Greedy Snake

May 2024

Content

1 Overview

项目贪吃蛇小游戏是一款基于 C++和 Qt 框架开发的桌面应用程序。游戏界面主要分为游戏显示区域和操作控制区域，前者为玩家提供了实时的游戏动态显示，后者包含了游戏控制的各种功能按钮，如开始游戏、暂停游戏、继续游戏、重新开始、游戏存档的载入与保存，以及退出游戏的操作。

在游戏机制设计上，本游戏采用键盘的方向键（或 WASD 键）作为输入设备，控制贪吃蛇的移动方向。玩家的主要任务是操作蛇在一个封闭的空间内移动，通过吞食出现的食物来增加蛇的长度，同时避免蛇头触碰到自身或边界。游戏的挑战性主要体现在对玩家反应速度和策略规划的考验。游戏开始前，玩家可手动点击鼠标界面设置个性化的障碍物，增加挑战趣味性。此外，游戏还设有速度调节功能，允许玩家根据个人偏好调整蛇的移动速度，增加游戏的可玩性和挑战性。该功能通过一个界面滑块来实现，用户可以实时调整速度，以应对游戏难度的变化。

本项目重现了贪吃蛇游戏的基本功能，还通过软件开发技术增强了用户交互和视觉体验，适合不同年龄层的玩家进行娱乐和休闲。通过不断优化和添加新的功能，可以进一步提升游戏体验和趣味性。

2 Language and IDE

2.1 C++

C++是一种强大的通用编程语言，由贝尔实验室的 Bjarne Stroustrup 于 20 世纪 80 年代开发。它建立在 C 语言的基础上，扩展了面向对象编程的概念，为程序员提供了更丰富的抽象和结构工具。C++的设计目标是平衡高性能和灵活性，适用于系统级编程和高性能应用开发，包括操作系统、游戏引擎、图形处理、科学计算和数据库系统。

C++具有多个重要的特性。首先，面向对象编程支持封装、继承和多态性，使得代码更具模块化和可维护性。模板编程为泛型编程提供了强大的功能，允许开发者创建通用的算法和数据结构，同时保持类型安全性。标准模板库（STL）作为 C++的一部分，提供了一套高效、通用的数据结构和算法，使得程序开发更加便捷。此外，C++还具有直接的硬件访问能力和内存管理功能，这在一些关键领域，例如嵌入式系统、游戏开发等尤为重要。C++的标准库涵盖广泛的功能，包括文件 I/O、多线程、网络编程和数学计算等，使其成为一个全面的编程工具。

C++在不断进化，以满足现代软件开发的需求。C++11、C++14、C++17 和 C++20 等新标准引入了许多新的语言特性，如 lambda 表达式、智能指针、并发库和模块化等，提高了开发效率和代码的可维护性。总之，C++以其高性能、灵活性和多样性，依然是现代软件开发的重要工具之一。

2.2 QT

2.3.1 Introduction of QT

Qt（Qt Application Framework）是一款由 Qt 公司（原 Trolltech 公司）开发的跨平台应用程序开发框架，提供全面的工具和库，以简化桌面、嵌入式和移动

应用程序的开发。Qt 框架最初诞生于 1990 年代，经过多年的发展与完善，已经成为全球广泛使用的 UI 设计工具和应用开发平台。

Qt 的核心优势在于它的跨平台特性。通过一次性编写代码，开发者可以轻松地将应用程序部署到 Windows、Linux、macOS 等不同类型设备上，无需针对不同系统进行单独调整。这大大降低了开发和维护成本，提高了效率。

Qt 还提供强大的图形用户界面（GUI）功能。它的 Qt Widgets 模块允许开发者创建精美的传统桌面应用界面，而 Qt Quick 模块通过 QML 和 JavaScript 提供了一种更现代的界面开发方式，适用于移动应用或需要快速响应的界面。

除 GUI 之外，Qt 还提供丰富的工具和模块，用于多媒体处理、网络通信、数据库访问、文件操作、多线程和国际化等。其综合的开发环境 Qt Creator 为开发者提供了全面的代码编辑、调试和可视化设计工具，进一步简化了开发过程。作为一个高度可扩展的框架，Qt 的模块化设计允许开发者根据项目需求进行裁剪和扩展。

2.3.2 Why choose QT

Qt 是一个跨平台的应用程序开发框架，相对于传统的 C++，它提供了丰富的 GUI 组件和功能，使得开发者能够轻松创建现代化、美观且响应式的用户界面。Qt 引入了信号与槽机制，简化了对象之间的通信。信号是类的成员函数，用于在特定事件发生时触发，但它本身并不包含任何处理逻辑。信号仅用于通知系统事件已经发生，比如按钮被点击或定时器超时等。槽是普通的成员函数，当与之相连的信号被触发时，槽函数会被调用。槽可以关联任何信号，无论信号源自本地还是远程对象。信号和槽的连接通过 QObject::connect() 函数实现，确保当信号被触发时，适当的槽函数得到执行。连接可以在信号与普通函数之间、甚至在不同的对象之间建立。

此外，Qt 还提供了跨线程通信的支持，方便开发者进行并发编程，处理多线程问题更加简单。Qt 的丰富库支持涵盖了图形、网络、数据库、多媒体等方面的功能，使得开发者能够更轻松地实现游戏所需的各种功能，例如游戏界面的绘制和音频的播放。Qt 提供了 Qt Creator 等集成开发环境，配备了丰富的调试工具和设计界面，使得开发者能够高效地进行项目开发。此外，Qt 还拥有完善的官方文档和教程，为开发者提供了学习和参考的资源。

3 Program functionalities

3.1 Game Design

游戏主界面被分为游戏显示区域和操作控制区域两部分。游戏显示区域为贪吃蛇通过移动躲避障碍物吃到目标食物；操作控制区域为游戏功能按钮和游戏进度情况。

3.1.1 界面 UI 设计

游戏地图尺寸为 820x820 像素，通过 GameWidget 类设置，其界面大小为游戏界面的主要显示区域。这个显示区域不仅承载了游戏的整体画面，还要能够准确地展示游戏地图上的各种元素。游戏地图的二维数组表示为 map_label[MAX_X][MAX_Y]，每个元素指向一个 QLabel 控件，这些控件以 20x20 像素的大小在界面上呈现，背景为灰色，形成了游戏区域的空白。

在这个基础上，蛇身、食物和边界等元素被动态地展示在地图上。蛇身由灰色的 QLabel 控件表示，每节独立展示，通过数组中的指针进行访问和控制。食物以红色的 QLabel 控件随机生成在地图的空白格子上，为游戏增添了挑战和乐趣。而地图的边界则由灰色的 QLabel 控件围绕在地图的四周，不仅起到了界面装饰的作用，还限制了蛇的移动范围，为游戏增加了策略性。值得注意的是，使用圆形作为蛇头，从而蛇头与蛇神转向时更为顺滑。

为了有效管理游戏元素，采用了二维数组 map_label[MAX_X][MAX_Y]来表达游戏地图，这种结构设计简洁高效，使得地图元素能够方便地被访问和操作。每个数组元素指向一个 Snake 结构体，该结构体包含一个 QLabel 指针和表示蛇身位置及类型的成员变量。这样的设计使得游戏元素在地图上的展示和更新变得十分便捷，同时也提高了游戏的性能和可维护性。整个游戏界面的构建与元素的动态展示得以实现，为玩家提供了一个流畅、直观的游戏体验。

3.1.2 障碍物设计

在贪吃蛇游戏中，障碍物的设计被整合到游戏地图的构建中。障碍物可通过随机方式或由玩家自行定义的方式进行放置，成为游戏中固定的障碍。它们的存在旨在提高游戏的难度和策略深度。

障碍物的默认行为是围绕游戏地图的边缘创建一圈，以限制蛇的移动范围并防止其超出游戏区域。这种默认边界在 initBorder() 函数中实现，它设置边缘格子的 type 为 border_label 并将其 QLabel 样式更改为显著的颜色（通常是浅灰色），以使其与游戏背景区分开。

除了这些固定边界，玩家还可以通过鼠标点击在游戏开始前动态添加障碍物。这种动态添加在 mousePressEvent() 函数中处理，允许玩家通过点击游戏区内边缘之外的任何位置放置或移除障碍物。鼠标点击位置被计算为相应的格子坐标，然后检查该位置是否已有障碍物：

- 如果已经是障碍物，则将其还原为普通背景格子，将 type 设为 bg_label 并更改样式为背景色。
- 如果不是障碍物，则将格子设置为新的障碍物，将 type 设为 border_label 并更新样式为障碍物特有的样式。

在游戏运行时，障碍物对玩法有着显著影响。在 moveSnake() 函数中，蛇的每次移动都会检查其头部即将进入的新位置是否为障碍物。如果是障碍物，游戏将调用 gameOver() 函数，表示玩家撞到了障碍物，从而导致游戏结束。

3.1.3 蛇的本体移动

在游戏中，蛇的移动是通过一个核心的定时器（QTimer）来控制的。定时器触发 timeout() 信号，该信号连接到 snakeMoveSlots() 槽函数，这个槽函数再调用 moveSnake() 函数以实施具体的移动逻辑。在 moveSnake() 函数中，首先更新游戏步数并通过信号 displayStepSignal(steps) 通知界面更新步数显示。接下

来，根据蛇当前的移动方向（由 dX 和 dY 表示），计算蛇头的新位置，并检查这个新位置是否有效：如果新位置是边界或蛇的其它部分，游戏结束；如果是食物，则蛇身增长一节，分数增加，并触发 displayScoreSignal(scores) 信号以更新分数显示，同时在地图上随机生成新的食物。如果新位置不是食物，蛇头简单地移向新位置，而蛇尾的位置更新为背景，形成蛇的移动效果。

此外，蛇的移动方向由玩家通过键盘控制。根据用户按下的方向键或 WASD 键更新蛇的移动方向，同时确保蛇不会直接反向移动。

3.1.4 食物生成

在贪吃蛇游戏中，食物的生成是一个至关重要的功能，直接影响到游戏的可玩性和挑战性。createFood()函数负责封装这一逻辑，并利用随机数生成技术决定食物在地图上的位置，确保食物不会出现在蛇身或其他障碍物上。首先，使用 srand((unsigned)time(0))初始化随机数生成器，确保每次游戏开始时食物的位置都是随机的。接着，利用 rand() % MAX_X 和 rand() % MAX_Y 生成随机的 X 和 Y 坐标。这里的 MAX_X 和 MAX_Y 分别表示地图的宽度和高度，代表地图上可用的最大行和列数。

为确保食物不会生成在蛇身或障碍物上，createFood()会通过一个循环检查随机生成的位置是否为空闲（背景标签）。这个验证过程利用 while 循环持续生成新的位置，直到找到一个有效的空位置为止。一旦找到合适的地方，函数将相应位置的 Snake 结构体的 type 属性设置为 food_label，表示该位置为食物。关联的 QLabel 控件样式会被设为红色背景，确保在游戏界面中与蛇身和背景区分开来。最后，通过调用 QLabel 的 show()方法，将食物在界面上呈现给玩家。

3.1.5 背景音乐

在贪吃蛇游戏中，背景音乐的处理通过 Qt 音频播放接口实现，增添了游戏的趣味性，让玩家沉浸在更具吸引力的游戏环境中。

通过实例化一个 QMediaPlaylist 对象管理音频文件的播放，该播放列表可以包含一个或多个音频文件，以支持顺序播放或循环播放。将背景音乐文件（通常是.wav 格式）添加到播放列表，通过 Qt 资源系统将文件嵌入程序内部。最后，配置播放列表为循环播放模式，确保音乐在结束后自动重新开始，持续为整个游戏过程提供背景音乐。

3.2 Game Control

3.2.1 控制按钮

在贪吃蛇游戏中，玩家通过一组丰富的控制按钮实现游戏操作，这些按钮包括开始、暂停、继续、重新开始、保存、加载、退出和帮助。每个按钮都绑定了特定的槽函数，确保了玩家操作的即时响应和适当的游戏流程控制。例如，开始按钮启动游戏，暂停按钮则使游戏暂停等待后续操作。此外，按钮的可用状态会根据游戏的当前阶段进行调整，以防止不合适的操作，同时通过样式变化增强视觉体验。

值得注意的是游戏的保存和加载功能，通过相应按钮与槽函数的联动，确保玩家操作可以即时触发动作，实现游戏状态的保存与恢复，为玩家提供了灵活中断和继续游戏的便利性。

- 游戏保存功能

当玩家点击保存按钮时，saveGame() 函数被调用。首先，通过 timer.stop() 暂停游戏，确保在保存过程中游戏状态不会变化。随后，函数遍历蛇的每个部分，记录每节蛇身的坐标，并保存至 snakeInfo 中；同时，遍历整个游戏地图，将所有边界和食物的位置记录到 borderInfo 和 foodInfo。这些信息与当前的步数和分数一起合成一个完整字符串，并通过文件对话框选择文件路径，写入一个文本文件，保存当前游戏状态。保存完成后，用户会看到一个弹窗提示保存成功。

■ 游戏加载功能

在玩家点击加载按钮时，loadGame() 函数被调用。文件对话框打开后，玩家可以选择之前保存的游戏状态文件。函数读取并解析保存的数据，恢复蛇、边界和食物的坐标，以及步数和分数。地图中每个格子的显示状态根据加载的数据进行更新，蛇的移动方向也根据保存时的最后状态设定。加载成功后，用户会看到一个弹窗提示加载成功。

3.2.2 显示窗口

游戏中的分数和步数通过两个独立的 QLCDNumber 控件实时显示。这些数字会在游戏进行中根据玩家的表现实时更新，提供反馈如分数增加和蛇的移动步数。这种实时更新功能通过连接游戏逻辑发出的信号与显示分数和步数的槽函数实现，确保了游戏信息的准确传递。

3.2.3 速度调节

速度调节是通过一个 QSpinBox 控件实现，允许玩家在游戏开始前或重新开始时设置蛇的移动速度。速度的调整是通过修改游戏循环的时间间隔来实现的，这直接影响蛇的响应速度和游戏的难度级别。速度调节器的数值改变会通过连接到相应槽函数的信号进行处理，从而在游戏中实现动态的速度调整。这种设计不仅提升了游戏的可玩性，还增加了策略性，玩家需要根据自己的反应速度和游戏进程来选择合适的速度。

4 Demonstrations

4.1 Main Interface

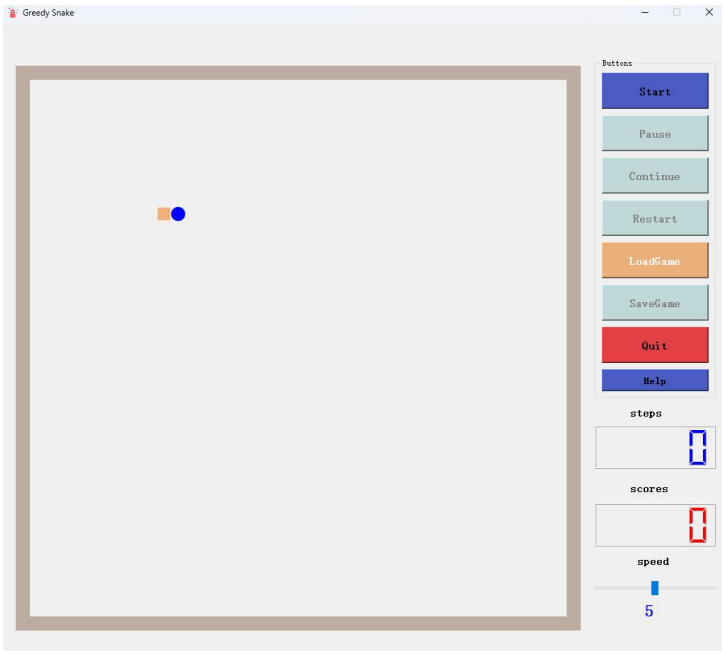

Figure 4.1 Game main interface

4.2 Game Execution

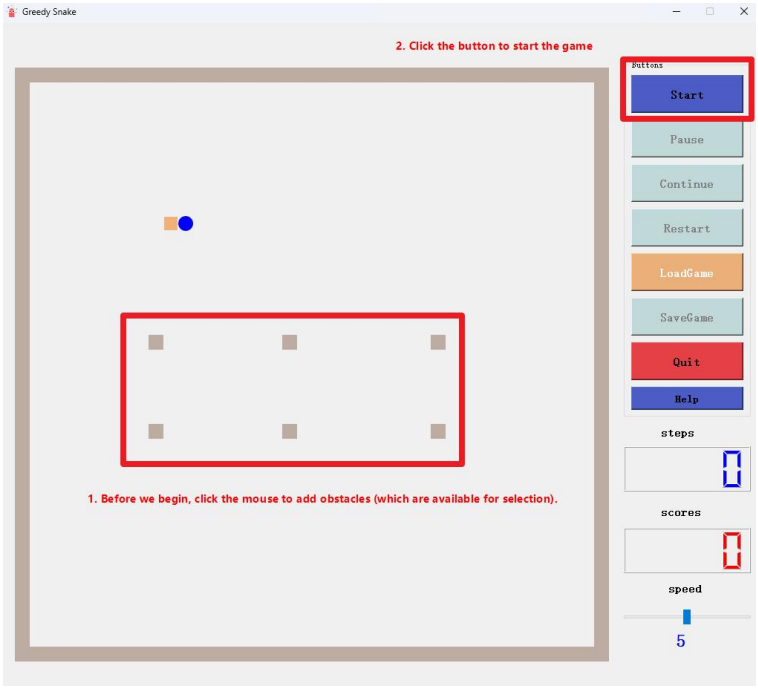

Figure 4.2 Start game and manually adding obstacles

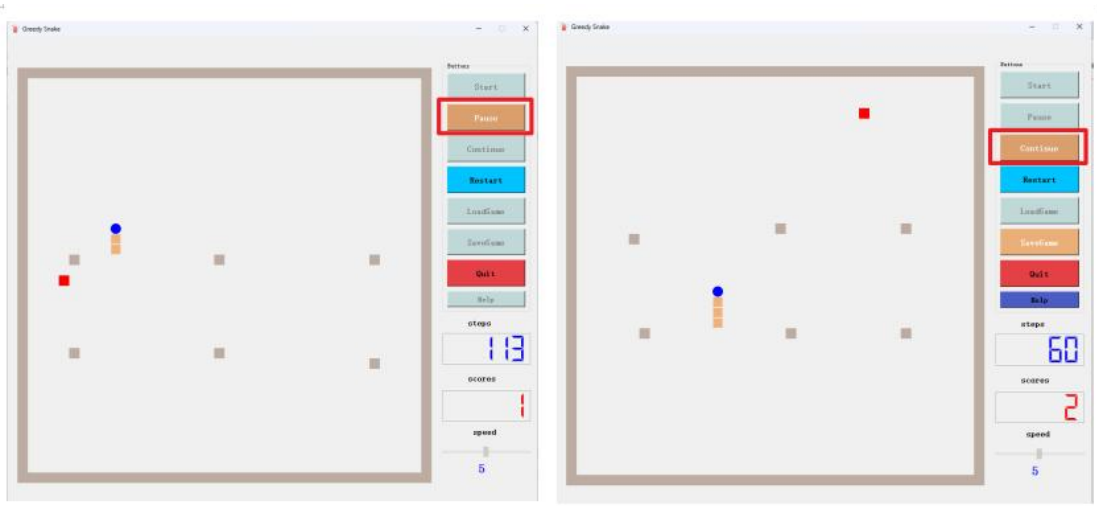

Figure 4.3 Pause and continue games

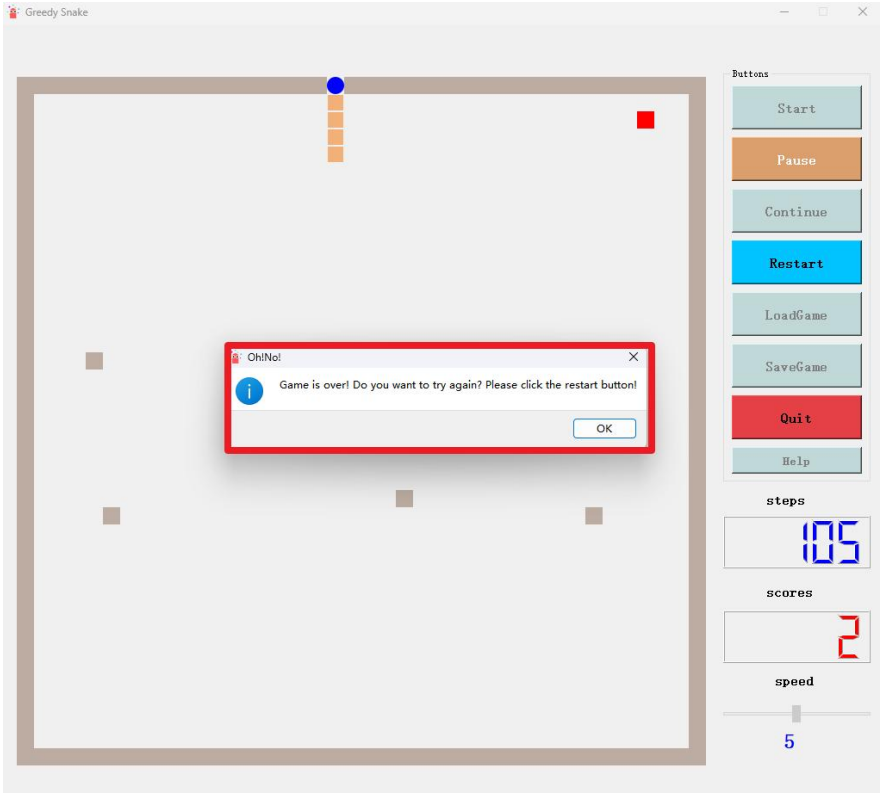

Figure 4.4 Game failed

4.3 Game Restart

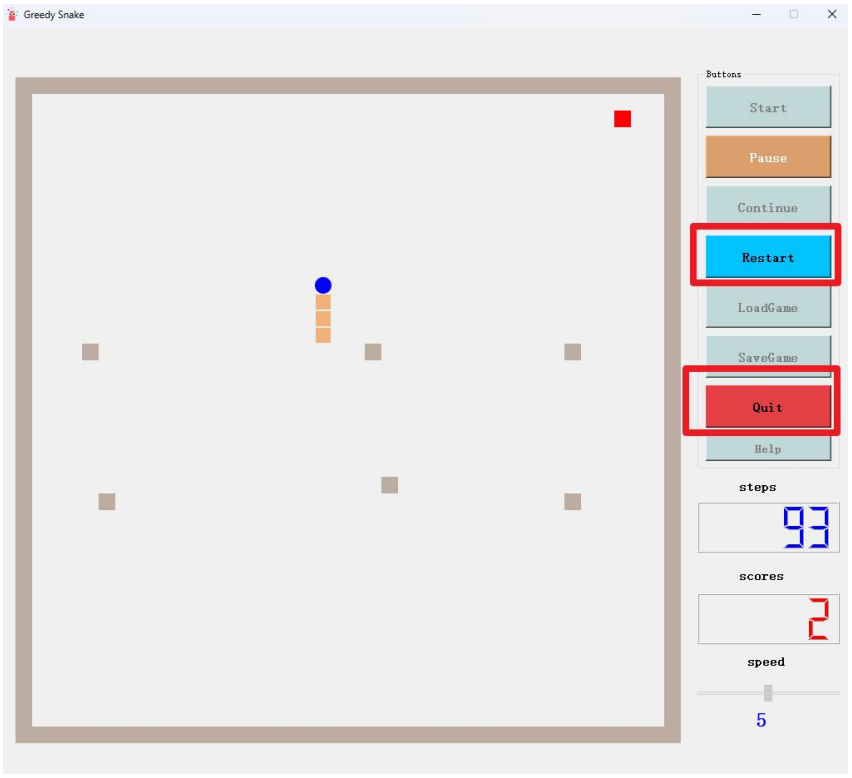

Figure 4.5 Restart or exit the game

4.4 Game Save and Load

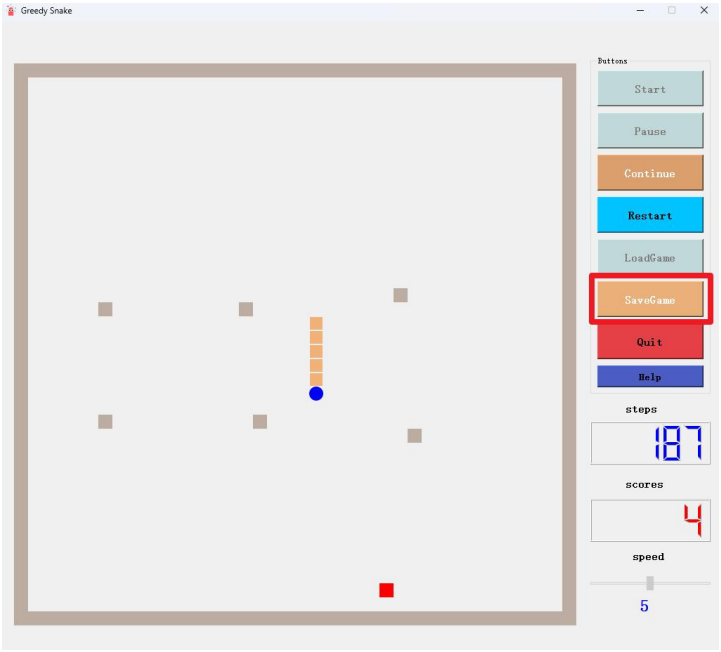

Figure 4.6 Click the save game button

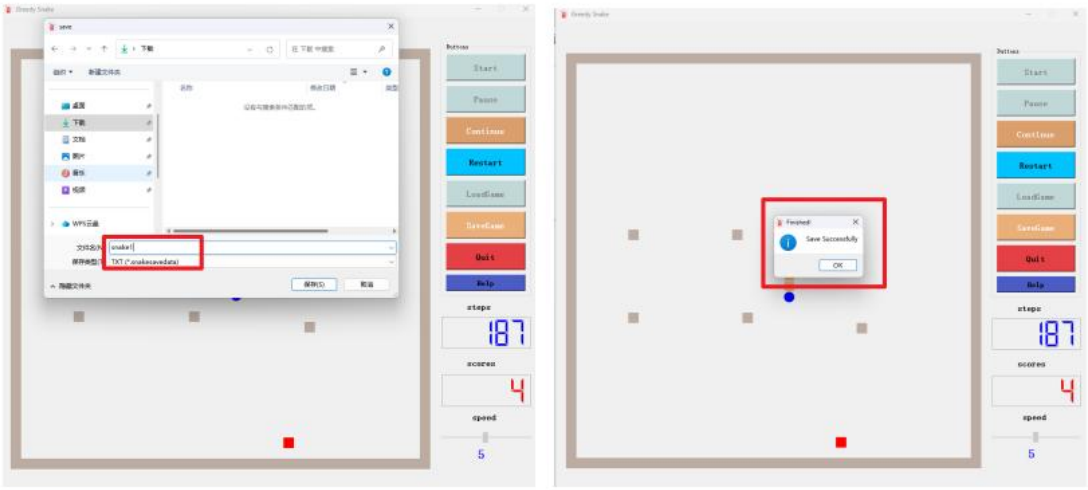

Figure 4.7 Save current game progress

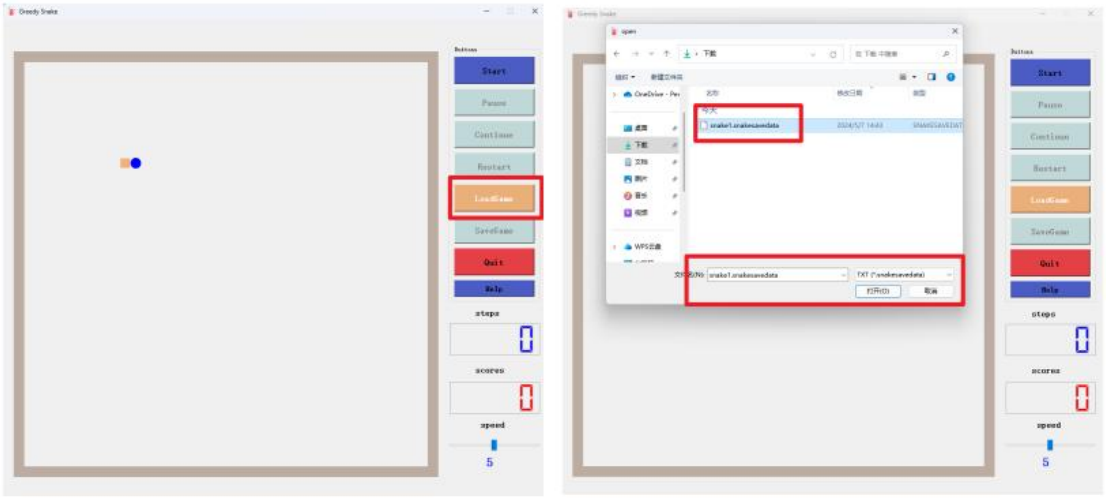

Figure 4.8 Read historical game progress

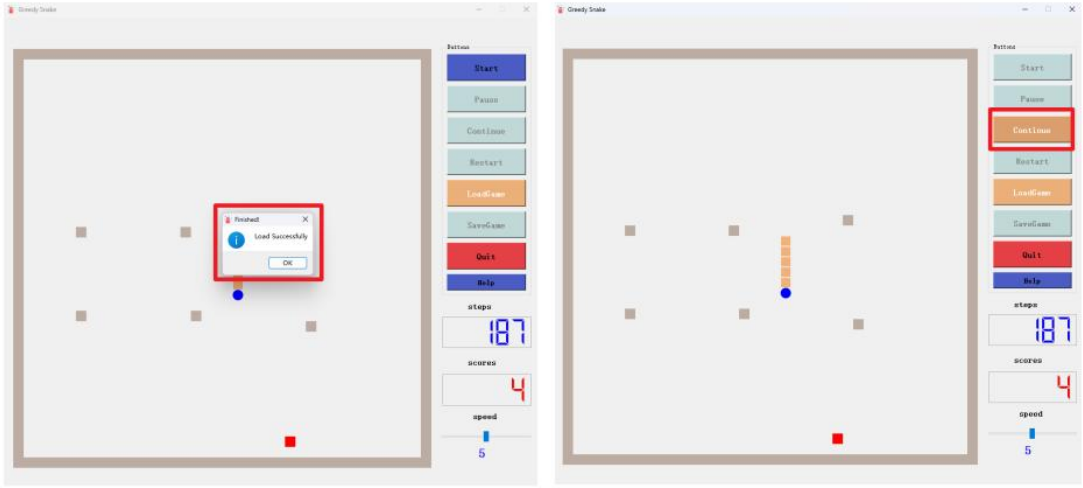

Figure 4.9 Continuing the game

5 Originality and external Tools

5.1 Originality

项目在现有开源贪吃蛇游戏的数据结构设计（如游戏地图设计、贪吃蛇结构设计等）和游戏运行逻辑（游戏初始化逻辑、随机食物生成逻辑等）的基础上，尝试使用 QT 代替控制台终端类型开发本游戏，并使用不同的色彩设计良好的游戏界面。在功能方面，游戏可以在开始前通过放置或移除障碍物与游戏环境进行交互；游戏包含一个速度调节滑块，实时影响蛇的移动速度，游戏设计完整的控制按钮，包括开始、暂停、继续、重新开始、保存、加载和退出等按钮；游戏配备了背景音乐，增强了游戏氛围；游戏添加了保存和加载功能，可以在任何时候保存游戏并稍后加载，使他们可以在不丢失进度的情况下恢复游戏；游戏还包含一个通过专用按钮访问的帮助部分，为玩家提供游戏规则和说明。

5.2 External Tools

开发组件中主要使用 Qt 内置库，提供了一套丰富的功能来构建图形界面和处理多媒体元素，极大地简化了复杂用户界面的开发过程。此外，项目采用 Git 作为版本控制系统，它支持分布式操作，便于多人协作和版本历史的管理。项目代码托管在 GitHub/Gitee 上，这不仅便于代码的版本追踪和协作开发，还提供了代码审查和问题跟踪的工具，增强了项目的可见性和开放性。

6 Program highlights

本次项目开发的收获如下：

① 增强项目架构规划能力：针对项目整体的运行逻辑和每个细节的设计，如游戏逻辑、UI 界面和资源管理，使得项目更具灵活性和可维护性。这次实践不仅对系统的架构设计有了更深刻的认识，也培养了在开发流程中全局思维与精细掌控的能力。

② 加强代码 bug 解决的能力：通过调试和查找 bug 源头，学会系统地分析错误，快速定位并修复代码中的缺陷。面对代码，培养了编写更清晰、可维护性高的代码习惯，并学会了利用调试工具，精准追踪 bug 的发生过程。

③ 增强沟通协作的能力：软件开发是一个团队协作的过程，良好的沟通至关重要。在本项目中，使用了 Git 和 Gitee 作为协作平台，这不仅帮助管理代码的版本，更重要的是提供了一个沟通和协作的平台。团队成员能够有效地交流思想、审阅代码并提出改进建议，这极大地提升了项目的整体质量和团队成员之间的合作效率。

7 References

[1] https://blog.csdn.net/w1u1kan/article/details/124073931

[2] https://juejin.cn/post/7022846211786801159

[3] https://www.kancloud.cn/kancloud/qt-study-road-2/99470

[4] https://www.jb51.net/article/212485.htm

[5] https://github.com/Maserhe/MyeatSnack

[6] https://github.com/Wytrix/Gluttonous-Snake-Game

[7] https://github.com/muzhi621/Snakes

[8] https://github.com/liplay/QT-Snake

Appendix 1: Game Installation Guide

1. QT 安装

参考安装链接： https://blog.csdn.net/qq_42257666/article/details/123285125

版本：qt-opensource-windows-x86-5.14.2

2. 项目运行与编译

双击项目中 snake.pro 或者 QT 主界面打开现有项目，在 debug 或者 release 下运行即可成功编译。

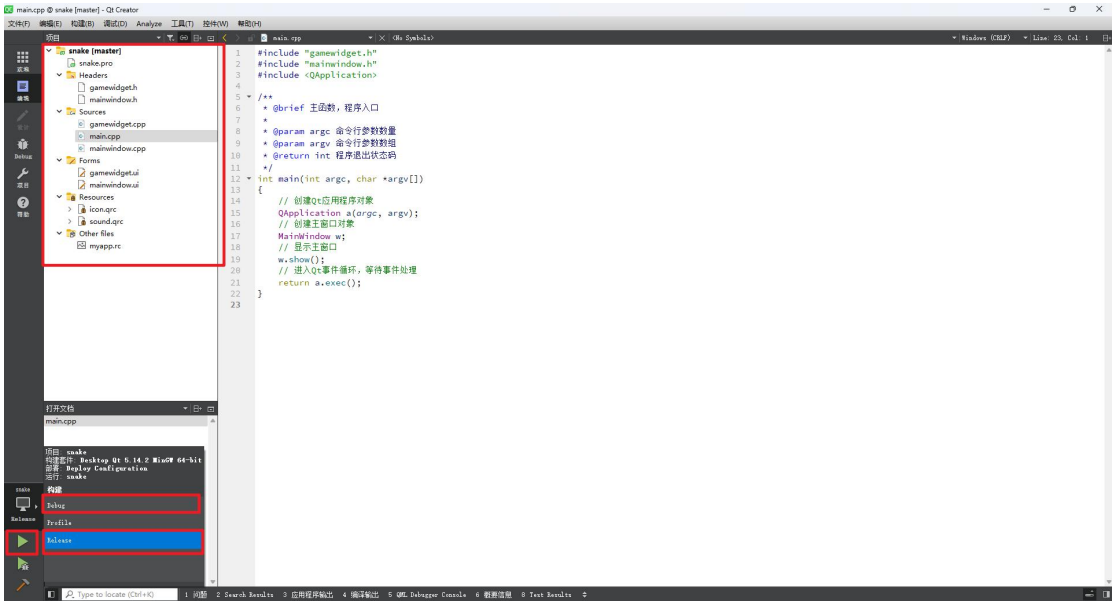

Figure 8.1 QT main interface

Appendix 2: Brief Analysis of Code

1. 代码结构

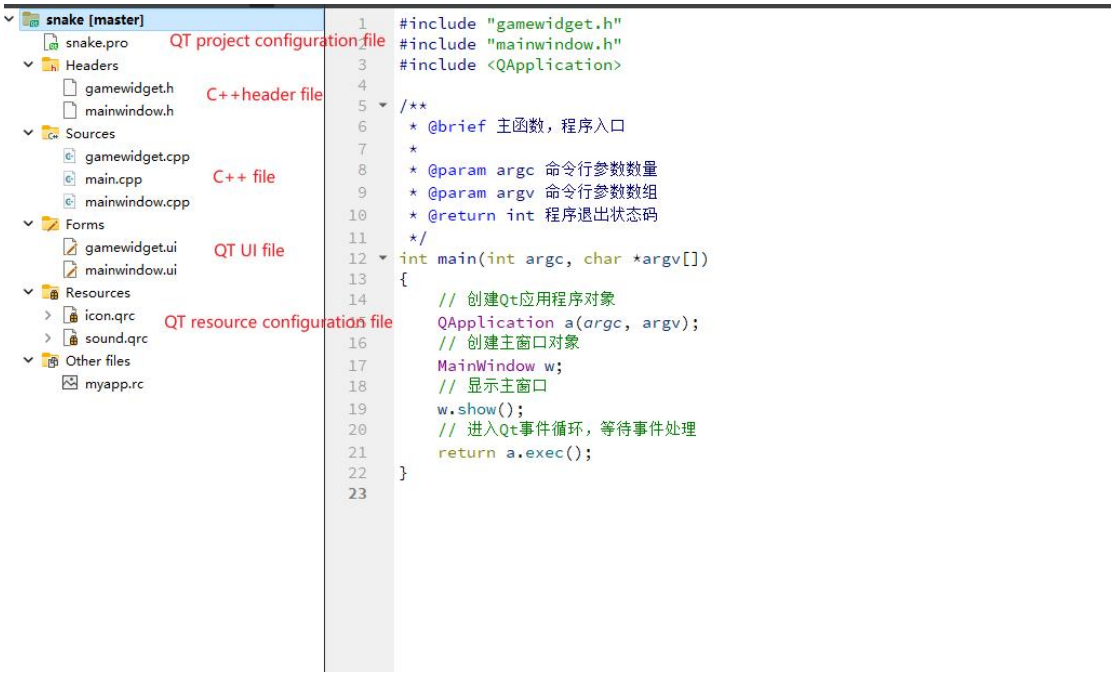

Figure 8.2 Project Structure

2. 函数简要分析

① **main.cpp:** 文件为程序的入口，设置了应用程序环境，启动了主窗口，并让应用程序进入事件循环，从而实现基本的用户界面操作。

```
#include "gamewidget.h"
#include "mainwindow.h"
#include <QApplication>

/**
 * @brief 主函数，程序入口
 *
 * @param argc 命令行参数数量
 * @param argv 命令行参数数组
 * @return int 程序退出状态码
 */
int main(int argc, char *argv[])
{
```

```
    // 创建 Qt 应用程序对象
    QApplication a(argc, argv);
    // 创建主窗口对象
    MainWindow w;
    // 显示主窗口
    w.show();
    // 进入 Qt 事件循环，等待事件处理
    return a.exec();
}
```

② **mainwindow.h**：提供了主窗口的接口定义和信号槽机制，是游戏主界面功能的框架。

```
namespace Ui
{
    class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    // 构造函数
    // @param parent 父窗口指针，默认为 nullptr
    explicit MainWindow(QWidget *parent = nullptr);

    // 析构函数
    ~MainWindow();
signals:

private slots:
    // 开始游戏触发的槽函数
    void on__start_triggered();
    // 暂停游戏触发的槽函数
    void on__pause_triggered();
```

```
23.     // 保存游戏触发的槽函数
24.     void on__save_triggered();
25.     // 继续游戏触发的槽函数
26.     void on__continue_triggered();
27.     // 重新开始游戏触发的槽函数
28.     void on__restart_triggered();
29.     // 加载游戏触发的槽函数
30.     void on__load_triggered();
31.     // 退出游戏触发的槽函数
32.     void on__quit_triggered();
33.     // 游戏结束槽函数
34.     void gameOverSlots();
35.     // 点击开始按钮触发的槽函数
36.     void on_start_clicked();
37.     // 点击加载按钮触发的槽函数
38.     void on_load_clicked();
39.     // 点击暂停按钮触发的槽函数
40.     void on_pause_clicked();
41.     // 点击继续按钮触发的槽函数
42.     void on_con_clicked();
43.     // 点击重新开始按钮触发的槽函数
44.     void on_restart_clicked();
45.     // 点击保存按钮触发的槽函数
46.     void on_save_clicked();
47.     // 点击退出按钮触发的槽函数
48.     void on_quit_clicked();
49.     // 显示步数的槽函数
50.     // @param step 步数值
51.     void displayStepSlots(int step);
52.     // 显示分数的槽函数
53.     // @param score 分数值
54.     void displayScoreSlots(int score);
55.     void showHelpDialog();      // 显示游戏规则
56.
57. private:
58.     Ui::MainWindow *ui;
```

```
59.     QMediaPlayer *backgroundMusicPlayer;     // 背景音乐播放器指针
60.     QMediaPlaylist *backgroundMusicPlaylist; // 背景音乐播放列表指针
61. };
```

③ **gamewidget.h**：管理游戏的核心逻辑，包括蛇的移动、地图管理、键盘事件处理、游戏开始、暂停、保存和加载等功能。

```
1.  // 定义枚举类型 Labeltype，表示标签的类型
2.  enum Labeltype
3.  {
4.      bg_label,     // 背景标签
5.      snake_label,  // 蛇标签
6.      food_label,   // 食物标签
7.      border_label, // 边界标签
8.  };
9.
10. // 定义结构体 Snake，表示蛇的信息
11. struct Snake
12. {
13.     QLabel *label; // QLabel 指针，表示蛇的标签
14.     int type;      // 蛇的类型，使用 Labeltype 枚举类型表示
15.     int x;         // 蛇的 x 坐标
16.     int y;         // 蛇的 y 坐标
17. };
18.
19. namespace Ui
20. {
21.     class GameWidget;
22. }
23.
24. // GameWidget 类，继承自 QWidget
25. class GameWidget : public QWidget
26. {
27.     Q_OBJECT
28. protected:
```

```
29.     void keyPressEvent(QKeyEvent *e);          // 键盘按下事件处理函数
30.     void mousePressEvent(QMouseEvent *event); // 鼠标点击事件处理函数
31. public:
32.     explicit GameWidget(QWidget *parent = nullptr); // 构造函数
33.     ~GameWidget();                                  // 析构函数
34.     void initGame();                                // 初始化游戏
35.     void initBorder();                              // 初始化边界
36.     void initSnake();                               // 初始化蛇
37.     void moveSnake();                               // 移动蛇
38.     void createFood();                              // 创建食物
39.     void startGame(double);                         // 开始游戏
40.     void pauseGame();                               // 暂停游戏
41.     void continueGame(double);                      // 继续游戏
42.     void enableCreate();                            // 允许创建
43.     void saveGame();                                // 保存游戏
44.     void loadGame();                                // 加载游戏
45.     void restartGame();                             // 重新开始游戏
46.     void quitGame();                                // 退出游戏
47.     void gameOver();                                // 游戏结束
48. signals:
49.     void gameOverSignal();         // 游戏结束信号
50.     void displayStepSignal(int);  // 显示步数信号
51.     void displayScoreSignal(int); // 显示分数信号
52. private:
53.     int steps = 0;                  // 步数
54.     int scores = 0;                 // 分数
55.     double speed;                   // 速度
56.     Ui::GameWidget *ui;             // UI 指针
57.     Snake *map_label[MAX_X][MAX_Y]; // 地图标签数组
58.     QTimer timer;                   // 计时器
59.     QList<Snake *> snake;           // 蛇
60.     Snake *head;                    // 蛇头
61.     Snake *tail;                    // 蛇尾
62.     int dX, dY;                     // 移动方向
63.     bool canCreat;                  // 是否可以创建
64.     bool clicked[MAX_X][MAX_Y];     // 点击状态
```

```
65.     bool pressed;                   // 按下状态
66.     bool canMove;                   // 是否可以移动
67. public slots:
68.     void snakeMoveSlots(); // 蛇移动槽函数
69. };
70.
```