AppDelegate : private cocos2d::Application {  
  
public:  
  
 AppDelegate();  
  
 virtual ~AppDelegate();  
  
 virtual void initGLContextAttrs();  
  
 virtual bool applicationDidFinishLaunching();  
  
 virtual void applicationDidEnterBackground();  
  
 virtual void applicationWillEnterForeground();  
  
};

**File CrashGuard.h**

#pragma once  
  
#include <cstdio>  
  
#define CG\_LOGF(path, fmt, ...) \  
  
 do { FILE\* f=nullptr; fopen\_s(&f, (path), "a"); if(f){std::fprintf(f, fmt "\n", ##\_\_VA\_ARGS\_\_); std::fclose(f);} } while(0)

**Folder Classes/enemies**

**File Goomba.cpp**

// Goomba.cpp  
  
#include "game/enemies/Goomba.h"  
  
USING\_NS\_CC;  
  
Goomba\* Goomba::create(){ auto p=new(std::nothrow) Goomba(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Goomba::init(){  
  
 if(!Enemy::init()) return false;  
  
 setColor(Color3B(200,120,60));  
  
 return true;  
  
}

**File Goomba.h**

// Goomba.h  
  
#pragma once  
  
#include "game/Enemy.h"  
  
class Goomba : public Enemy {  
  
public:  
  
 static Goomba\* create();  
  
 bool init() override;  
  
};

**File Spiker.cpp**

// Spiker.cpp  
  
#include "game/enemies/Spiker.h"  
  
USING\_NS\_CC;  
  
Spiker\* Spiker::create(){ auto p=new(std::nothrow) Spiker(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Spiker::init(){  
  
 if(!Enemy::init()) return false;  
  
 setColor(Color3B(180,200,255));  
  
 return true;  
  
}

**File Spiker.h**

// Spiker.h  
  
#pragma once  
  
#include "game/Enemy.h"  
  
class Spiker : public Enemy {  
  
public:  
  
 static Spiker\* create();  
  
 bool init() override;  
  
};

**Folder Classes/game**

**File Enemy.cpp**

// Enemy.cpp  
  
#include "game/Enemy.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
  
  
Enemy\* Enemy::create(){ auto p=new(std::nothrow) Enemy(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Enemy::init(){  
  
 if(!Sprite::init()) return false;  
  
 setTextureRect(Rect(0,0,42,42));  
  
 setColor(Color3B::RED);  
  
 \_gfx = this;  
  
 scheduleUpdate();  
  
 return true;  
  
}  
  
void Enemy::enablePhysics(const Vec2& pos, const Size& sz){  
  
 if(\_body) return;  
  
 \_body = PhysicsBody::createBox(sz, PhysicsMaterial(0.1f,0,0.9f));  
  
 \_body->setDynamic(true);  
  
 \_body->setRotationEnable(false);  
  
 \_body->setCategoryBitmask(phys::CAT\_ENEMY);  
  
 \_body->setCollisionBitmask(phys::CAT\_WORLD|phys::CAT\_PLAYER|phys::CAT\_CRATE);  
  
 \_body->setContactTestBitmask(phys::all());  
  
 setPhysicsBody(\_body);  
  
 setPosition(pos);  
  
}  
  
void Enemy::setPatrol(const Vec2& a, const Vec2& b){ \_pA=a; \_pB=b; \_right = (b.x>=a.x); }  
  
void Enemy::takeHit(int dmg){ \_hp -= dmg; if(\_hp<=0) removeFromParent(); }  
  
void Enemy::\_stepPatrol(float dt){  
  
 if(!\_body) return;  
  
 float dir = \_right ? 1.f : -1.f;  
  
 auto v = \_body->getVelocity();  
  
 v.x = dir \* \_speed;  
  
 \_body->setVelocity(v);  
  
 float x = getPositionX();  
  
 if(\_right && x >= \_pB.x) \_right=false;  
  
 else if(!\_right && x <= \_pA.x) \_right=true;  
  
}  
  
void Enemy::update(float dt){ \_stepPatrol(dt); }

**File Enemy.h**

// Enemy.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
  
  
class Enemy : public cocos2d::Sprite {  
  
public:  
  
 static Enemy\* create();  
  
 bool init() override;  
  
  
  
 // API mà code cũ từng dùng  
  
 void setPatrol(const cocos2d::Vec2& a, const cocos2d::Vec2& b);  
  
 void setSpeed(float s){ \_speed = s; }  
  
 virtual void takeHit(int dmg);  
  
  
  
 void enablePhysics(const cocos2d::Vec2& pos, const cocos2d::Size& sz);  
  
  
  
 void update(float dt) override;  
  
  
  
protected:  
  
 cocos2d::PhysicsBody\* \_body = nullptr;  
  
 cocos2d::Sprite\* \_gfx = nullptr;  
  
 float \_speed = 80.f;  
  
 int \_hp = 3;  
  
 bool \_right = true; // để thỏa mãn tham chiếu cũ  
  
 cocos2d::Vec2 \_pA{0,0}, \_pB{200,0};  
  
  
  
 void \_stepPatrol(float dt);  
  
};

**File Entity.cpp**

// Entity.cpp  
  
#include "game/Entity.h"

**File Entity.h**

// Entity.h  
  
#pragma once  
  
#include "cocos2d.h"  
  
#include "physics/PhysicsDefs.h"  
  
  
  
class Entity : public cocos2d::Node {  
  
public:  
  
 CREATE\_FUNC(Entity);  
  
 bool init() override { return Node::init(); }  
  
 void setTagEx(phys::Tag t){ \_gtag = t; setTag(static\_cast<int>(t)); }  
  
 phys::Tag getTagEx() const { return \_gtag; }  
  
protected:  
  
 phys::Tag \_gtag = phys::Tag::NONE;  
  
};

**File Player.cpp**

// Player.cpp  
  
#include "game/Player.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
  
  
Player\* Player::create(){ auto p=new(std::nothrow) Player(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Player::init(){  
  
 if(!Sprite::init()) return false;  
  
 setTextureRect(Rect(0,0,40,56));  
  
 setColor(Color3B::BLUE);  
  
 auto dn=DrawNode::create(); Vec2 r[4]={{-20,-28},{20,-28},{20,28},{-20,28}}; dn->drawPoly(r,4,true,Color4F::WHITE); addChild(dn);  
  
 scheduleUpdate(); return true;  
  
}  
  
void Player::enablePhysics(const Vec2& pos){  
  
 if(\_body) return;  
  
 \_body = PhysicsBody::createBox(Size(40,56), PhysicsMaterial(0.1f,0,0.9f));  
  
 \_body->setDynamic(true); \_body->setRotationEnable(false);  
  
 \_body->setCategoryBitmask(phys::CAT\_PLAYER);  
  
 \_body->setCollisionBitmask(phys::CAT\_WORLD|phys::CAT\_ENEMY|phys::CAT\_ITEM|phys::CAT\_GATE|phys::CAT\_CRATE);  
  
 \_body->setContactTestBitmask(phys::all());  
  
 setPhysicsBody(\_body); setPosition(pos);  
  
}  
  
void Player::setMoveDir(const Vec2& dir){ \_moveDir=dir; }  
  
void Player::jump(){  
  
 if(\_onGround && \_body){ \_body->setVelocity({ \_body->getVelocity().x, 0}); \_body->applyImpulse({0, 500}); \_onGround=false; }  
  
}  
  
void Player::\_syncGroundState(){ if(!\_body) return; \_onGround = std::abs(\_body->getVelocity().y) < 0.1f; }  
  
void Player::update(float){ if(!\_body) return; auto v=\_body->getVelocity(); v.x = \_moveDir.x \* \_speed; \_body->setVelocity(v); \_syncGroundState(); }

**File Player.h**

// Player.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
  
  
class Player : public cocos2d::Sprite {  
  
public:  
  
 static Player\* create();  
  
 bool init() override;  
  
 void enablePhysics(const cocos2d::Vec2& pos);  
  
 void setMoveDir(const cocos2d::Vec2& dir);  
  
 void jump();  
  
 void update(float dt) override;  
  
private:  
  
 cocos2d::PhysicsBody\* \_body=nullptr;  
  
 cocos2d::Vec2 \_moveDir{0,0};  
  
 float \_speed=220.f;  
  
 bool \_onGround=false;  
  
 void \_syncGroundState();  
  
};

**Folder Classes/objects**

**File Coin.cpp**

// Coin.cpp  
  
#include "game/objects/Coin.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
Coin\* Coin::create(){ auto p=new(std::nothrow) Coin(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Coin::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::COIN);  
  
 auto dn = DrawNode::create();  
  
 dn->drawSolidCircle(Vec2::ZERO, 10, 0, 24, Color4F(1,0.85f,0,1));  
  
 addChild(dn);  
  
 auto body = PhysicsBody::createCircle(10);  
  
 body->setDynamic(false);  
  
 body->setCategoryBitmask(phys::CAT\_ITEM);  
  
 body->setCollisionBitmask(0); // sensor-only  
  
 body->setContactTestBitmask(phys::CAT\_PLAYER);  
  
 setPhysicsBody(body);  
  
 return true;  
  
}

**File Coin.h**

// Coin.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class Coin : public Entity {  
  
public:  
  
 static Coin\* create();  
  
 bool init() override;  
  
};

**File Crate.cpp**

// Crate.cpp  
  
#include "game/objects/Crate.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
Crate\* Crate::create(){ auto p=new(std::nothrow) Crate(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Crate::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::CRATE);  
  
 auto dn = DrawNode::create();  
  
 dn->drawSolidRect({-16,-16},{16,16}, Color4F(0.6f,0.4f,0.2f,1));  
  
 addChild(dn);  
  
 auto b = PhysicsBody::createBox(Size(32,32), PhysicsMaterial(0.3f,0,0.9f));  
  
 b->setDynamic(true);  
  
 b->setCategoryBitmask(phys::CAT\_CRATE);  
  
 b->setCollisionBitmask(phys::CAT\_WORLD|phys::CAT\_PLAYER|phys::CAT\_ENEMY|phys::CAT\_CRATE);  
  
 b->setContactTestBitmask(phys::all());  
  
 setPhysicsBody(b);  
  
 return true;  
  
}

**File Crate.h**

// Crate.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class Crate : public Entity {  
  
public:  
  
 static Crate\* create();  
  
 bool init() override;  
  
};

**File Gate.cpp**

// Gate.cpp  
  
#include "game/objects/Gate.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
Gate\* Gate::create(){ auto p=new(std::nothrow) Gate(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Gate::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::GATE);  
  
 \_dn = DrawNode::create();  
  
 \_dn->drawSolidRect({-12,-40},{12,40}, Color4F(0.7f,0.7f,0.9f,1));  
  
 addChild(\_dn);  
  
 auto b = PhysicsBody::createBox(Size(24,80));  
  
 b->setDynamic(false);  
  
 b->setCategoryBitmask(phys::CAT\_GATE);  
  
 b->setCollisionBitmask(phys::CAT\_WORLD|phys::CAT\_PLAYER|phys::CAT\_ENEMY|phys::CAT\_CRATE);  
  
 b->setContactTestBitmask(phys::all());  
  
 setPhysicsBody(b);  
  
 return true;  
  
}  
  
void Gate::open(bool on){  
  
 \_open = on;  
  
 if(auto\* b=getPhysicsBody()){  
  
 b->setEnabled(!on);  
  
 }  
  
 \_dn->clear();  
  
 if(on) \_dn->drawRect({-12,-40},{12,40}, Color4F(0.2f,1,0.2f,1)); // khung trống  
  
 else \_dn->drawSolidRect({-12,-40},{12,40}, Color4F(0.7f,0.7f,0.9f,1));  
  
}

**File Gate.h**

// Gate.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class Gate : public Entity {  
  
public:  
  
 static Gate\* create();  
  
 bool init() override;  
  
 void open(bool on);  
  
 bool isOpen() const { return \_open; }  
  
private:  
  
 bool \_open=false;  
  
 cocos2d::DrawNode\* \_dn=nullptr;  
  
};

**File PressurePlate.cpp**

// PressurePlate.cpp  
  
#include "game/objects/PressurePlate.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
PressurePlate\* PressurePlate::create(){ auto p=new(std::nothrow) PressurePlate(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool PressurePlate::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::PLATE);  
  
 auto dn = DrawNode::create();  
  
 dn->drawSolidRect({-18,-4},{18,4}, Color4F(0.9f,0.4f,0.4f,1));  
  
 addChild(dn);  
  
 auto b = PhysicsBody::createBox(Size(36,8));  
  
 b->setDynamic(false);  
  
 b->setCategoryBitmask(phys::CAT\_SENSOR);  
  
 b->setCollisionBitmask(0);  
  
 b->setContactTestBitmask(phys::CAT\_PLAYER|phys::CAT\_CRATE);  
  
 setPhysicsBody(b);  
  
 return true;  
  
}

**File PressurePlate.h**

// PressurePlate.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class PressurePlate : public Entity {  
  
public:  
  
 static PressurePlate\* create();  
  
 bool init() override;  
  
 void setCallback(const std::function<void(bool)>& cb){ \_cb = cb; }  
  
private:  
  
 std::function<void(bool)> \_cb;  
  
 int \_pressCount = 0;  
  
};

**File Star.cpp**

// Star.cpp  
  
#include "game/objects/Star.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
Star\* Star::create(){ auto p=new(std::nothrow) Star(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Star::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::STAR);  
  
 auto dn = DrawNode::create();  
  
 Vec2 pts[5];  
  
 for(int i=0;i<5;++i){ float a = CC\_DEGREES\_TO\_RADIANS(72\*i-90); pts[i]=Vec2(0,14).rotateByAngle(Vec2::ZERO,a); }  
  
 dn->drawPoly(pts,5,true,Color4F(1,1,0.2f,1));  
  
 addChild(dn);  
  
 auto body = PhysicsBody::createCircle(12);  
  
 body->setDynamic(false);  
  
 body->setCategoryBitmask(phys::CAT\_ITEM);  
  
 body->setCollisionBitmask(0);  
  
 body->setContactTestBitmask(phys::CAT\_PLAYER);  
  
 setPhysicsBody(body);  
  
 return true;  
  
}

**File Star.h**

// Star.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class Star : public Entity {  
  
public:  
  
 static Star\* create();  
  
 bool init() override;  
  
};

**File Upgrade.cpp**

// Upgrade.cpp  
  
#include "game/objects/Upgrade.h"  
  
#include "physics/PhysicsDefs.h"  
  
USING\_NS\_CC;  
  
Upgrade\* Upgrade::create(){ auto p=new(std::nothrow) Upgrade(); if(p && p->init()){p->autorelease(); return p;} CC\_SAFE\_DELETE(p); return nullptr; }  
  
bool Upgrade::init(){  
  
 if(!Entity::init()) return false;  
  
 setTagEx(phys::Tag::UPGRADE);  
  
 auto dn = DrawNode::create();  
  
 dn->drawSolidRect({-10,-10},{10,10}, Color4F(0.2f,1,1,1));  
  
 addChild(dn);  
  
 auto body = PhysicsBody::createBox(Size(20,20));  
  
 body->setDynamic(false);  
  
 body->setCategoryBitmask(phys::CAT\_ITEM);  
  
 body->setCollisionBitmask(0);  
  
 body->setContactTestBitmask(phys::CAT\_PLAYER);  
  
 setPhysicsBody(body);  
  
 return true;  
  
}

**File Upgrade.h**

// Upgrade.h  
  
#pragma once  
  
#include "game/Entity.h"  
  
class Upgrade : public Entity {  
  
public:  
  
 static Upgrade\* create();  
  
 bool init() override;  
  
};

**Folder Classes/physics**

**File PhysicsDefs.h**

#pragma once  
  
#include "cocos2d.h"  
  
#include <cstdint>  
  
  
  
namespace phys {  
  
 using Mask = uint32\_t;  
  
  
  
 enum : Mask {  
  
 CAT\_WORLD = 0x0001,  
  
 CAT\_PLAYER = 0x0002,  
  
 CAT\_ENEMY = 0x0004,  
  
 CAT\_ITEM = 0x0008,  
  
 CAT\_GATE = 0x0010,  
  
 CAT\_CRATE = 0x0020,  
  
 CAT\_SENSOR = 0x0040  
  
 };  
  
 inline Mask all() { return 0xFFFFFFFFu; }  
  
  
  
 // Game Tag (thay cho GTag cũ)  
  
 enum class Tag : int {  
  
 NONE = 0,  
  
 WORLD,  
  
 PLAYER,  
  
 ENEMY,  
  
 ITEM,  
  
 GATE,  
  
 CRATE,  
  
 PLATE,  
  
 BOSS,  
  
 STAR,  
  
 COIN,  
  
 UPGRADE  
  
 };  
  
}

**Folder Classes/scenes**

**File GameScene.cpp**

// GameScene.cpp  
  
#include "scenes/GameScene.h"  
  
#include "ui/HUDLayer.h"  
  
#include "game/Player.h"  
  
#include "game/enemies/Goomba.h"  
  
#include "game/enemies/Spiker.h"  
  
#include "game/bosses/BossGolem.h"  
  
#include "game/objects/Coin.h"  
  
#include "game/objects/Star.h"  
  
#include "game/objects/Upgrade.h"  
  
#include "game/objects/Crate.h"  
  
#include "game/objects/Gate.h"  
  
#include "game/objects/PressurePlate.h"  
  
#include "physics/PhysicsDefs.h"  
  
  
  
USING\_NS\_CC;  
  
  
  
Scene\* GameScene::createScene(){ return GameScene::create(); }  
  
  
  
bool GameScene::init(){  
  
 if(!Scene::initWithPhysics()) return false;  
  
 getPhysicsWorld()->setGravity({0,-980});  
  
#if COCOS2D\_DEBUG  
  
 getPhysicsWorld()->setDebugDrawMask(PhysicsWorld::DEBUGDRAW\_ALL);  
  
#endif  
  
 return true;  
  
}  
  
  
  
void GameScene::onEnter(){  
  
 Scene::onEnter();  
  
 \_setupWorld();  
  
 \_hud = HUDLayer::create(); addChild(\_hud, 10);  
  
 \_spawnDemo();  
  
 \_bindInput();  
  
  
  
 auto cl = EventListenerPhysicsContact::create();  
  
 cl->onContactBegin = CC\_CALLBACK\_1(GameScene::\_onContactBegin, this);  
  
 \_eventDispatcher->addEventListenerWithSceneGraphPriority(cl, this);  
  
}  
  
  
  
void GameScene::\_setupWorld(){  
  
 auto vs=Director::getInstance()->getVisibleSize();  
  
 auto origin=Director::getInstance()->getVisibleOrigin();  
  
  
  
 auto bg = LayerColor::create(Color4B(25,25,32,255)); addChild(bg,-10);  
  
  
  
 // Viền  
  
 auto edge = Node::create();  
  
 auto ebody = PhysicsBody::createEdgeBox(vs, PhysicsMaterial(0.1f,0,1), 2.0f);  
  
 ebody->setCategoryBitmask(phys::CAT\_WORLD);  
  
 ebody->setCollisionBitmask(phys::all());  
  
 ebody->setContactTestBitmask(phys::all());  
  
 edge->setPhysicsBody(ebody);  
  
 edge->setPosition(origin + vs/2);  
  
 addChild(edge);  
  
  
  
 // Nền đất  
  
 auto ground = Node::create();  
  
 auto gBody = PhysicsBody::createBox(Size(vs.width\*0.9f, 24), PhysicsMaterial(0.2f,0,1));  
  
 gBody->setDynamic(false);  
  
 gBody->setCategoryBitmask(phys::CAT\_WORLD);  
  
 gBody->setCollisionBitmask(phys::all());  
  
 gBody->setContactTestBitmask(phys::all());  
  
 ground->setPhysicsBody(gBody);  
  
 ground->setPosition(origin + Vec2(vs.width/2, vs.height\*0.18f));  
  
 auto dn = DrawNode::create();  
  
 dn->drawSolidRect({-vs.width\*0.45f,-12},{vs.width\*0.45f,12}, Color4F(0.15f,0.8f,0.25f,1));  
  
 ground->addChild(dn);  
  
 addChild(ground);  
  
}  
  
  
  
void GameScene::\_spawnDemo(){  
  
 auto vs=Director::getInstance()->getVisibleSize();  
  
 auto origin=Director::getInstance()->getVisibleOrigin();  
  
  
  
 // Player  
  
 \_player = Player::create(); addChild(\_player, 1);  
  
 \_player->enablePhysics(origin + Vec2(vs.width\*0.15f, vs.height\*0.25f));  
  
  
  
 // Coin/Star/Upgrade  
  
 for(int i=0;i<4;++i){ auto c=Coin::create(); c->setPosition(origin+Vec2(200+60\*i, 400)); addChild(c); }  
  
 auto s=Star::create(); s->setPosition(origin+Vec2(600, 420)); addChild(s);  
  
 auto u=Upgrade::create(); u->setPosition(origin+Vec2(680, 420)); addChild(u);  
  
  
  
 // Crate + Plate + Gate  
  
 auto crate = Crate::create(); crate->setPosition(origin+Vec2(420, 250)); addChild(crate);  
  
  
  
 \_plate = PressurePlate::create(); \_plate->setPosition(origin+Vec2(520, 240)); addChild(\_plate);  
  
  
  
 \_gate = Gate::create(); \_gate->setPosition(origin+Vec2(840, 280)); addChild(\_gate);  
  
  
  
 // Plate callback mở cổng khi có vật đè  
  
 \_plate->setCallback([this](bool pressed){  
  
 if(\_gate) \_gate->open(pressed);  
  
 });  
  
  
  
 // Goomba  
  
 auto g1 = Goomba::create(); addChild(g1);  
  
 g1->enablePhysics(origin+Vec2(300,260), Size(42,42));  
  
 g1->setPatrol(origin+Vec2(270,260), origin+Vec2(360,260));  
  
 // Spiker  
  
 auto sp = Spiker::create(); addChild(sp);  
  
 sp->enablePhysics(origin+Vec2(500,260), Size(42,42));  
  
 sp->setPatrol(origin+Vec2(480,260), origin+Vec2(560,260));  
  
 // Boss (đặt xa)  
  
 auto boss = BossGolem::create(); addChild(boss);  
  
 boss->enablePhysics(origin+Vec2(1000,300), Size(72,72));  
  
 boss->setPatrol(origin+Vec2(960,300), origin+Vec2(1080,300));  
  
}  
  
  
  
void GameScene::\_bindInput(){  
  
 auto l = EventListenerKeyboard::create();  
  
 l->onKeyPressed = [this](EventKeyboard::KeyCode c, Event\*){  
  
 if(!\_player) return;  
  
 switch(c){  
  
 case EventKeyboard::KeyCode::KEY\_A:  
  
 case EventKeyboard::KeyCode::KEY\_LEFT\_ARROW: \_player->setMoveDir({-1,0}); break;  
  
 case EventKeyboard::KeyCode::KEY\_D:  
  
 case EventKeyboard::KeyCode::KEY\_RIGHT\_ARROW: \_player->setMoveDir({ 1,0}); break;  
  
 case EventKeyboard::KeyCode::KEY\_W:  
  
 case EventKeyboard::KeyCode::KEY\_UP\_ARROW:  
  
 case EventKeyboard::KeyCode::KEY\_SPACE: \_player->jump(); break;  
  
 default: break;  
  
 }  
  
 };  
  
 l->onKeyReleased = [this](EventKeyboard::KeyCode c, Event\*){  
  
 if(!\_player) return;  
  
 if(c==EventKeyboard::KeyCode::KEY\_A || c==EventKeyboard::KeyCode::KEY\_LEFT\_ARROW ||  
  
 c==EventKeyboard::KeyCode::KEY\_D || c==EventKeyboard::KeyCode::KEY\_RIGHT\_ARROW)  
  
 \_player->setMoveDir({0,0});  
  
 };  
  
 \_eventDispatcher->addEventListenerWithSceneGraphPriority(l, this);  
  
}  
  
  
  
bool GameScene::\_match(Node\* n, uint32\_t cat, int tag){  
  
 if(!n || !n->getPhysicsBody()) return false;  
  
 bool catOk = (n->getPhysicsBody()->getCategoryBitmask() & cat) != 0;  
  
 if(!catOk) return false;  
  
 if(tag==0) return true;  
  
 return n->getTag() == tag;  
  
}  
  
  
  
bool GameScene::\_onContactBegin(PhysicsContact& c){  
  
 auto a = c.getShapeA()->getBody()->getNode();  
  
 auto b = c.getShapeB()->getBody()->getNode();  
  
 if(!a || !b) return true;  
  
  
  
 // 1) Player ăn Item (Coin/Star/Upgrade)  
  
 Node\* item=nullptr;  
  
 if(\_match(a, phys::CAT\_PLAYER) && \_match(b, phys::CAT\_ITEM)) item=b;  
  
 else if(\_match(b, phys::CAT\_PLAYER) && \_match(a, phys::CAT\_ITEM)) item=a;  
  
 if(item){  
  
 switch(static\_cast<phys::Tag>(item->getTag())){  
  
 case phys::Tag::COIN: \_score+=1; break;  
  
 case phys::Tag::STAR: \_score+=5; break;  
  
 case phys::Tag::UPGRADE: \_lives+=1; break;  
  
 default: break;  
  
 }  
  
 if(\_hud){ \_hud->setScore(\_score); \_hud->setLives(\_lives); }  
  
 item->removeFromParent();  
  
 return false;  
  
 }  
  
  
  
 // 2) Plate kích hoạt bởi Player/Crate -> mở Gate  
  
 Node\* plate=nullptr; Node\* other=nullptr;  
  
 if(\_match(a, phys::CAT\_SENSOR) && a->getTag()==(int)phys::Tag::PLATE){ plate=a; other=b; }  
  
 else if(\_match(b, phys::CAT\_SENSOR) && b->getTag()==(int)phys::Tag::PLATE){ plate=b; other=a; }  
  
 if(plate && ( \_match(other, phys::CAT\_PLAYER) || \_match(other, phys::CAT\_CRATE) )){  
  
 if(\_plate) \_plate->setCallback([this](bool pressed){ if(\_gate) \_gate->open(pressed); });  
  
 if(\_gate) \_gate->open(true);  
  
 // Hạ “pressCount” đơn giản – ở Drop 1 coi như chỉ có 1 vật đè  
  
 return true;  
  
 }  
  
  
  
 return true;  
  
}

**File GameScene.h**

// GameScene.h  
  
#pragma once  
  
#include "cocos2d.h"  
  
class Player; class HUDLayer; class Gate; class PressurePlate;  
  
  
  
class GameScene : public cocos2d::Scene {  
  
public:  
  
 CREATE\_FUNC(GameScene);  
  
 static cocos2d::Scene\* createScene();  
  
 bool init() override;  
  
 void onEnter() override;  
  
private:  
  
 Player\* \_player=nullptr;  
  
 HUDLayer\* \_hud=nullptr;  
  
 Gate\* \_gate=nullptr;  
  
 PressurePlate\* \_plate=nullptr;  
  
 int \_score=0, \_lives=3;  
  
  
  
 void \_setupWorld();  
  
 void \_spawnDemo();  
  
 void \_bindInput();  
  
 bool \_onContactBegin(cocos2d::PhysicsContact& c);  
  
 bool \_match(cocos2d::Node\* n, uint32\_t cat, int tag = 0);  
  
};

**File MenuScene.cpp**

// MenuScene.cpp  
  
#include "scenes/MenuScene.h"  
  
#include "scenes/GameScene.h"  
  
USING\_NS\_CC;  
  
  
  
Scene\* MenuScene::createScene(){ return MenuScene::create(); }  
  
bool MenuScene::init(){  
  
 if(!Scene::init()) return false;  
  
 auto vs = Director::getInstance()->getVisibleSize();  
  
 auto origin = Director::getInstance()->getVisibleOrigin();  
  
 auto title = Label::createWithSystemFont("MyGame", "Arial", 48);  
  
 title->setPosition(origin + Vec2(vs.width/2, vs.height\*0.65f));  
  
 auto hint = Label::createWithSystemFont("[ENTER] New Game | [ESC] Quit","Arial",24);  
  
 hint->setPosition(origin + Vec2(vs.width/2, vs.height\*0.45f));  
  
 addChild(title); addChild(hint);  
  
 \_bindInput();  
  
 return true;  
  
}  
  
void MenuScene::\_bindInput(){  
  
 auto l = EventListenerKeyboard::create();  
  
 l->onKeyPressed = [this](EventKeyboard::KeyCode c, Event\*){  
  
 if(c==EventKeyboard::KeyCode::KEY\_ENTER || c==EventKeyboard::KeyCode::KEY\_KP\_ENTER) \_goGame();  
  
 else if(c==EventKeyboard::KeyCode::KEY\_ESCAPE) Director::getInstance()->end();  
  
 };  
  
 \_eventDispatcher->addEventListenerWithSceneGraphPriority(l, this);  
  
}  
  
void MenuScene::\_goGame(){  
  
 Director::getInstance()->replaceScene(TransitionFade::create(0.25f, GameScene::createScene()));  
  
}

**File MenuScene.h**

// MenuScene.h  
  
#pragma once  
  
#include "cocos2d.h"  
  
class MenuScene : public cocos2d::Scene {  
  
public:  
  
 CREATE\_FUNC(MenuScene);  
  
 static cocos2d::Scene\* createScene();  
  
 bool init() override;  
  
private:  
  
 void \_bindInput();  
  
 void \_goGame();  
  
};

**Folder Classes/ui**

**File HUDLayer.cpp**

// HUDLayer.cpp  
  
#include "ui/HUDLayer.h"  
  
USING\_NS\_CC;  
  
bool HUDLayer::init(){  
  
 if(!Layer::init()) return false;  
  
 auto vs = Director::getInstance()->getVisibleSize();  
  
 auto origin = Director::getInstance()->getVisibleOrigin();  
  
 \_lblLives = Label::createWithSystemFont("Lives: 3","Arial",22);  
  
 \_lblScore = Label::createWithSystemFont("Score: 0","Arial",22);  
  
 \_lblLives->setAnchorPoint({0,1});  
  
 \_lblLives->setPosition(origin + Vec2(10, vs.height-10));  
  
 \_lblScore->setAnchorPoint({1,1});  
  
 \_lblScore->setPosition(origin + Vec2(vs.width-10, vs.height-10));  
  
 addChild(\_lblLives); addChild(\_lblScore);  
  
 return true;  
  
}  
  
void HUDLayer::setScore(int v){ \_score=v; if(\_lblScore) \_lblScore->setString(StringUtils::format("Score: %d",v)); }  
  
void HUDLayer::setLives(int v){ \_lives=v; if(\_lblLives) \_lblLives->setString(StringUtils::format("Lives: %d",v)); }

**File HUDLayer.h**

// HUDLayer.h  
  
#pragma once  
  
#include "cocos2d.h"  
  
class HUDLayer : public cocos2d::Layer {  
  
public:  
  
 CREATE\_FUNC(HUDLayer);  
  
 bool init() override;  
  
 void setScore(int v);  
  
 void setLives(int v);  
  
private:  
  
 cocos2d::Label\* \_lblScore=nullptr;  
  
 cocos2d::Label\* \_lblLives=nullptr;  
  
 int \_score=0, \_lives=3;  
  
};