高级语言程序设计

实验报告

南开大学 计算机大类
姓名：李锦程
班级：计算机一班
2025 年 5 月 15 日

目录

一. 作业题目

基于 SDL2 的肉鸽游戏的多种实现。

二. 开发软件

1、 开发语言：C++

2、 集成环境：Visual Studio 2022

3、 图形化库：

（1）SDL2（图形渲染）

(2) SDL2_image（图片处理）

(3) SDL2_mixer（音效处理）

(4) SDL2_ttf（文字渲染）

(5) SDL2_gfx（几何图形辅助绘制）

三. 课题要求

1、 运用 C++ 面向对象编程能力，独立完成一个具备交互功能的程序 学生需充分掌握类的定义、继承与多态机制，能够独立设计程序的对象模型并实现其行为逻辑。在项目开发过程中，要求能够根据游戏的运行结构抽象出适当的类（如玩家、敌人、控制器、界面渲染器等），实现逻辑清晰、封装性良好的代码架构

2、 使用图形库，实现动态界面渲染与用户输入响应 项目应选用合适的图形库实现游戏场景的图像绘制和实时更新。界面需要具备动态元素（如敌人出现、子弹飞行、面部追踪）和用户输入响应机制（如键盘输入、摄像头捕捉、单词检测等），以确保程序具备交互性与实时反馈性能。

3、 具备完整的数据组织结构、良好的代码风格和可维护性 代码结构应采用清晰的头文件与源文件分离方式进行模块划分，类之间的耦合度要低、职责分明。在逻辑设计上要求保持变量命名规范、注释齐全、格式统一，以提升代码的可读性与维护性。使用合适的容器（如 vector、map 等）管理游戏对象的生命周期和状态更新。

4、 项目逻辑清晰，模块化程度高，便于后续功能扩展 整个程序应采用模块化设计思路，将功能划分为若干独立子模块（例如：游戏逻辑模块、图像处理模块、

用户输入模块、状态管理模块等），每个模块之间通过明确接口进行通信。该架构需支持后续加入新功能（如得分系统、难度调节、联网对战等），具备良好的扩展性和适配性。

四. 主要流程

1．版本 1.0——基础肉鸽游戏的实现

在本实验项目的构建过程中，我们围绕 2D 射击类交互场景，采用模块化的设计思想，逐步完成了基础类、渲染系统、动画系统、角色系统与交互系统的搭建。以下将按照项目构建的逻辑顺序，逐一介绍核心类的功能与作用:

1.1 基础工具类的设计

(1) Vector2 类

该类为项目中所有坐标、速度、方向等二维数据的基础表示形式。它封装了常用的向量运算，如加减、数乘、点积、归一化与模长计算等，确保了代码中的位置与速度计算既简洁又具可读性。

(2) Timer 类

Timer 是整个动画与行为调度的时间控制核心。它支持单次或循环计时，能够通过设置回调函数实现定时行为的触发，如动画帧的切换或摄像机震动结束，是驱动系统动态行为的关键组件。

1.2 渲染系统的构建—— Camera 类

Camera 管理整个画面的视角偏移与渲染接口。它支持画面抖动效果的控制，增强击打反馈的表现力。其 render_texture 方法负责对带旋转角度和中心点偏移的纹理进行正确渲染，是角色与场景最终视觉呈现的统一出口。

1.3 资源管理与动画系统

（1）Atlas 类

Atlas 类是图像资源的集中管理器，用于加载图集或精灵图帧。它支持按路径模板批量加载图像资源，也可从一张精灵图中拆分多个帧，为动画系统提供素材来源。

（2）Animation 类

Animation 类负责帧动画的更新与控制。它通过 Timer 类驱动帧间切换，可设置是否循环播放、帧间隔时间与播放结束时的回调函数。动画对象可被绑定到任意角色，用于表现其不同状态（如奔跑、爆炸等）。

1.4 对象系统的实现

（1）Bullet 类

Bullet 是玩家发射的子弹实体。它根据输入角度计算速度向量，实现任意方向的直线飞行。子弹具备自动失效机制（飞出屏幕范围时无效），并提供渲染与被击中处理方法。

（2）Chicken 类

Chicken 表示普通敌人单位。其生成位置具有随机性，在屏幕上方随机出现后匀速下落。Chicken 包含运行与爆炸两种动画状态，被子弹击中后切换动画并播放音效，动画播放完毕后可自动从场景中移除。

（3）Boss 类

Boss 是一个抽象基类，用于定义复杂敌人角色。它封装了生命值管理、动

画切换、爆炸效果与背景音乐触发等完整行为逻辑。Boss 的移动行为通过纯虚函数 on_move 留给派生类实现，以支持多种不同移动模式。Boss 的死亡行为会触发爆炸动画并播放音效，且在首次受到攻击时启动背景音乐播放。

1.5 主文件

本项目的主函数负责游戏初始化、资源加载、主循环控制以及各类游戏对象的更新与渲染，构成程序的顶层框架。其主要内容可归纳为以下几个步骤：

(1) 初始化阶段

程序首先通过 SDL 库进行基本图形系统的初始化，包括窗口创建与渲染器初始化。随后调用 `IMG_Init` 初始化图像模块、`Mix_OpenAudio` 启动音频模块，并设定窗口尺寸（1280×720）。同时，设置帧率控制变量（FPS = 60）以保证动画播放的流畅性。

(2) 资源加载阶段

加载各类资源，包括背景图片、子弹纹理、Boss 图集、爆炸图集等，同时实例化动画图集（`Atlas`）并将其传入到动画类或 Boss 构造函数中。此阶段还包括音效和背景音乐的加载，用于增强玩家的视听体验。

(3) 对象创建与初始化

创建主摄像机对象 `Camera`，并实例化游戏角色，如 `Chicken`（普通敌人）、`Bullet`（子弹）、以及 `Boss`（Boss 敌人）。这些对象将被统一维护于容器中，便于后续循环中统一更新与渲染处理。

(4) 主循环控制

程序通过一个 while 循环实现游戏主循环，循环持续直到窗口关闭。在每一帧内完成如下操作:

① 事件处理：响应用户输入

② 状态更新：调用所有对象的 `on_update()` 方法，更新其位置、状态、动画帧等。

③ 碰撞检测：判断子弹是否击中敌人，调用 `on_hurt()` 改变敌方状态。

④ 渲染操作：清空画面后调用所有对象的 `on_render()` 方法，绘制背景与各角色。

⑤ 帧率控制：通过时间延迟方式控制每帧耗时，以维持稳定帧率。

(5) 资源释放与退出清理

主循环退出后，程序依次释放所有动态加载的资源（如纹理、音效、音乐等），销毁 SDL 窗口与渲染器，并调用相关退出函数清理内存，确保程序结束后系统资源不被占用。

(6) 小结

该 `main` 函数逻辑清晰，采用模块化对象管理，各类实体对象各司其职。程序实现了从资源加载、对象构建、主循环更新到渲染输出的完整流程，充分体现了面向对象编程与事件驱动机制在交互式图形项目中的应用，具有良好的可扩展性与可维护性。整个项目以主函数为调度核心，各类模块有机结合，共同实现了一个基础但完整的 2D 游戏系统。

1.6 总结

以上各类在系统中层层构建、逐级依赖。基础类 Vector2 与 Timer 提供数学与

时间支持；Camera 提供画面控制；Atlas 与 Animation 搭建动态表现系统；Bullet、Chicken、Boss 作为场景中的实体对象，在动画与交互基础上，体现出完整的行为逻辑。通过这样的分层设计，使得系统结构清晰，便于维护与拓展，成功实现了一个具有动效与交互的完整 2D 射击实验平台。

2．版本 2.0——拼写单词肉鸽小游戏

本版本在继承 1.0 版本基础射击逻辑的同时，引入“拼写单词”与“Boss 肉鸽机制”作为核心创新点，使游戏从简单的射击行为进化为具备教育属性、节奏策略性和阶段性挑战的综合系统。其主要区别和提升体现在以下几个方面：

(1) 引入拼写单词模块，实现寓教于乐

与 1.0 版本的“纯射击交互”不同，2.0 加入了单词拼写系统。游戏在运行期间会在屏幕中间展示一个英文单词的中文释义，玩家需逐字母拼写该单词。每输入一个正确字母即触发一次发射动画并生成子弹，实现“击打与拼写行为”的融合。该机制通过 `current_word`, `player_input` 以及事件监听系统实现，并在 `mainloop` 中处理玩家按键逻辑。正确完成一个单词将获得额外得分，并自动刷新新的拼写目标，使游戏兼具教育性与操作性。

(2) Boss 系统登场，强化阶段性挑战与视觉表现

本版本首次引入多个 Boss 敌人（Boss1、Boss2、Boss3），并通过分数门槛触发不同阶段的 Boss 出现。Boss 采用专属动画与爆炸效果，具备独立的生命周期与攻击检测逻辑，为玩家带来更具压力的战斗节奏。程序在 `on_update` 中加入对 Boss 生命周期的管理、自动锁定射击方向的调整，并在 `on_render` 中专门绘制 Boss 图层。

(3) 自适应瞄准系统，提升交互便捷性

为提升玩法流畅性，本版本取消了原版本中由鼠标控制炮管旋转的手动瞄准方式，改为自动对准当前最近的目标（包括敌人和 Boss）。程序在 `on_update` 中引入对敌人 Y 坐标的比较，实时判断并自动调整 `angle_barrel` 角度，实现智能辅助瞄准，提高了游戏节奏与操控效率。

(4) 游戏节奏控制强化：发射反馈与屏幕震动

发射动作由 `animation_barrel_fire` 控制，并配有三种随机音效和 `camera->shake()` 屏幕震动，使每一次正确拼写都伴随强烈的物理反馈与视觉刺激。这一设计显著增强了玩家在学习过程中的成就感与节奏感。

(5) 单词词库动态加载

游戏启动时从外部文本文件 `words_list.txt` 加载英文-中文单词对，存储于 `std::map` 中，具备较高扩展性。未来可轻松替换词库以适应不同教育主题（如小学英语、托福单词等）。

3．版本 3.0——人脸识别控制射击方向

版本 3.0 在保留版本 2.0 中“自动开火+拼写判定”的基础上，首次打通了“计算机视觉识别结果”与游戏行为之间的控制链，实现了以**人脸在画面中的水平位置偏移**为输入，动态控制炮管的旋转角度，使射击方向始终跟随玩家面部位置变化。这一特性显著增强了游戏的人机交互体验，并实现了更具沉浸感的控制方式。

(1) **控制方式从键盘/鼠标转向面部控制**

相比于前两个版本中以鼠标移动（或自动锁定目标）实现炮管旋转，3.0 版

本取消了传统鼠标输入逻辑，转而从摄像头获取视频流，并通过 face-api.js 实时检测人脸边框。通过计算人脸中心点与画面几何中心的水平偏移量，控制变量 angle_barrel 随之动态调整，从而实现精准的方向控制。

(2) **Web 前端模块与 SDL 后端的协同工作**

项目通过一个前端 HTML 页面（配合 face-api.js）处理视频输入与人脸检测逻辑，在画布中绘制识别框并实时计算偏移量；而 C++/SDL 后端程序根据偏移量输入调整 angle_barrel，保持游戏内炮管与人脸方向对齐。这种“双端耦合”结构是版本 3.0 的技术亮点，实现了跨平台控制通道的打通。

(3) **偏移量转换为角度输入的映射机制**

在前端识别系统中，通过检测人脸中心点与画面中心在 X 轴上的偏移量，可以推导出一个线性映射关系：偏移量越大，炮管偏转角度越明显。后端程序可通过约定接口或共享数据方式读取该偏移值并计算旋转角度，从而控制发射方向。该机制有效替代了鼠标事件 SDL_MOUSEMOTION 的处理逻辑。

(4) **交互性显著增强，游戏更具沉浸感**

玩家只需面对摄像头左右移动，即可完成炮管指向敌人的操作。结合原有自动发射与动画抖动机制，极大增强了沉浸式体验，适合作为体感类互动项目的原型系统，具有推广到教学、康复、游戏控制等多个场景的潜力。

(5) 小结

版本 3.0 的核心创新在于打通了“视觉识别结果”与“物理反馈行为”之间的数据闭环，标志着本项目从键盘/鼠标控制向自然交互（NUI）范式的过渡。通过人脸识别作为射击方向输入源，程序实现了更加智能、灵敏且具有个性化的控制方式，为今后的高互动性游戏与智能设备控制提供了良好实验范式。整体结构仍保

持 SDL 为渲染骨架、face-api.js 为视觉识别支撑的双引擎框架，具有很强的拓展性与跨平台能力。

五. 单元测试

1. 向量类 Vector2 的运算测试

测试了向量加法、减法、数乘、点积、归一化等基本运算，确保其满足代数规律，例如：

```
(Vector2(1, 2) + Vector2(3, 4)).x == 4
Vector2(0, 3).normalize().length() ≈ 1
```

2. 子弹类 Bullet 的边界检测

通过构造多个不同角度发射的子弹，验证其位置更新逻辑与边界判断是否正确。例如：初始位于画面中心的子弹在速度下移出边界后，其 `can_remove()` 应返回 `true`。

3. 定时器 Timer 的回调触发

设置不同的 wait_time 与 on_timeout 回调函数，验证定时器能否在正确时间触发。例如：设置 `wait_time = 1.0s` 后，调用 `on_update(1.2f)` 应触发一次回调。

4. 动画类 Animation 的帧更新测试

测试动画是否能根据帧间隔切换帧并在结束时调用 on_finished。模拟播放完所有帧后：动画 index 是否归零或停止

5. 鸡类 Chicken 的死亡逻辑

测试 Chicken 在调用 `on_hurt()` 后是否能正确切换动画状态，以及 `can_remove()` 是否在爆炸动画完成后返回 true。

6. 拼写控制模块模拟测试（版本 2.0）

通过模拟连续输入正确拼写的字母序列，验证 `player_input` 能够被正确累计，并在完成拼写时刷新新的单词词条，分数增加是否正确。

7. 人脸识别偏移量模拟测试（版本 3.0）

构造伪检测结果，测试识别坐标向炮管角度 angle_barrel 的转换是否合理。例如：模拟偏移 +100 px，应得到右偏若干度的 angle_barrel

虽然未使用专业测试框架（如 Google Test 或 Jest），但通过这些功能性测试保证了项目中关键类和逻辑块的运行正确性，增强了系统的可靠性与可维护性。今后可逐步引入自动化测试框架实现更完善的回归验证机制。

六. 收获

1. **掌握了完整的 SDL 图形编程流程**

通过多个版本的迭代开发，我系统学习并掌握了 SDL 库在图像加载、音效处理、事件捕捉、动画渲染等方面的使用方法，建立了从窗口初始化到实时渲染的完整思维框架。

2. **实现了模块化面向对象设计**

项目中将炮弹、敌人、Boss、动画、定时器等功能分别封装为类，有效提升了代码的可读性与可维护性，强化了我对面向对象程序设计的理解与实践能力。

3. **增强了游戏机制设计与节奏控制能力**

通过设计自动开火、敌人分型、分阶段 Boss 挑战等机制，我体会到了游戏节奏与玩家反馈之间的平衡原则。特别是在引入音效、屏幕抖动、动画切换等细节后，对交互性与沉浸感的营造有了更深认识。

4. **初步掌握了人脸识别技术及其应用接口**

在版本 3.0 中，我首次接触并成功集成了基于 `face-api.js` 的人脸检测功能，实现了从摄像头获取实时人脸坐标并用于控制射击方向的功能。这一过程让我了解了浏览器端计算机视觉的基本原理，并学习了如何在前端与后端之间建立数据交互通道。

5. **提升了跨模块协作与调试能力**

本项目涉及 C++ 后端逻辑与 HTML/JavaScript 前端识别的协同调试，尤其在人脸偏移量与炮管角度映射环节中，需要不断进行参数调试与逻辑修正，提升了我对多技术融合项目的整合与调试能力。