项目概述

项目名称

DesktopPet - 跨平台桌面宠物应用

项目背景

现在大家居家办公、上网课的时间越来越长，整天对着电脑屏幕很容易觉得枯燥疲惫。我们想开发一个轻量级的桌面宠物应用，给用户的工作学习环境加点“小乐趣”。桌宠能带来一点互动感，缓解视觉疲劳，提升使用电脑的心情。

项目目标

我们计划使用 Qt6 框架开发一个能在不同操作系统（如 Windows, macOS, Linux）上运行的桌面宠物应用。核心目标包括：

- **养只“活”宠物**：在桌面上显示可爱的动画宠物，并且能自由切换不同造型。
- **互动玩耍**：支持用鼠标和宠物进行多种互动（比如拖拽、点击）。
- **个性化定制**：允许用户对宠物外观和互动行为进行一定程度的自定义。

项目需求分析

功能性需求

核心功能

1. **宠物显示系统**
   - 流畅播放 GIF 动画作为宠物动作。
   - 支持显示静态图片模式。
   - 宠物窗口背景透明（只看到宠物，没有方框）。
   - 窗口无边框设计，更自然融入桌面。
2. **交互功能**
   - 能用鼠标随意拖拽宠物在桌面上移动。
   - 右键点击宠物弹出菜单（用于切换、退出等操作）。
   - 可以在不同宠物形象之间切换。
   - 提供便捷的退出应用方式。
3. **界面管理**
   - 宠物窗口始终保持在其他窗口之上（不会被挡住）。
   - 窗口大小能根据宠物图片/动画自动调整。

- 确保用户操作（拖拽、点击）流畅不卡顿。

扩展功能（争取实现）

- 提供更多种宠物类型供用户选择。
- 让宠物有不同状态（比如发呆、走路、睡觉），并能自动切换或响应事件。
- 支持用户自定义编辑宠物（比如换装、加装饰）。
- 保存用户设置（如最后使用的宠物、位置等）到配置文件。
- 其他有趣的功能（如定时提醒、宠物小游戏等，视开发进度而定）。

技术开发规划

技术架构

整体设计思路

我们采用 **MVVM (Model-View-ViewModel)** 架构来组织代码，让各部分职责更清晰，方便开发和维护：

- **Model (数据层)**：负责管理宠物的核心数据（位置坐标、当前状态、播放的动画等）。
- **ViewModel (逻辑层)**：处理用户操作（如拖拽、菜单点击）的业务逻辑，作为 View 和 Model 之间的桥梁。
- **View (视图层)**：负责把宠物画出来显示在屏幕上，并接收用户的鼠标键盘操作。

这种结构让界面显示(View)和后台数据(Model)分离，通过 ViewModel 来连接和协调。

核心模块设计

1. **数据模型 (Model)**
   - `PetModel`： 管理宠物数据（位置、状态、动画资源路径等）。
   - `PetInfo`： 定义一个结构体，存放单个宠物的所有信息。
   - 使用 Qt 的信号槽机制实现数据变化时自动通知界面更新。
2. **视图模型 (ViewModel)**
   - `PetViewModel`： 核心逻辑所在地。处理拖拽移动、菜单命令、切换宠物等操作的业务逻辑。
   - 使用 Qt 的命令模式封装用户操作。
   - 负责将 Model 的数据变化反映到 View，并将 View 的用户操作通知 Model。
3. **视图层 (View)**
   - `PetMainWindow`： 主窗口类，负责显示宠物。
   - 基于 Qt Widgets 框架构建。
   - 处理鼠标事件（按下、移动、释放、右键点击）并触发 ViewModel 的命令。
   - 根据 ViewModel 提供的数据更新宠物显示（位置、动画帧）。

技术选型

- **编程语言**：C++17
- **UI框架**：Qt 6.5+
- **构建工具**：CMake 3.14+
- **测试框架**：Google Test (用于单元测试和集成测试)
- **开发工具**：
  - **代码编辑器**：Visual Studio Code (轻量，跨平台，插件丰富)。
  - **版本控制**：Git (管理代码版本，团队协作必备)。

开发流程规划（分阶段推进）

1. **第一阶段：搭建项目框架 (约 2-3 天)**
   - 搭建项目目录结构，配置好 CMake 构建脚本。
   - 实现 MVVM 架构的基础骨架（定义好 Model、ViewModel、View 的接口和基本交互）。
   - 创建最基础的透明无边框窗口，能显示一张静态宠物图片。
   - 配置好所有组员的开发环境（VSCode + Qt + CMake + Git）。
   - 注意设计好模块接口，降低耦合度。
2. **第二阶段：实现核心玩法 (约 1-2 天)**
   - 让静态宠物动起来：实现 GIF 动画的加载和流畅播放。
   - 实现鼠标拖拽移动宠物的功能。
   - 实现右键点击弹出功能菜单（包含切换宠物和退出选项）。
   - 完成宠物切换功能（加载不同的宠物资源并显示）。
3. **第三阶段：打磨完善 & 测试 (约 1 天)**
   - 加入必要的错误处理和异常恢复机制。
   - 编写单元测试和集成测试用例，用 Google Test 跑起来。
   - 优化性能（内存占用、CPU使用率、动画流畅度）。
   - 在目标平台（至少 Win + 一个 Linux）上测试运行效果。
   - 编写中期检查文档和初步的用户使用说明。
4. **第四阶段：拓展功能 & 持续集成 (结题前)**
   - 根据时间和兴趣，逐步添加计划中的扩展功能（多状态、自定义编辑等）。
   - 配置持续集成(CI)流程（如 GitHub Actions），自动编译和运行测试。

小组成员分工安排

姓名	学号	主要职责
陈诺	3230103847	**架构 & 核心逻辑**：负责 ViewModel(VM) 和 Model(M) 模块的设计与实现。中期前完成基础框架搭建。

姓名	学号	主要职责
李俊希	3230104189	**数据管理 & 功能 & 交付**：负责 Common、App层开发， 定义模块接口与命令模式；协助 Model 层开发，管理宠物数据加载、 状态逻辑、配置文件读写等。进行功能测试和跨平台测试， 确保软件质量，并负责最终应用的打包部署工作。
高梓云	3230105902	**界面 & 交互 & 质量保证**：负责 View 层开发，实现处理宠物显示、 动画播放、用户鼠标事件、设计菜单等UI交互部分。 编写集成测试用例。