南开大学 计算机大类

姓名：谢雨婷

学号：2311623

班级：0979

2025年5月4日

高级语言程序设计

实验报告

目录

[高级语言程序设计大作业实验报告 2](#_Toc197236721)

[**一.** **程序题目** 2](#_Toc197236722)

[**二.** **开发软件** 2](#_Toc197236723)

[**三.** **功能要求** 2](#_Toc197236724)

[**四.** **主要流程** 3](#_Toc197236725)

[**1． 整体流程** 3](#_Toc197236726)

[**五.** **单元测试** 9](#_Toc197236727)

[**六.** **收获** 9](#_Toc197236728)

高级语言程序设计大作业实验报告

1. **程序题目**

音频播放与记时。该程序的核心目标是整合环境音频播放与专注计时功能，为用户营造一个沉浸式的工作或学习氛围，从而辅助用户提升专注力与效率。

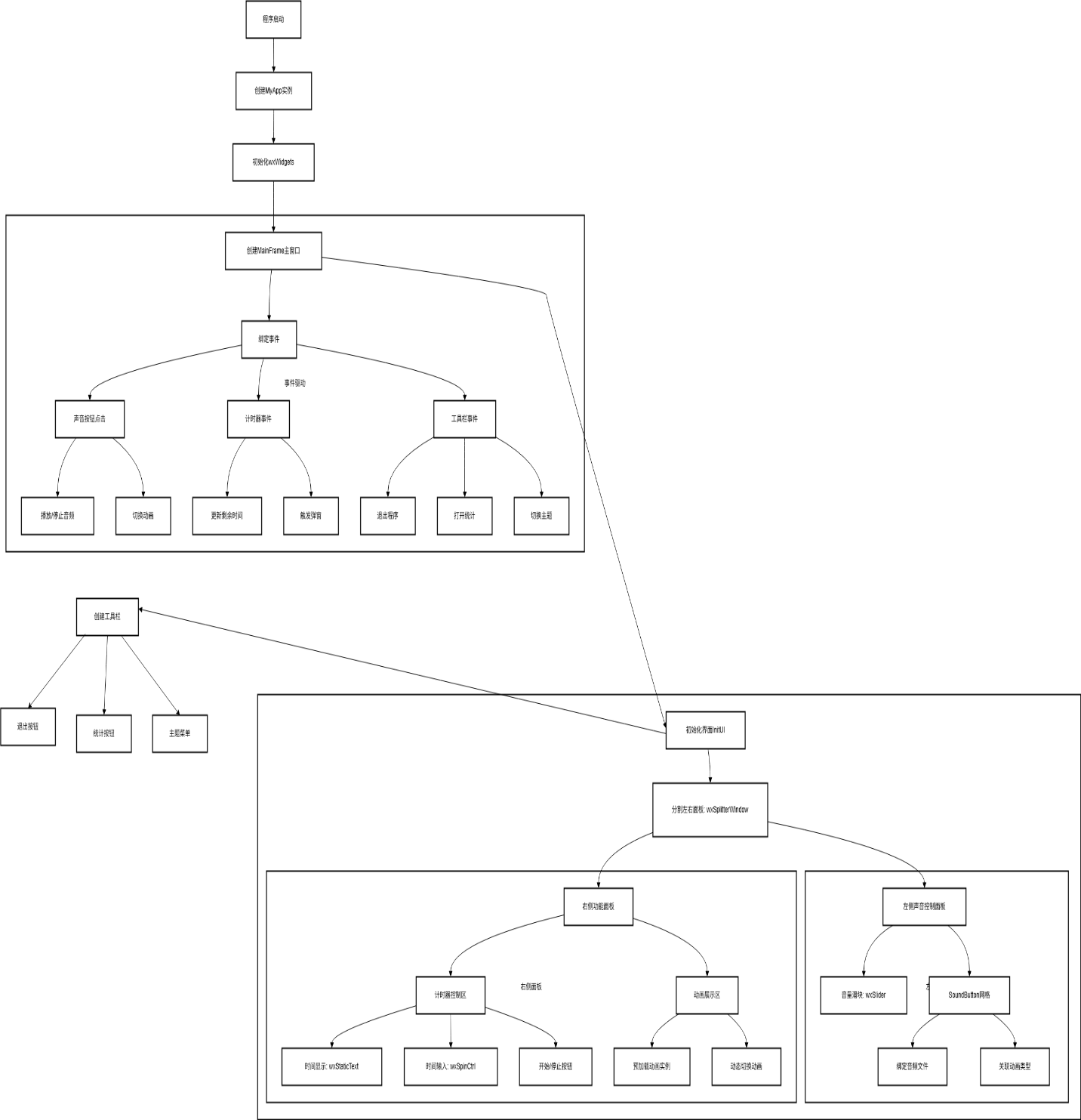
1. **开发软件**

开发环境：Code::Blocks,MinGW, 图形框架：wxWidgets, 音频处理：SFML

1. **功能要求**
2. 多元环境音播放: 内置多种环境音效（雨声、咖啡馆嘈杂声、森林风声、海浪拍岸声、炉火燃烧声、冰川碎裂声），并支持音效的循环播放及用户选择切换。提供可定制的专注计时器（支持分钟级别设置）
3. 音量调节: 提供直观的全局音量控制机制，允许用户根据偏好调整背景音量。
4. 可定制专注计时器: 用户能够自由设定专注时长（以分钟为单位，范围1-120分钟），程序需实时、清晰地展示剩余时间。专注记录统计与可视化
5. 提醒与记录: 专注时段结束后，应有明确的弹窗提示；同时，系统需自动记录每次完成的专注会话信息（起始时间与持续时长），并将其持久化存储。
6. 数据可视化: 提供专注数据的统计查阅功能，采用趣味性的“成就卡片”形式展示用户的累计专注成果。
7. 视觉反馈与个性化: 根据当前播放的音效，同步显示相应的动态视觉动画（如雨滴下落、雪花飘舞、火焰跳动等），增强沉浸感；提供多种界面主题选项（包括预设的 Nord Light、Nord Dark 主题以及用户自定义颜色方案），满足用户的个性化审美需求。
8. **主要流程**
   1. **整体流程**

应用程序的启动入口位于 main.cpp，通过 MyApp 类（继承自 wxApp）实例化并展示主窗口 MainFrame。MainFrame 作为程序的核心，负责整体界面的构建、用户交互的响应以及各功能模块的协调。其初始化过程 (InitUI) 设计了界面布局：采用 wxSplitterWindow 将主工作区分割为左右两个区域。左侧为可滚动的声音控制面板 (CreateSoundPanel)，包含音量滑块及一系列 SoundButton 按钮，用于选择不同的环境音。右侧面板 (rightPanel) 则集成了上方的计时器控制区 (CreateTimerPanel) 和下方的动画展示区 (CreateAnimationPanel)。

程序的核心交互逻辑围绕用户的操作展开：点击声音按钮触发音频播放/停止及对应动画的切换；操作计时器控件启动或终止专注时段；通过工具栏访问主题设置和统计数据。整个系统基于 wxWidgets 的事件驱动模型，确保了用户操作的即时响应和各模块间的有效通信。其大致流程可由下图概括：



实现思路：

1. **程序启动**
   * main.cpp 创建 MyApp 应用实例，初始化 wxWidgets 图像处理模块，创建 MainFrame 主窗口，设置程序图标，并显示窗口。
2. **界面初始化**
   * MainFrame 构造函数：设置窗口基本属性，绑定定时器事件，调用 InitUI。
   * InitUI:创建 wxSplitterWindow 分割界面；分别调用 CreateSoundPanel, CreateTimerPanel, CreateAnimationPanel 构建左侧声音面板、右侧计时器面板和动画面板；创建并配置工具栏 (m\_toolbar)，添加退出、统计、主题切换等按钮及下拉菜单；加载历史专注记录 (LoadRecords)；应用初始主题 (ApplyNordLightTheme)；注册主题更新回调。
3. **声音控制 (CreateSoundPanel, SoundButton.cpp):**
   * **CreateSoundPanel**： 创建 wxSlider 用于全局音量控制，并绑定 wxEVT\_SLIDER 事件，该事件会调用所有 SoundButton 的 SetVolume 方法。创建 wxGridSizer，循环生成多个 SoundButton 实例，每个实例关联一个音效标签和对应的 .wav 文件路径。绑定 wxEVT\_TOGGLEBUTTON 事件，触发声音播放/停止 (ToggleSound) 和动画切换 (SwitchAnimation)。
   * **SoundButton 类**：继承 wxToggleButton，在构造函数中调用 LoadAudio 方法使用 SFML 的 sf::SoundBuffer 加载音频文件。ToggleSound 方法根据按钮状态调用 sf::Sound 的 play() 或 stop()。SetVolume 方法调整 sf::Sound 的音量。UpdateTheme 方法根据 ThemeManager 更新按钮的背景色、前景色和字体。
   * **状态管理**：remainingSeconds 记录剩余时间，isRunning 标记运行状态。
4. 计时器模块（CreateTimerPanel, MainFrame::OnTimerTick）：
   * CreateTimerPanel：创建 wxStaticText (timeDisplay) 显示时间，wxSpinCtrl (timeInput) 设置分钟数（绑定 wxEVT\_SPINCTRL 更新初始显示），以及 wxButton (startBtn, stopBtn) 控制计时（绑定 wxEVT\_BUTTON）。
   * 点击 "开始"：设置 isRunning = true，计算 remainingSeconds，启动 wxTimer (timer->Start(1000))。
   * 点击 "停止"：调用 timer->Stop()，设置 isRunning = false，调用 RecordSession 记录本次专注时长（如果有效），重置计时器显示。
   * OnTimerTick (由 wxTimer 每秒触发)：若 isRunning 且 remainingSeconds > 0，则 remainingSeconds-- 并更新 timeDisplay；若 remainingSeconds == 0，则停止计时器，弹出提示框，记录会话，重置显示。
5. 动画系统（CreateAnimationPanel, AnimationBase.h, \*Animation.cpp）：
   * CreateAnimationPanel：创建 wxPanel (animPanel) 作为动画容器。循环创建所有具体的动画类实例（如 RainAnimation, FireAnimation 等，均继承自 AnimationBase），并将它们添加到 animPanel 的 Sizer 中，初始状态设为隐藏。将动画实例存入 animMap 以便后续查找。绑定 wxEVT\_SIZE 事件，确保动画区域大小变化时，所有动画实例能调整自身尺寸。
   * AnimationBase：定义动画类的公共接口（Start, Stop, OnPaint, OnTimer, UpdateTheme）和基础成员（timer, m\_isActive, frameCount）。OnPaint 和 OnTimer 为纯虚函数，由子类实现具体绘制和逻辑更新。提供 UpdateTheme 虚函数用于响应主题变化。
   * 具体动画类（如 RainAnimation.cpp）：实现 Start (启动定时器，设置 m\_isActive=true)，Stop (停止定时器，m\_isActive=false)，OnPaint (使用 wxAutoBufferedPaintDC 进行绘制，如画雨滴线条)，OnTimer (更新动画状态，如雨滴位置，并调用 Refresh() 触发重绘)。大多使用粒子系统模型，结合 <random> 制造随机性
6. 数据记录与统计(MainFrame::LoadRecords, SaveRecords, RecordSession, OnShowStatistics):
   * FocusRecord结构体：存储单次专注记录 (std::chrono::system\_clock::time\_point startTime, int durationSec)。
   * LoadRecords：程序启动时调用，使用 std::ifstream 和 std::istringstream 从 focus\_data.csv 文件读取历史记录，解析时间字符串（std::get\_time）并填充 focusRecords 向量。
   * SaveRecords：当 focusRecords 更新后调用（如 RecordSession 后），使用 std::ofstream 将 focusRecords 中的所有数据写入 CSV 文件，格式化时间（std::put\_time）。
   * RecordSession：在计时结束或手动停止时调用，计算本次有效专注时长，创建 FocusRecord 对象，添加到 focusRecords 向量，并调用 SaveRecords。
   * OnShowStatistics：响应工具栏按钮点击。遍历 focusRecords，计算总专注时长等统计量。调用 CreateAchievementCard 创建包含统计信息（如专注时长、等效价值）和图标的 wxPanel 卡片，并将这些卡片添加到弹出的 wxDialog 中展示给用户。
7. 主题管理（StyleConfig.h,MainFrame::Apply\*Theme,:UpdateTheme）:
   * ThemeConfig 结构体 (StyleConfig.h)：定义主题包含的各种颜色（如 mainBg, cardBg, primary, textPrimary）和字体 (titleFont, timeFont)。预定义了 Nord Light/Dark 的颜色常量。
   * ThemeManager 类 (StyleConfig.h):提供 Get(), SetTheme(), GetCurrentTheme()等接口。
   * MainFrame 中的 Apply\*Theme 方法：创建对应的 ThemeConfig 对象，调用 ThemeManager::Get().SetTheme()。
   * UpdateTheme 方法（在 MainFrame, SoundButton, AnimationBase 中实现）：作为 ThemeManager 的回调函数。当主题改变时被调用，从 ThemeManager 获取当前主题配置，并更新自身及子控件的背景色、前景色、字体等视觉属性，最后调用 Refresh()、Update()、Layout() 应用更改。
8. **单元测试**
9. 计时器逻辑 (MainFrame 相关逻辑):
   1. 测试案例: 设置不同初始时长（如1分钟、60分钟），模拟启动计时器
   2. 验证点: 验证 remainingSeconds 是否按秒递减；验证 timeDisplay 显示的格式是否恒为 "MM:SS" 且数值正确；验证计时结束后 isRunning 状态是否变为 false，是否触发了 RecordSession 调用，以及是否弹出了提示。
   3. 结果：一开始计时结束后不断弹出结束的对话框，修改前后代码执行顺序后解决重复出现的问题。
10. 数据持久化 (MainFrame::LoadRecords, SaveRecords):
    1. 测试案例: 准备包含不同记录数量（0条、1条、多条）、特殊时间点（如跨年、闰年）的模拟 CSV 文件；测试写入后再读取数据的一致性。
    2. 验证点: LoadRecords 能否正确解析 CSV 文件并填充 focusRecords；SaveRecords 生成的 CSV 文件格式是否正确，内容是否与 focusRecords 一致；对空文件或格式错误的文件的处理是否健壮
    3. 方法: 使用文件操作模拟读写，比较读出的数据结构与预期是否一致
11. 音频控制 (SoundButton):
    1. 测试案例: 使用有效的 .wav 文件路径和无效路径创建 SoundButton；模拟按钮点击事件。
    2. 验证点: LoadAudio 对有效/无效文件的返回值是否正确；ToggleSound 是否能正确改变内部播放状态；SetVolume 是否正确设置了内部音量值。
    3. 方法: 创建 SoundButton 实例进行测试，。
12. 动画逻辑 (\*Animation 类):
    1. 测试案例: 针对特定动画类（如 RainAnimation），模拟 OnTimer 事件多次触发。
    2. 验证点: 粒子是否按预期生成、移动、变化（如透明度、大小）和消失；粒子数量是否在合理范围内。
    3. 方法: 创建动画类实例，调用 Start() 后模拟 OnTimer 事件，检查内部粒子集合 (m\_particles, m\_drops, m\_flakes 等) 的状态。
13. **收获**

本次高级语言程序设计大作业——"focused" 专注计时器的开发过程，是一次全面而深入的 C++ 应用实践。通过本次项目，我在以下几个方面获得了显著的提升和宝贵的经验：

1. **GUI 编程与 wxWidgets:** 熟练掌握了使用 wxWidgets 构建功能完善的桌面应用程序的核心技术，包括窗口与控件布局（熟练运用 Sizers 和 Splitter）、事件处理机制（Bind 与事件表）、自定义绘图（尤其是在实现多种动态背景动画方面积累了经验）以及跨平台 GUI 开发的基本原则。
2. **多媒体集成 (SFML):** 成功地将 SFML 音频库无缝集成到 wxWidgets 项目中，实现了背景音效的加载、循环播放、实时音量控制等功能，理解了不同库之间协同工作的接口与方法。
3. **面向对象设计原则的应用:** 通过设计 AnimationBase 抽象基类及其派生体系，以及功能封装良好的 SoundButton 类，我更深刻地体会了继承、多态、封装等面向对象思想在构建模块化、可维护、易扩展软件系统中的实际价值。特别地，对基类虚析构函数 (virtual ~AnimationBase()) 重要性的认识得到了巩固，确保了多态场景下资源的正确释放。
4. **C++ 语言与标准库的深化运用:** 项目中广泛应用了 C++11 及后续标准的特性（如 Lambda 表达式简化事件绑定、auto 类型推导），并熟练运用了标准库中的核心组件，包括但不限于容器 (std::vector, std::unordered\_map)、时间处理 (<chrono>)、随机数生成 (<random>)、文件流操作 (<fstream>, <sstream>) 以及算法库 (<algorithm>)，提升了代码的现代性和效率。
5. **软件工程实践能力的提升:** 对 GUI 应用中常见的事件驱动编程模型有了更扎实的理解。掌握了使用 CSV 文件进行简单数据持久化的实用技术。通过对单元测试重要性的认识和设计思考，提升了对软件质量保障的意识。

**总结:** 本次大作业成功实现了一款集成了音频、视觉、计时与数据管理功能的 "focused" 专注辅助工具，达到了预期的设计目标。整个开发过程不仅是对 C++ 编程技能和第三方库应用能力的一次综合检验，更是一次宝贵的软件设计与工程实践锻炼。面对挑战并最终完成项目，极大地增强了本人的技术自信，并为未来承担更复杂的软件开发任务奠定了坚实的基础。