回忆长廊项目作业报告

一、程序功能介绍

本项目名为**回忆长廊**，是一款基于Qt 框架开发的互动游戏式相册。主要功能包括：

- **横向卷轴游戏界面** 用户可以通过键盘（WASD、空格、Shift）控制主角角色在横向卷轴场景中移动、跳跃、加速，感受类似马里奥的游戏体验。
- **动态背景与音乐** 游戏背景支持动态切换，用户可以在设置页面选择不同的背景和背景音乐，营造个性化氛围。
- **相框添加与管理** 用户在游戏中可以通过鼠标点击添加相框（包含图片、文字、时间戳），这些相框会按顺序排列在长廊中，记录下玩家的回忆。
- **相册页面** 除了游戏界面，程序还提供一个专门的相册页面，展示所有相框，支持查看、编辑、删除、保存和加载相框数据。
- **设置页面** 用户可以通过设置页面自定义主角角色形象、背景图像，并对游戏体验进行调整。
- **数据持久化** 所有相框信息可以保存到文件中，并在程序启动时加载，实现完整的数据持久化。

二、项目模块与类设计细节

项目采用模块化设计，主要模块和核心类包括：

1️⃣ 主窗口模块(MainWindow)

(1) 程序功能介绍：MainWindow 模块

MainWindow 是程序的主入口窗口，承担以下核心功能：

- 提供主菜单，供用户进入游戏、查看相册或调整设置。
- 通过 `QStackedWidget` 管理多个页面（主菜单、相册、游戏），实现界面切换。
- 管理和转发跨模块的信号，例如：从设置页传来的角色、背景、BGM 修改信号，会通过 MainWindow 转发到 GamePage。
- 负责加载和管理项目的全局资源，例如读取相框数据文件、初始化并播放背景音乐。

- 动态调整界面大小，例如游戏界面进入全屏、主菜单恢复普通窗口。

(2) 项目模块与类设计细节：MainWindow 模块

MainWindow类

- 继承自 `QMainWindow` ，是整个项目的顶层主窗体。

主要成员组件

- `QStackedWidget *stackedWidget`：页面堆叠管理器，用于切换主菜单、相册、游戏页面。
- `QWidget *mainMenuPage`：主菜单页面，含有项目主题名、开始、相册、设置、总结报告按钮。
- `AlbumPage *albumPage` ：相册页面，展示由 `PhotoFrameManager` 管理的相框集合。
- `GamePage *gamePage`：游戏页面，包含 `GameView` ，展示滚动场景和相框互动。
- `SettingsPage *settingsPage`：设置页面，独立窗口，用户可调整角色、背景、BGM 等。
- `PhotoFrameManager m_photoFrameManager`：管理项目中所有相框数据的加载、保存、统一访问。
- `QLabel *themeLabel` 、 `QPushButton *startButton` 等: 主菜单上的控件，用于界面交互。
- `QMediaPlayer *bgmPlayer` 、 `QAudioOutput *bgmAudioOutput`：音乐播放器，用于播放和管理背景音乐。

主要功能

- **初始化构造**
  - 设置主界面、布局、加载各个页面。
  - 初始化信号槽连接，例如：
    - 按钮点击切换页面。
    - `SettingsPage` 发出的信号传到 `MainWindow` ，再由它转发到 `GamePage` 。
    - 当进入游戏页面（index == 2），自动切换到全屏并将焦点转交 `GameView` 。
  - 加载本地保存的相框数据文件，填充相册和游戏场景。
  - 初始化背景音乐播放器，播放默认 BGM。
- **事件响应**
  - `resizeEvent()` ： 根据窗口大小动态调整字体大小，保证主菜单界面在不同分辨率下美观。
  - `showSettingsPage()` ： 首次打开时动态创建 `SettingsPage` ，并连接信号槽。
  - **信号处理**
    - `onCharacterImageChanged()` 、 `onCharacterScaleChanged()` 等：接收设置页信号，转发到游戏界面。
    - `onBgmTrackChanged()` ： 切换背景音乐并循环播放。
    - `onBgmVolumeChanged()` 、 `onBgmMuteToggled()` ： 调整音量或静音。

- **资源管理**
  - 自动加载保存的相框数据，统一由 `PhotoFrameManager` 提供。
  - 音乐播放器采用 Qt 6 的 `QMediaPlayer` 和 `QAudioOutput`，支持音量、静音、循环播放等操作。

2️⃣ 游戏模块（GamePage）

(1) 程序功能介绍：GamePage模块

`GamePage` 模块是整个程序的 **核心游戏界面模块**，它提供了：

✅ 横向卷轴式游戏视图（GameView），包含角色、背景和场景；
✅ 角色控制（WASD/方向键移动、空格/Shift 跳跃和加速）；
✅ 场景背景平铺、角色精灵帧动画播放、角色与视角同步；
✅ 主菜单、设置页面跳转按钮；
✅ 相框加载与显示功能（通过 `PhotoFrameManager` 和 `PhotoFrameItem` 管理）。

在这个模块中，用户可以体验完整的游戏操作（例如左右移动、加速、跳跃等），并且可以动态加载和展示之前保存的相框数据，形成独特的互动体验。

(2) 项目各模块与类设计细节

GamePage类

- 继承自 `QWidget`

职责：

- 整合游戏视图（GameView）；
- 提供主菜单、设置、加载相框等按钮；
- 管理和协调用户界面与底层逻辑的交互。

主要成员组件

- `GameView *gameView` ： 游戏场景视图。
- `QPushButton *backButton` ： 返回主菜单按钮。

- `QPushButton *settingsButton` ： 进入设置页面按钮。
- `QPushButton *loadButton` ： 加载相框按钮。
- `PhotoFrameManager *frameManager` ： 相框数据管理器。

主要方法：

- `setFocusToGameView()` ： 将焦点集中到游戏视图上。
- `setCharacterImage(const QString &imagePath)` ： 设置主角精灵图片。
- `setCharacterScale(double scale)` ： 设置主角缩放比例。
- `setCharacterYOffset(int offset)` ： 设置主角垂直偏移。
- `setBackgroundImage(const QString &path)` ： 设置背景图片。
- `onCharacterSpeedChanged(int speed)` ： 修改主角移动速度。
- `loadAndDisplayFrames()` ： 加载并显示相框。

GameView类

- 继承自 `QGraphicsView`

职责：

- 管理游戏场景（QGraphicsScene）；
- 渲染背景、角色、相框等元素；
- 处理角色移动、跳跃、重力、速度等物理逻辑；
- 播放角色动画帧；
- 响应键盘事件。

主要成员：

- `QGraphicsScene *scene` ： 场景。
- `QGraphicsPixmapItem *backgroundItem` ： 背景图片项。
- `QGraphicsPixmapItem *characterItem` ： 角色图片项。
- `QTimer *moveTimer` ： 移动控制定时器。
- `QTimer *animTimer` ： 动画控制定时器。
- `QVector<QPixmap> walkFrames` ： 角色行走帧图集。
- 位置与速度参数（ `groundY` ， `frameWidth` ， `frameHeight` ， `verticalVelocity` 等）。
- 状态参数（ `isJumping` ， `movingLeft` ， `movingRight` ， `isSpeedUp` 等）。

主要方法：

- `initScene()` ： 初始化场景，包括背景、角色。

- `updateCharacterPosition()` ： 更新角色位置和物理状态。
- `updateAnimationFrame()` ： 播放下一个动画帧。
- `keyPressEvent()` / `keyReleaseEvent()` ： 键盘事件处理。
- `setCharacterImage(const QString &imagePath)`： 重新加载角色帧图。
- `setBackgroundImage(const QString &path)`： 设置背景图。
- `setCharacterScale(double scale)` ： 缩放角色。
- `setCharacterYOffset(int offset)` ： 调整垂直位置。
- `setCharacterSpeed(int speed)` ： 修改角色速度。
- `loadFramesAndDisplay()` ： 加载并展示相框数据。

🔶 类：PhotoFrameManager

文件： `PhotoFrameManager.h`

职责：

- 管理相框数据（图片路径、描述、时间戳等）；
- 提供保存、加载等持久化接口。

主要方法：

- `loadFrames()` — 从文件或目录加载相框数据。
- `saveFrames()` — 保存相框数据。
- `getFrames()` — 获取当前相框列表。

PhotoFrameItem类

职责：

- 继承 `QGraphicsPixmapItem` ，扩展图片项，附带图片路径信息。

主要方法：

- `getImagePath()` — 获取图片路径。

模块之间的协作关系

模块/类

交互方式

GamePage ↔ GameView

GamePage 控制 GameView，调用其方法来调整角色、背景、速度、焦点等。

GameView ↔ PhotoFrameManager

GameView 调用 PhotoFrameManager 加载相框数据，并将相框渲染为 PhotoFrameItem。

GameView ↔ PhotoFrameItem

GameView 将每个相框作为 QGraphicsItem 添加到场景。

3 **设置模块(SettingsPage) **

(1) 程序功能介绍：SettingsPage模块

该设置模块为程序提供一站式个性化配置，主要包括以下功能：

1 背景设置

- 提供多张默认背景图片供用户选择。
- 允许用户从本地文件选择自定义背景图片。
- 所选背景即时预览并通过信号通知游戏主页面刷新显示。

2 声音设置

- 切换不同的背景音乐（BGM）曲目。
- 调整背景音乐音量。
- 静音或取消静音。

3 游戏设置

- 调整角色移动速度（如人物的跑步、跳跃速度）。
- 提供滑动条界面，让用户实时调整，灵活体验不同速度感。

4️⃣ 角色设置

- 更换主角外观（支持男生、女生多款造型）。
- 改变主角在画面中的缩放比例（如变大或变小）。
- 调整主角在画面中的垂直偏移量，微调视觉位置。

5️⃣ 桌宠设置

- 选择一个本地图片作为桌宠（目前是示例功能，预留扩展接口）。

(2) 项目各模块与类设计细节

SettingsPage

总控页面，包含 `QTabWidget` 多个子页面。负责子页面的初始化、信号转发，将子页面发出的设置变化通知主界面。

BackgroundSettingsPage

提供背景图片选择（默认 + 自定义），点击按钮时发射 `backgroundImageChanged(path)` 信号。

MusicSettingsPage

提供音乐曲目选择、音量调节、静音切换，分别发射 `musicTrackChanged(path)`、`musicVolumeChanged(value)`、`musicMutedChanged(state)` 信号。

GameSettingsPage

提供角色速度调节滑块，发射 `characterSpeedChanged(value)` 信号。

CharacterSettingsPage

提供角色外观选择、大小缩放、Y轴偏移，分别发射 `characterImageChanged(path)`、`characterSizeChanged(scale)`、`characterYOffsetChanged(offset)` 信号。

PetSettingsPage（示例）

提供选择桌宠图片的按钮，未来可扩展为桌宠设置模块（当前仅打印日志，未实现信号与槽联动）。

4️⃣ 相册模块（AlbumPage ）

(1) 程序功能介绍：AlbumPage模块

AlbumPage 是项目中负责展示、管理和操作所有相框（PhotoFrameWidget）的界面模块。
它提供了一个可视化相册页，允许用户：

- 浏览所有保存的相框（含图片、描述、日期）；
- 添加新相框（包括选择图片、填写描述、指定日期）；
- 删除已有相框；
- 编辑已有相框（包括更换图片、修改描述和日期）；
- 按时间顺序（拍摄日期 + 创建时间）排序显示相框；
- 保存与加载相册数据到文件，实现持久化存储。

界面布局通常采用 `FlowLayout` 或 `QGridLayout` ，确保相框排列整齐、齐平、统一大小、间距一致，带来美观的用户体验。

(2) 项目各模块与类设计细节

AlbumPage 类

主要成员组件

- `QVector<PhotoFrameWidget*> frameWidgets` ： 当前显示的相框控件列表
- `PhotoFrameManager manager` ： 底层数据管理器
- `QScrollArea` ： 滚动显示区域
- `QWidget + FlowLayout` ： 相框展示区
- `QPushButton` ： 添加、保存、加载，返回按钮

主要信号槽

- 槽：添加相框、删除相框、保存到文件、从文件加载、刷新界面
- 数据持久化：通过 `PhotoFrameManager manager` 的 `saveToFile()` 和 `loadFromFile()` 进行 JSON 文件读写

PhotoFrameWidget 类

功能

单个相框控件，显示一张图片、描述、日期，支持点击、双击、右键等操作

主要成员组件

- `ClickableLabel`： 图片区域
- `QLineEdit`： 描述输入框
- `QLabel`： 日期显示

特殊操作

- 左键点击：添加或查看图片
- 右键点击：弹出删除确认
- 双击：弹出编辑对话框

持久化接口

`toJson()` 和 `fromJson()`： 将自身状态保存为 JSON 或从 JSON 恢复

PhotoFrameManager 类

功能

纯数据管理类，不负责界面，只存储 PhotoFrameData 列表

数据结构

`QVector<PhotoFrameData> m_frames`： 保存所有相框数据

核心操作

`saveToFile(QString)`： 保存到 JSON 文件

`loadFromFile(QString)`： 从 JSON 文件加载

用途

AlbumPage 用于保存和恢复相框数据状态

注意

仅管理数据，不管理 PhotoFrameWidget

📦 PhotoFrameData 类说明

这个类是用于表示单个相框的数据模型，包含一张图片、描述文字、创建时间、日期等信息。

主要成员组件

- `imagePath`： QString 图片的文件路径
- `description`： QString 相框的文字描述
- `creationTime`：QDateTime 相框的创建时间戳（通常是毫秒）
- `date`：QString: 人类可读的日期字符串（如 2025-05-24）

🔑 关键流程

1️⃣ 加载相册

- AlbumPage 调用 manager.loadFromFile(file)
- 遍历 manager.getAllFrames()，为每个数据生成对应的 PhotoFrameWidget
- 添加到界面布局中展示

2️⃣ 保存相册

- AlbumPage 遍历当前界面中的 PhotoFrameWidget
- 更新 manager.m_frames 列表
- 调用 manager.saveToFile(file) 保存到 JSON 文件

3️⃣ 添加新相框

- AlbumPage 创建一个新的 PhotoFrameWidget，放入布局中
- 也将其数据（PhotoFrameData）添加到 manager.m_frames 中

4️⃣ 删除相框

- 用户右键点击相框 → 弹出确认框
- AlbumPage 删除对应的 PhotoFrameWidget 和数据项

5️⃣ 编辑相框

- 用户双击相框 → 弹出对话框修改
- AlbumPage 刷新布局，更新 manager.m_frames

三、小组成员分工情况

李致远

桌宠互动与AI

- 设计并实现桌宠自定义控件（QWidget），包含动态形象（帧动画/状态切换）。
- 实现桌宠的点击、拖拽、右键菜单等鼠标交互。
- 实现桌宠情绪反馈机制（基于应用状态调整形象）。
- 实现右键菜单的“年度报告”“AI对话”功能入口。
- 实现AI对话界面，集成NLP API（如deepseek），处理网络请求及JSON解析。
- 管理桌宠与用户的AI交互体验。

任金洋

数据分析与年度报告（总结报告）

- 设计年度报告展示界面（如独立QDialog或专用视图）。
- 实现与UI核心模块的数据接口对接，获取全部相框数据。
- 开发数据分析功能，包括记录频次统计、时间分布分析、高频关键词提取。
- 实现图表/可视化展示（用QtCharts或QPainter绘制）。
- 整理总结报告、撰写开发文档。

钟熙胤

UI核心与内容管理

- 搭建项目主窗口与整体UI布局。
- 实现横向滚动走廊视图。
- 实现自定义背景（主题选择、图片加载）。
- 实现BGM播放与控制。
- 设计并实现“相框”自定义控件（QWidget），含任务名、日期、缩略图、部分感想等展示。
- 开发相框编辑面板，包括图片上传、文字输入、日期选择。
- 实现相框数据的存储（如JSON或SQLite）和读取。

- 管理相框的创建、展示、销毁。
- 提供接口供年度报告和桌宠模块调用相框数据。

> **协作说明**：
> 各模块通过提前定义好的数据结构（如相框数据类/结构体）、接口函数、信号与槽连接，形成松耦合、高协作的开发方式。每个成员专注模块内部细节，同时通过Git版本控制和定期会议保持团队同步。

四、项目总结与反思

本项目开发过程中，我们不仅分工明确，而且通过模块化设计和接口定义提高了开发效率和代码质量。以下是我们的主要收获与反思：

✅ 项目收获

- 熟悉了 **Qt框架** 中的多模块协作、信号与槽、界面设计、QWidget自定义控件等技术。
- 掌握了 **动态界面交互**（如桌宠拖拽、右键菜单）与 **后台数据处理**（如年度报告分析、NLP API对接）的综合实现。
- 提升了 **项目管理能力**，包括任务拆分、接口设计、代码版本控制、团队协作沟通等。
- 通过总结报告，深入理解了需求分析、设计、开发、测试、文档撰写的完整项目开发流程。

⚠ 项目不足与改进点

- 模块初期的接口定义不够早，导致部分功能对接时需要返工。改进建议：下一次项目中，**接口先行、实现后置**，确保并行开发顺畅。
- 团队内部部分进度沟通不够频繁，导致中期有短暂的开发卡点。改进建议：制定固定的周会/日报，及时同步进度和遇到的问题。
- 年度报告的数据分析深度有待加强，目前主要以可视化为主，未来可以引入更多文本挖掘、情感分析等丰富内容。

💡 总体反思

这是一次非常有挑战和成长的团队合作经历。大家分工清晰、责任明确，通过互相支持和持续调整，最终完成了一个兼具功能性与趣味性的桌宠+年度总结项目。我们认识到，优秀的项目不仅需要良好的技术实现，还需要高效的协作机制、清晰的任务拆解和不断优化的开发习惯。