Maze.h

#pragma once

//迷宫类

class Maze

{

public:

// 假设我们的迷宫是20\*20的

static const int MAZE\_SIZE = 20;

Maze() : layout{

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1},

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},

}

{}

int getWidth() const { return layout[0].size(); }

int getHeight() const { return layout.size(); }

//检查是否为墙壁，是1则为墙壁

bool isWall(int x, int y) const { return layout[y][x] == 1; }

private:

std::vector<std::vector<int>> layout;

};

Entity.h

#pragma once

#include <QGraphicsEllipseItem>

#include <QGraphicsView>

#include <QRandomGenerator>

#include "Maze.h"

#include <QTimer>

#include<QObject>

// 实体类，玩家和敌人都继承此类

class Entity : public QObject

{

Q\_OBJECT // 添加Q\_OBJECT宏

public:

Entity(int x, int y);

int getX()const;

int getY()const;

void setX(int x);

void setY(int y);

QGraphicsEllipseItem\* getItem();

private:

int posX, posY;

QGraphicsEllipseItem\* item;

};

// 玩家类

class PacMan : public Entity

{

public:

enum Direction { UP, DOWN, LEFT, RIGHT };

PacMan(int x, int y);

void move(Direction direction, const Maze& maze);

};

// 敌人类

class Ghost : public Entity

{

public:

Ghost(int x, int y);

public slots:

void move(const PacMan& pacMan, const Maze& maze);

private:

//QTimer\* timer;

};

Entity.cpp

#include "Entity.h"

#include <QRandomGenerator>

Entity::Entity(int x, int y) : posX(x), posY(y) {

item = new QGraphicsEllipseItem(x \* 20, y \* 20, 20, 20);

}

int Entity::getX() const {

return posX;

}

int Entity::getY()const {

return posY;

}

void Entity::setX(int x) {

posX = x;

}

void Entity::setY(int y) {

posY = y;

}

QGraphicsEllipseItem\* Entity::getItem() {

return item;

}

PacMan::PacMan(int x, int y) : Entity(x, y) {

getItem()->setBrush(Qt::yellow);

}

void PacMan::move(Direction direction, const Maze& maze) {

int dx = 0, dy = 0;

switch (direction)

{

case UP:

dy = -1;

break;

case DOWN:

dy = 1;

break;

case LEFT:

dx = -1;

break;

case RIGHT:

dx = 1;

break;

}

// 检查目标位置是否是墙壁

if (!maze.isWall(getX() + dx, getY() + dy))

{

setX(getX() + dx);

setY(getY() + dy);

getItem()->setPos(getX() \* 20, getY() \* 20);

}

}

Ghost::Ghost(int x, int y)

: Entity(x, y)

{

getItem()->setBrush(Qt::red);

}

void Ghost::move(const PacMan& pacMan, const Maze& maze)

{

// 实现敌人的移动逻辑

// 根据当前敌人和PacMan的位置、迷宫信息等进行移动操作

int currentX = getX();

int currentY = getY();

int pacManX = pacMan.getX();

int pacManY = pacMan.getY();

if (currentX < pacManX) {

setX(currentX + 1);

}

else if (currentX > pacManX) {

setX(currentX - 1);

}

if (currentY < pacManY) {

setY(currentY + 1);

}

else if (currentY > pacManY) {

setY(currentY - 1);

}

getItem()->setPos(getX() \* 20, getY() \* 20);

}

Game.h

#pragma once

#include "Entity.h"

#include "Maze.h"

#include <QGraphicsScene>

#include <QTimer>

class Game : public QObject

{

Q\_OBJECT

public:

Game();

QGraphicsScene\* getScene();

PacMan\* getPacMan();

const Maze& getMaze() const;

public slots:

void update();

void moveGhost();

private:

QGraphicsScene scene;

PacMan\* pacMan;

Ghost\* ghost;

Maze maze;

QTimer\* ghostTimer;

};

Game.cpp

#include "Game.h"

Game::Game()

{

pacMan = new PacMan(6, 6);

ghost = new Ghost(19, 19);

scene.addItem(pacMan->getItem());

scene.addItem(ghost->getItem());

ghostTimer = new QTimer(this);

connect(ghostTimer, &QTimer::timeout, this, &Game::moveGhost);

ghostTimer->start(1000); // 设置定时器间隔时间

for (int y = 0; y < maze.getHeight(); ++y)

{

for (int x = 0; x < maze.getWidth(); ++x)

{

if (maze.isWall(x, y))

{

int adjustedX = (x + 6) \* 20;

int adjustedY = (y + 6) \* 20;

auto item = new QGraphicsRectItem(adjustedX, adjustedY, 20, 20);

item->setBrush(Qt::blue);

scene.addItem(item);

}

}

}

}

QGraphicsScene\* Game::getScene()

{

return &scene;

}

PacMan\* Game::getPacMan()

{

return pacMan;

}

const Maze& Game::getMaze() const

{

return maze;

}

void Game::update()

{

// 更新游戏状态

}

void Game::moveGhost()

{

const PacMan\* pacMan = getPacMan();

const Maze& maze = getMaze();

ghost->move(\*pacMan, maze);

}

GameWindow.h

#pragma once

#include"Game.h"

#include <QKeyEvent>

#include <QtWidgets/QMainWindow>

#include "ui\_GameWindow.h"

class GameWindow : public QMainWindow

{

Q\_OBJECT

public:

GameWindow(QWidget\* parent = nullptr);

~GameWindow();

protected:

void keyPressEvent(QKeyEvent\* event)

{

const Maze& maze = game->getMaze(); // 获取迷宫对象

switch (event->key())

{

case Qt::Key\_W:

game->getPacMan()->move(PacMan::UP, maze);

break;

case Qt::Key\_S:

game->getPacMan()->move(PacMan::DOWN, maze);

break;

case Qt::Key\_A:

game->getPacMan()->move(PacMan::LEFT, maze);

break;

case Qt::Key\_D:

game->getPacMan()->move(PacMan::RIGHT, maze);

break;

}

}

private:

Ui::GameWindowClass ui;

Game\* game;

QGraphicsView\* view;

};

GameWindow.cpp

#include "GameWindow.h"

#include <QTimer>

GameWindow::GameWindow(QWidget\* parent)

: QMainWindow(parent)

{

ui.setupUi(this);

// 创建游戏对象

game = new Game();

// 创建 QGraphicsView 对象，并将 Game 的 QGraphicsScene 对象设置为其场景

view = new QGraphicsView(game->getScene(), this);

// 设置窗口为 QGraphicsView 对象

setCentralWidget(view);

// 设置窗口大小

setFixedSize(800, 600);

// 设置窗口标题

setWindowTitle("My Game");

// 允许窗口接收键盘事件

setFocusPolicy(Qt::StrongFocus);

}

GameWindow::~GameWindow()

{}