TongDesktopSprite 项目报告

王翔宇，苗雨旸，吕俊辰

June 30, 2024

一、程序功能介绍

TongDesktopSprite 项目(以下简称“本项目”)主要实现了三部分功能。第一部分功能是普通桌面宠物/助手（以下简称“桌宠”）的常见功能，包括：在显示器最上层显示动画，实现桌宠的鼠标事件响应，以及搭载小工具实现辅助功能。第二部分是编写了桌宠的命令行修复程序，实现了对桌宠关键配置&资源文件夹（主程序目录下的 repos 文件夹）的初始化和修复。第三部分是实现了桌宠的通用可视化拓展接口，即用户可以通过程序界面自主引入美术资源和工具程序，并配置鼠标事件和两者之间的链接关系。可以说本项目不是一个单纯的桌宠程序，而是一个通用的桌宠制作框架。

二、项目各模块与类设计细节

（一）项目结构总览

本项目完全基于 Window11 系统、C++17 编程语言、QT（version 6.7.0）框架和 QT Visual Studio 扩展实现，项目结构和代码架构分别如图 1、图 2 所示。

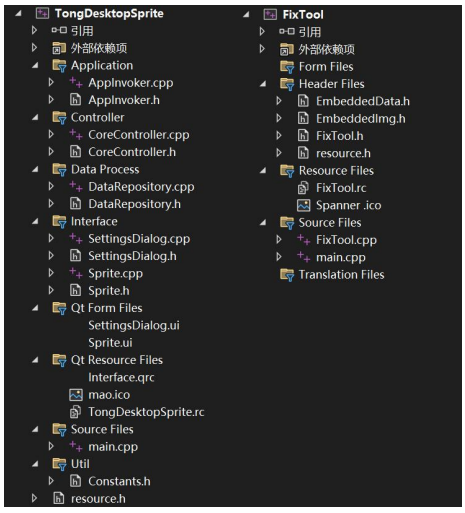

图 1

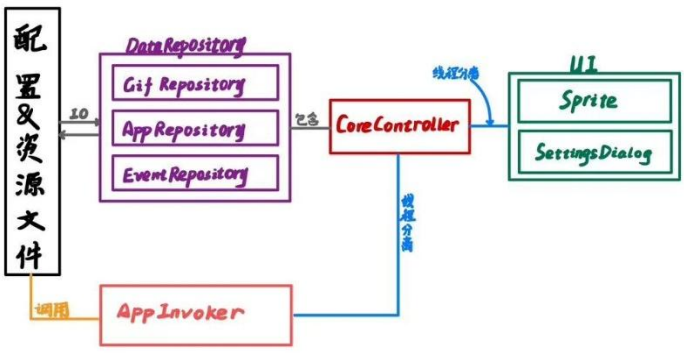

图 2

图 1 展示了项目主体部分（不包括修复程序）的结构。项目主程序由四个模块组成，分别是：负责显示和用户交互的 UI 模块，负责衔接数据仓库模块、外部工具程序调用模块以及 UI 模块的核心控制模块（CoreController），负责实现

资源与配置的管理与文件 IO 的数据仓库模块（DataRepository），负责实现程序调用和反馈收集的外部工具程序调用模块（AppInvoker）。

图 2 展示了项目代码的组织结构。主项目的四个模块与代码文件的对应关系为：UI 模块对应 Sprite.h、Sprite.cpp、SettingsDialog.h、SettingsDialog.cpp；核心控制模块对应 CoreController.h、CoreController.cpp；数据仓库模块对应 DataRepository.h、DataRepository.cpp；外部工具程序调用模块对应 AppInvoker.h、AppInvoker.cpp。项目文件中还包括必要的资源文件和 QT 框架中控制 UI 外观设计的.ui 文件。对修复程序的代码结构的介绍见下文对应小节。

（二）UI 模块介绍

UI 模块由两部分构成，Sprite 控制桌宠的动画显示与鼠标响应，SettingsDialog 控制桌宠可视化拓展面板的显示与交互。下图 3 展示了二者的运行效果。

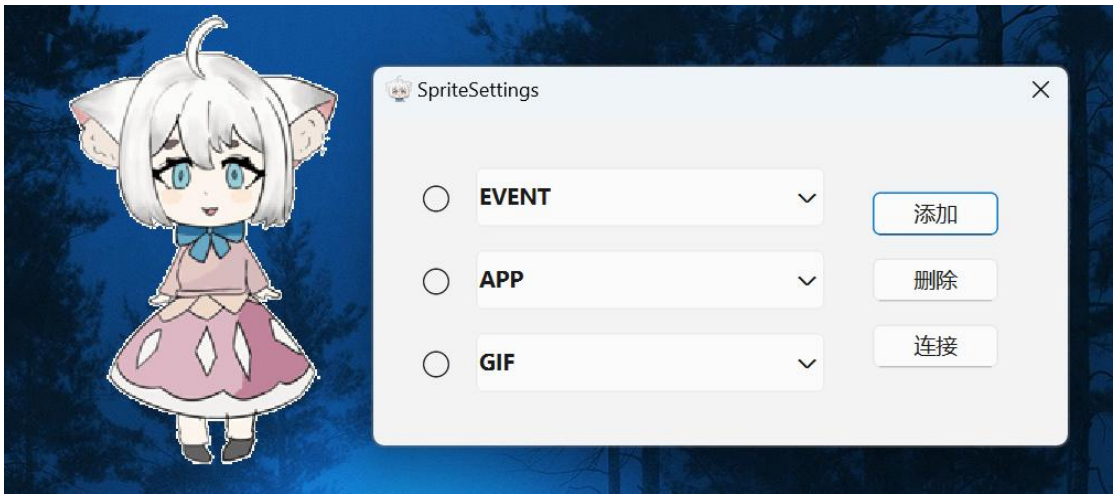

图 3

Sprite 文件中实现了 Sprite 类，该类继承 QMainWindow 类，在代码中通过 setAttribute 和 setWindowFlags 方法设置窗口无边框且透明背景。在窗口中通过 QLabel 和 QMovie 结合实现给定 gif 文件的动画播放效果。鼠标交互则是利用 QT 的事件系统中的 QMouseEvent 相关的事件处理函数的重载实现。

SettingsDialog 文件中实现了 SettingsDialog 类，该类继承 QDialog 类，其窗口布局主要通过 Qt Creator 编辑 SettingsDialog.ui 文件实现，其功能主要通过 QRadioButton、QComboBox 以及 QPushButton 组合实现。

（三）核心控制模块介绍

为实现程序前后端的分离，防止前后端的运行相互堵塞，本项目使用多线程

方法将程序显示和交互的前端与后端各模块分离运行。本节描述的核心控制模块就通过这种技术实现了与前端Sprite分离——具体方法是在Sprite类中内置一个QThread类型的private成员变量，并在Sprite初始化函数中动态创建（运用new关键字）CoreController指针，再通过moveToThread方法将CoreController托管于该线程进行运行。利用线程技术分离的两部分通过QT的信号函数和槽函数进行信息交换。

CoreController类本身继承QObject类以能够调用moveToThread方法，其内部直接内置了DataRepository模块，而其与AppInvoker模块的连接则是同上描述的线程分离的关系。该类主要内容就是大量信号与槽函数，以实现多个模块的信息交换。

（四）数据仓库模块介绍

该模块由基类DataRepository和三个派生类GifRepository、AppRepository、EventRepository组成。三者分别实现对repos文件夹下的gifs文件夹、apps文件夹以及events文件夹的管理。具体原理是三者先将文件夹内的目录文件content.dat读入内存，以实现资源信息的快速响应；在需要增添、修改、删除时，先修改内存中的内容，再通过文件操作将内存内容的改变落实到文件层面。其中涉及的大量文件操作主要通过QDir、QFile、QFileInfo、QDataStream实现，目录文件和内存中的数据结构为QSet和QHash实现的哈希表。

（五）外部工具程序调用模块介绍

该模块中的AppInvoker类实现了对符合要求（目前内置的可执行文件要求是可执行文件可接受单个文件地址作为参数）的可执行文件的调用，以及判断其运行是否出错的功能。该模块主要是通过C++中Windows相关API库ShlObj.h和Shellapi.h直接调用Windows系统接口实现。

（六）修复工具模块介绍

该部分独立于主程序存在，也单独编译成独立的可执行文件。该模块设计的初衷是为了解决主程序所需要的配置文件的初始化问题。配置文件为二进制文件，无法直接编辑（也不应该支持直接通过文本修改），那么最初版本的初始化就需要单独的程序写入初始信息。后经过功能拓展，实现了对同目录下repos文件夹多种缺损形式的检查和修复，故单独打包成一个修复工具。值得一提，为了提供

初始的可视界面，该工具内置了 5 个准备好的 gif 文件,会在修复缺损时自动导出。

（七）杂项

本项目的美术资源（7 个 gif 文件，其中 5 个内置于修复程序内，所有皆打包于 rc 文件夹的 gifs.7z 中）由同学绘制。

本项目提供两个供测试的可执行文件（功能分别是 word 转 pdf 和彻底删除文件，打包于 rc 文件夹的 apps.7z 中）由 python 编写并打包为可执行文件。

本项目将 gif 内置到修复程序内部的方法是通过程序直接生成含有原文件二进制信息的巨大 unsigned char 数组的头文件，该辅助程序（.py 文件，存储于 rc 文件夹中）由 python 编写并直接用 python 解释器运行。

三、小组成员分工情况

王翔宇：总体架构设计，全部 CoreController 模块，部分 AppInvoker 模块，部分 UI 模块，全部 DataRepository 模块，主程序各模块之间的连接与测试，全部修复程序，内置图片用的辅助 python 程序，报告撰写。

苗雨旸：大部分 AppInvoker 模块，两个供测试的可执行文件，视频录制。

吕俊辰：大部分 UI 模块。

四、项目总结与反思

本次 QT 大作业项目持续时间很长，涉及内容很多，是一次软件设计、类与对象、文件操作、线程技术、GUI 制作等各方面程序设计课程内容的紧密结合。令人欣慰的是，在学期的最后，我们成功实现了在立项时规划的各种功能，还额外增添了一些立项时没有考虑的模块，我想，这是对我们程序设计实习这门课程的学习成果最好的总结。与此同时，这次实践也不是一帆风顺的：从开始时对 QT 的一无所知，到过程中遇到的各种闻所未闻的 bug 和无从下手的待实现功能，再到最后结束时仍遗留的代码可读性较弱，冗余度较高、程序鲁棒性不强，难以应对非理想用户、项目测试不充分，可适配系统有限等五花八门的问题，这一路的开发过程是坎坷不断的。但是，在一切结束之后重新审视这段磕磕绊绊的旅程，我觉得更应该看见的是我们在解决问题时的“求索”，是我们在直面困难时的“勇气”，是我们在灵光闪现时的“创造”——这些是我们在程序设计之外的收获，也将会是我们求学道路上熠熠生辉的“启明星”。

最后，感谢小组中每一位成员的努力！感谢殷绍峰助教一学期的答疑与指导！感谢张勤建老师对我们的谆谆教诲！