# 高级语言程序设计

# 实验报告

**南开大学 计算机科学与技术系**

**姓名：[姚浩伟]**

**学号：[2210914]**

**班级：[0928]**

**日期：2024年5月**

**高级语言程序设计**

**实验报告**

---

## 目录



---

## 一、 作业题目

设计并实现一个坦克对战游戏。

## 二.、开发软件

- **Qt Creator 5.14.2**
- **C++**

## 三、课题要求

- **语言：** 使用C++。
- **平台：** 使用Qt作为开发平台。
- **主题：** 坦克对战。
- **形式：** 图形化应用程序。

# 四、 主要流程

## 1. 游戏初始化

游戏初始化是设置游戏基础环境的关键阶段。这个阶段涉及以下几个步骤：

- **地图加载：** 从文件中读取地图数据，包括障碍物位置、地图尺寸等。这些数据决定了游戏窗口的大小以及游戏中的交互环境。

```
void Game::prepare_map(Map* map, string map_file_address, int& win_width,
int& win_height, Point& first_tank_pos, Point& second_tank_pos) {
    // 读取地图文件，设置地图尺寸和坦克初始位置
}
```

- **窗口设置：** 根据地图尺寸初始化游戏窗口，设定标题和窗口大小。窗口是玩家与游戏交互的界面。

```
game_window = new Window(win_width * GRID_SIZE + 1, win_height * GRID_SIZE +
1, "Tank Battle");
```

- **坦克初始化：** 设置两辆坦克的初始位置和朝向。坦克是玩家操作的主要对象，它们的属性包括位置、方向、速度和健康状态。

```
first_tank = Tank(1, config.NORMAL, first_tank_pos, rand() % 360, map,
bullet_rep, game_window);
second_tank = Tank(2, config.NORMAL, second_tank_pos, rand() % 360, map,
bullet_rep, game_window);
```

- **子弹仓库创建：** 初始化子弹仓库，管理所有坦克发射的子弹。子弹仓库负责追踪每个子弹的状态和位置。

```
cpp
复制代码
bullet_rep = new Bullets_Repository(2, map, game_window);
```

## 2. 游戏主循环

游戏主循环是持续进行的，直到游戏结束。它负责处理游戏逻辑和更新游戏状态。

- **事件处理：** 游戏窗口会不断地捕捉和处理玩家的输入，如键盘和鼠标事件，这些输入决定坦克的移动和射击。

```
cpp复制代码void Game::handle_game() {
    Event event;
    while (game_window->poll_event(event)) {
        process_event(event);
    }
}
```

- **游戏逻辑更新：** 更新所有游戏对象的状态，包括坦克的位置、子弹的飞行和碰撞检测。

```
cpp复制代码map->update(); // 更新地图上的状态，如障碍物与子弹的交互
bullet_rep->update(); // 更新子弹位置和检查碰撞
first_tank.update(); // 更新坦克状态，处理移动和射击逻辑
second_tank.update();
```

- **渲染更新：** 清除屏幕，绘制所有游戏对象，然后更新显示，以反映最新的游戏状态。

```
cpp复制代码game_window->clear();
map->draw(game_window);
bullet_rep->draw(game_window);
first_tank.draw(game_window);
second_tank.draw(game_window);
game_window->update_screen();
```

- **游戏结束检测：** 检查是否有坦克的健康值降到零以下，或者玩家选择退出游戏。

```
cpp复制代码if (first_tank.is_destroyed() || second_tank.is_destroyed()) {
    display_end_game_message();
    break;
}
```

以上流程共同构成了游戏的运行机制，每个部分都是为了确保游戏可以平滑、连贯地运行，同时提供动态交互和良好的用户体验。五. 程序测试

# 五、SDL库的使用

在坦克对战游戏的开发中，Simple DirectMedia Layer（SDL）库起到了核心的作用，特别是在游戏图形渲染、事件处理以及音频管理方面。SDL是一个跨平台的开发库，能够简化对操作系统层的调用，是游戏开发中常用的工具之一。以下是SDL在本游戏项目中具体的应用：

## 图形渲染

SDL提供了基本的2D图形渲染功能，这在游戏开发中非常关键。通过SDL，游戏能够创建窗口、加载图片并在游戏窗口中渲染这些图像。这些功能通过`SDL_Renderer`和`SDL_Texture`对象实现，它们分别处理渲染的设备和渲染的内容。在坦克对战游戏中，SDL用于：

- **创建和管理游戏窗口：** 使用`SDL_CreateWindow`和`SDL_CreateRenderer`初始化游戏的主窗口和渲染器。
- **绘制图形：** 利用`SDL_RenderCopy`等函数，将坦克、子弹和地图元素（如墙体）绘制到屏幕上。

```
SDL_Texture* tankTexture = IMG_LoadTexture(renderer, "tank.png");
SDL_Rect dstRect = {x, y, width, height}; // 坦克的位置和尺寸
SDL_RenderCopy(renderer, tankTexture, NULL, &dstRect); // 将坦克图像渲染到窗口
```

## 事件处理

SDL的事件管理系统允许游戏响应用户输入，如键盘和鼠标事件。在本游戏中，SDL用来捕捉玩家的操作指令（如移动坦克和发射子弹），确保游戏能够互动和响应玩家的每一个动作。

```
SDL_Event event;
while (SDL_PollEvent(&event)) {
    if (event.type == SDL_QUIT) {
        running = false;
    } else if (event.type == SDL_KEYDOWN) {
        handle_key_down(event.key.keysym.sym);
    }
}
```

### 音频管理

SDL还支持音频文件的加载和播放，这对于增强游戏体验至关重要。在坦克对战游戏中，音效如射击声和爆炸声都是通过SDL来管理的。通过使用`Mix_OpenAudio`来初始化音频设备，以及`Mix_PlayChannel`来播放具体的音效，游戏的声音部分得以丰富和完善。

```
cpp复制代码Mix_Chunk* shotSound = Mix_LoadWAV("shot.wav");
Mix_PlayChannel(-1, shotSound, 0); // 在任一可用的频道上播放射击声音
```

### 综合作用

SDL作为游戏的底层支持库，使得开发者可以更加专注于游戏逻辑和用户体验的实现，而不需过多关注平台特异性的问题。它的跨平台特性也意味着用SDL编写的游戏能够在多种操作系统上运行，无需修改代码，极大地提高了游戏的可移植性。

## 六、游戏测试

### 登录界面测试

- **正常功能测试：** 输入正确用户名和密码，应能成功登录。
- **异常测试：** 输入错误密码，应提示错误。

### 游戏功能测试

- **坦克移动和旋转：** 检验坦克是否可以根据用户输入正确移动和旋转。
- **子弹发射：** 子弹应按照坦克的朝向正确发射并移动。

## 七、 收获

- **对面向对象编程的理解更为深入：** 游戏开发过程中大量使用了对象和类。
- **能更加熟练的使用Qt：** 通过实际项目应用Qt，理解其信号与槽机制。
- **有了编写较大规模代码文件的经验：** 本项目包括多个类和多种功能，增强了我对大型项目的管理能力。