程序设计实习大作业作业报告

一、程序功能介绍

主要功能	子功能	功能介绍
右键菜单	做出动作	用户可以让角色做出指定动作，包括向左/右行走、向上/下攀爬、跑步 5 个动作，到达屏幕边缘时会从屏幕另一侧出现
	设置	打开“设置”界面
	对话	打开“对话”界面
	最小化到托盘	最小化到托盘
	退出程序	退出程序
AI 对话	正常对话	正常对话
	打开文件/文件夹	可以通过输入文件/文件夹路径打开指定的文件/文件夹
	查找/读取文件	可以在文件夹中查找/读取指定文件
	访问网页	可以通过输入网址访问网页
桌面互动	随机游走	（可选）在桌面经过随机时间做出随机时长的随即动作
	拖拽	可以拖拽角色
	投掷	可以将角色丢出去，碰到边界会反弹（开启边缘弹跳）或消失后从角落出现（关闭边缘弹跳）
	鼠标点击事件	做出小幅度的受击动作并积攒怒气
	丢鼠标	怒气达到一定程度会在下次点击时将鼠标丢出
功能设置	自定义颜色	自定义角色颜色，可以在调色盘选色也可以从屏幕吸取颜色
	设置对话 API	自定义 AI 对话的 API
	设置对话 prompt	自定义 AI 对话的 prompt
	边缘弹跳/随机游走开关	是否开启边缘弹跳/随机游走
	恢复默认设置	重置

二、项目各模块与类设计细节

（一）、类设计

1．MyMainWindow：主窗口类

成员变量：

velocity_history(list)	存储鼠标移动速度历史记录
speed_sample_duration(int)	速度采样时间区间长度（毫秒）
throw_threshold(int)	抛出速度阈值（像素/秒）
gravity(int)	重力加速度（像素/秒²）
drag_threshold(int)	拖动判断阈值（像素）
angry_value(int)	愤怒值

is_dragging(bool)	拖动状态标记
offset(Optional[QPoint])	窗口拖动时的偏移量
press_pos(QPoint)	记录鼠标按下时的坐标
label(QLabel)	显示 GIF 动画
text_box(QLabel)	显示文字的文本框
color_effect(QGraphicsColorizeEffect)	控制 GIF 颜色效果
action_manager(ActionManager)	管理动作和动画切换
mouse_thrower(MouseThrower)	处理鼠标抛出逻辑
ai_window(ChatWindow)	AI 聊天窗口

成员方法：

__init__(self, parent=None) ：

构造函数

initUI(self)

初始化 UI 控件和布局

update_gif(self, gif_path, gif_speed=100)

加载 GIF 动画，默认播放速度 100

set_gif_color(self, color=None)

设置 GIF 颜色

show_text_box(self, text)

显示文本框（用于调试，无实际作用）

hide_text_box(self)

隐藏文本框（用于调试，无实际作用）

mousePressEvent(self, event)

处理鼠标按下事件，会记录按下时的全局坐标，初始化拖动相关参数，并重置状态。

mouseMoveEvent(self, event)

处理鼠标拖动事件，计算移动距离并添加新数据点，当超过阈值且不在下落状态时视为拖动，并播放拖动动作。

mouseReleaseEvent(self, event)

处理鼠标释放事件，鼠标释放时视当前速度播放掉落或抛出动画。

contextMenuEvent(self, event)

打开右键菜单

launch_ai(self)

打开 AI 聊天窗口

2．ActionManager：动作类

成员变量：

window(QMainWindow)	主窗口实例
settings_window(SettingsManager)	设置窗口实例
context_menu(QMenu)	右键菜单栏
action_menu(QMenu)	Actions 菜单栏（右键子菜单）
move_timer(QTimer)	窗口移动计时器
action_timer(QTimer)	限时动作计时器
auto_move_timer(QTimer)	随机游走计时器
throw_timer(QTimer)	抛体运动计时器
is_in_action(bool)	是否在执行动作

is_falling(bool)	是否抛出
auto_move_enabled(bool)	是否启用随机游走
can_bounce(bool)	是否启用边缘弹跳
config_file(str)	配置文件 settings.json 路径
min_interval(int)	最小移动时间间隔
max_interval(int)	最大移动时间间隔
default_gif_path(str)	待机动画路径
talk_gif_path(str)	对话动画路径
actions_config(str)	菜单动作配置
no_menu_actions_config(dict)	非菜单动作配置
direction(QPoint)	移动方向向量
walk_speed(int)	行走速度
run_speed(int)	奔跑速度
climb_speed(int)	攀爬速度
current_speed(QPoint)	当前速度向量
throw_speeed(QPointF)	抛体初速度
gravity(float)	重力加速度
tray_icon(QSystemTrayIcon)	托盘图标
tray_menu(QMenu)	托盘菜单
talk_action(QAction)	“对话”动作
minimize_to_tray_action(QAction)	“最小化到托盘”动作
exit_action(QAction)	“退出程序“动作
settings_action(QAction)	“设置“动作

成员方法：

__init__(window: QMainWindow):

构造函数，传入主窗口实例，加载配置文件，初始化定时器、动作菜单和托盘图标。

perform_action(action_name: str, duration: int = None):

播放动作 GIF，可设置播放时间。

perform_no_menu_action(action_name: str):

播放非菜单动作 GIF。

end_action():

停止当前动作并切换为待机动作

start_moving_window():

启动窗口移动

stop_moving_window():

停止窗口移动

move_window():

窗口移动事件

trigger_auto_move():

随机执行动作，模拟待机时的随机动作行为。

schedule_auto_move():

根据是否启用自动移动功能和配置的时间间隔，调度下一次随机动作。

handle_throw(initial_velocity: QPointF):

执行被抛出的动作，设置初速度并启动抛体运动定时器。

update_throw_motion():

基于抛体运动公式更新窗口位置，适配重力效果，并处理边界检测（如反弹、坠落）。

_come_back():

被扔出屏幕后，控制窗口返回屏幕内的方法，可以结合行走动作。

init_context_menu():

设置和初始化右键菜单，向菜单中添加动作和功能选项（如设置、退出、最小化到托盘）。

show_context_menu(position: QPoint):

在指定位置显示右键菜单。

show_talk_text():

显示对话框，同时播放对话动画，定时结束动作。

open_settings_window():

打开并激活设置窗口。

open_color_picker():

弹出颜色选择器，对窗口的 GIF 动画颜色进行修改（已移动到 SettingsManger）。

switch_to_default_gif():

重置动画为默认的待机 GIF，且刷新播放。

flip_gif(horizontal: bool, vertical: bool):

原本用于翻转 GIF，现在不再需要。

init_tray_icon():

初始化系统托盘图标和托盘菜单，并设置点击恢复窗口等功能。

on_tray_icon_activated(reason):

当托盘图标被点击或触发时，恢复主窗口。

minimize_to_tray():

将主窗口隐藏，并显示系统托盘图标，用于实现最小化到托盘的功能。

3. SettingsManager：设置类

成员变量：

window(QWidget)	父窗口对象，可以通过此连接到主窗口。
config_file(str)	配置文件路径，默认值 settings.json。
settings_data(dict)	存储应用设置的字典，从 settings.json 读取。
font(QFont)	默认字体（微软雅黑）
default_prompts(str)	默认的 prompt
layout(QHBoxLayout)	窗口主布局
left_widget(QWidget)	左侧菜单框，用于承载按键。
left_layout(QVBoxLayout)	左侧菜单框对应的垂直布局。
right_widget(QWidget)	右侧内容框，用于动态展示功能页面。
right_layout(QVBoxLayout)	右侧内容框对应的垂直布局。
line(QFrame)	垂直分割线，用于视觉分离左侧和右侧内容。
color_button(QPushButton)	左侧菜单“Change Gif Color”按钮。
API_button(QPushButton)	左侧菜单“Setting API”按钮。
prompt_button(QPushButton)	左侧菜单“Setting Your Prompt”按钮。
others_button(QPushButton)	左侧菜单“Other Settings”按钮。
reset_button(QPushButton)	左侧菜单“Reset to Default”按钮。
color_dialog(QColorDialog)	颜色选择器，用于更新 gif 颜色（动态加载）。
line1(QFrame)	水平分割线。

成员方法：

__init__(self, window)：

构造函数，设置窗口、默认值、UI 风格、左侧按键和右侧区域的布局，加载配置、初始化 UI 布局、设置窗口属性。

update_right_content(self, option)：

通过 option 参数在右侧动态加载页面。

update_gif_color(self, color=None)：

更新和保存 gif 显示背景颜色。

reset_to_default(self)：

重置所有设置为默认值。。

set_API(self, api_key)：

设置并保存 API key 到本地。

set_prompt(self, prompt_text)：

设置并保存自定义 Prompt 到本地。

set_bounce(self, value)：

设置是否启用边缘弹跳。

set_auto_move(self, enable)：

设置是否启用边缘弹跳。

set_interval(self, min_val, max_val)：

设置随机游走的时间间隔。

load_settings(self)：

从配置文件中加载设置到 settings_data。

save_settings(self)：

将当前 settings_data 保存到本地 JSON 文件。

initialize_ui_from_settings(self)

根据设置文件中保存的值，初始化 UI 控件的默认值，更新 UI 中的颜色选择器、输入框默认提示、API key 及 Prompt 文本框等。

clear_layout(layout)：

功能函数，清空一个布局中的所有子控件。

4. MouseThrower：鼠标丢出控制类

成员变量：

timer(QTimer)	用于控制动画定时更新位置的计时器。
vx(float)	鼠标在 X 轴方向上的速度（像素/帧）。
vy(float)	鼠标在 Y 轴方向上的速度（像素/帧）。
current_x(int)	当前鼠标在屏幕上的 X 坐标位置（像素）。
current_y(int)	当前鼠标在屏幕上的 Y 坐标位置（像素）。
initial_speed(float)	鼠标抛出时的初始速度（像素/帧）。
deceleration(float)	速度衰减系数，用于模拟阻力（0 到 1 之间）。
bounce_factor(float)	鼠标反弹时剩余能量的比例。
min_speed(float)	动画停止的最低速度阈值（像素/帧）。
interval(int)	动画更新位置的时间间隔（毫秒）。

成员方法：

__init__(self)：

初始化类并调用 initAnimation 函数，设置动画的初始状态。

initAnimation(self)

初始化动画参数，包括速度、位置、计时器等。

startThrow(self):

启动抛出效果，随机生成初始抛出方向和速度，并开始定时器来更新位置。

updatePosition(self):

计算鼠标的新位置，处理碰撞检测（壁撞反弹），更新位置，并检查动画停止的条件。

5．AIChat：AI 对话类

成员变量：

类变量	response_received(pyqtSignal)	发送 AI 响应数据和完成的状态
	error_occurred(pyqtSignal)	发送错误信息
实例变量	user_input(str)	用户输入的文本数据
	_stop_flag(bool)	控制线程的停止信号
	buffer(list)	存储解析 JSON 指令过程的内容
	pending_text(list)	存储在 JSON 之外的普通文本片段
	is_json_response(bool)	是否当前正在处理 JSON 指令
	json_braces(int)	计数当前 JSON 的大括号对数
全局变量	conversation_history(list)	存储对话历史记录
	TEST_MODE(bool)	代码运行时是否在测试模式下执行
	client(客户端类)	AI 客户端对象，调用接口等
	model_name(str)	使用的 AI 模型的名称
	PROMPT(str)	系统的提示语
	SHOW_CMD(bool)	是否显示结果到响应文本中
	FileProcessor(类或模块)	文件处理器模块的外部接口

成员函数：

__init__(self, user_input):

构造函数，使用用户输入消息初始化。

run(self):

线程的主要执行逻辑，处理用户输入的文本消息、解析 JSON 指令、执行相应操作、将输出文本通过信号发送给界面等。

_is_valid_command(self, json_str):

判断给定字符串是否为有效的 JSON 指令。

_execute_command(self, full_response):

解析并执行 JSON 指令，完成具体的操作。

_get_system_drives(self):

获取系统上所有可用磁盘驱动器的根目录路径，返回一个包含磁盘路径的字符串列表。

_find_files(self, root_dir, pattern):

在指定目录下递归搜索符合模式的文件，限制搜索深度。

_generate_feedback(self, feedback):

在执行 JSON 指令后，生成自然语言形式的反馈信息。

6．FileProcessor：文件处理类

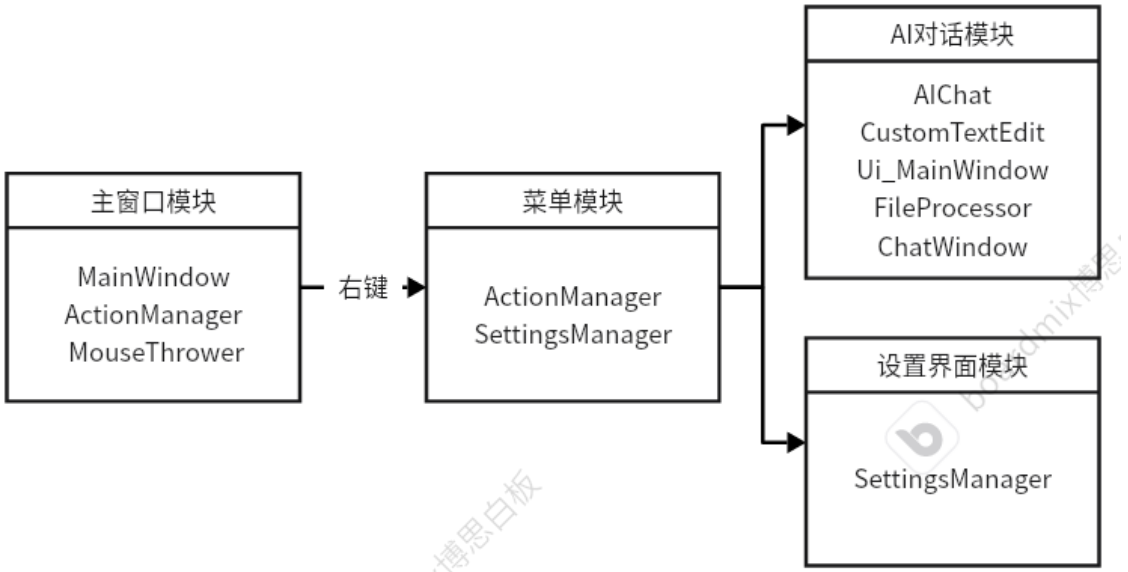
成员变量：

类变量 （无实例变量）	MAX_FILE_LEN(int)	限制读取文件内容的最大字符数

成员函数（均为静态方法）：

detect_file_type(path):
检测传入文件的类型。优先通过 filetype 库自动识别，若失败则返回文件扩展名。
process_file(path):
根据文件类型处理文件，返回处理状态和内容。
_process_pdf(path):
使用 pdfplumber 提取 PDF 文件的文本内容。
_process_text(path):
按多种编码尝试读取文本文件的内容。
_process_docx(path):
使用 python-docx 读取 Word 文档（.docx 格式）中的文本内容。

（二）、模块设计



1.主窗口模块

主窗口隐藏了 QMainWindow 的边框用以显示 gif，并且设置置于顶层来作为桌宠项目的主窗口。由于角色整体面积不大而且素材背景透明，相当于实现了鼠标穿透。

与主窗口角色的互动分为两大类：一种是直接通过鼠标的点击或拖动进行交互；另一部分通过右键的菜单选项进行互动。右键主窗口角色后接入右键菜单模块，暂不赘述。

在鼠标不点击的时候，角色处于待机状态，右键可以进行互动；如果开启随机游走，ActionManager 会从 SettingsManager 拉取 min_interval 和 max_interval，并在此范围的随机时间间隔触发随机的移动动作，包括左右行走和上下爬动。如果行走或爬动超越了屏幕边界，角色会将边界视为传送门，从屏幕另一侧出现。

鼠标单击，会触发 hit 动作，hit 会积累角色怒气值，达到一定程度时角色会将鼠标向随机方向甩出。该部分由 MouseThrower 类控制。

鼠标拖动时，会由待机动作变为拖拽动作，用户可以自己改变角色位置。MyMainWindow 类会检测鼠标松开时的速度，并决定角色是在当前位置停下并显示落地动画，还是显示投掷动画。投掷动画在开启“边缘弹跳”时会在屏幕内反复弹射直到落在地面，关闭“边缘弹跳”则会从飞出屏幕外，从屏幕对边的底部走出。该部分由 MyMainWindow 控制。

2.右键菜单模块

鼠标右键角色时，会显示右键菜单栏，Actions 菜单栏是其子菜单，均由 ActionoManager 类控制。菜单栏包括：“Action”(ActionManager)，“Settings”(SettingsManager)，“Talk”(ChatWindow)，“Minimize to Tray”(ActionManager)，“Exit”(ActionManager)。（括号内表示主要控制该功能的类）

Action 子菜单可以让用户自己控制角色做出动作，包括向左/右行走、向上/下攀爬、跑步 5 个动作，到达屏幕边缘时会从屏幕另一侧出现。

3.AI 对话模块

AI 对话模块通过 python 的 openai 库，接入 GitHub Models 的 API 进行会话，预设的 prompt 可以使给出的回答简短幽默。用户也可以在“设置”界面自己设置 API 和 prompt。

除了最基本的对话，也可以让角色打开文件/文件夹、查找文件和打开网页。该部分内容主要在子文件夹 chat 的（）类中实现。

与智能体对话的聊天记录存储在 chat_history.json 中，可以通过左下角按钮清除聊天记录并开启新话题。

4.设置界面模块

设置界面可以设置：角色颜色、自定义 API 和 prompt，是否开启边缘弹跳和随机游走功能以及设置随机游走的间隔时长。左侧为导航栏，依据左侧的选择不同，会在右侧栏改变显示内容。这部分完全由 SettingsManager 类控制，并将所有设置保存在本地的 settings.json 中，其他类可以从中调用，同时也支持热更新，不需要重启应用程序。

角色颜色的改变主要通过 PyQt6 自带的 QColorDialog 库实现，可以从调色板选色，也可以从屏幕上吸取颜色。

三、小组成员分工情况

- AI 聊天：李思源
- 动画素材：李瑞祥（大部分）、李思源（一点点）
- 桌宠框架：李瑞祥（移动、右键菜单、设置界面）
- 其他功能：李睿钦（点击、拖动、扔鼠标、被扔走等）
  李思源（随机移动）
- 视频与 PPT 制作：李思源

四、项目总结与反思

本项目灵感来源于一位同学桌面上一直挂着的桌宠“萝莉斯”和另一位同学对于桌面 AI 助手的想法。在三位成员的努力下，虽然离最初的设想有所差距，但是我们经过多次的讨论和尝试，最终在原有方案的基础上删去了不必要或难以实现的功能，同时加入了更多用户可以自定义的设置和类似于《虚拟桌宠模拟器》的动画功能。

这个软件的设计过程中涉及了非常多的技术和困难，例如如何调用 API？如何保证 AI 的回复不会过长而且有特定风格？怎么识别用户打开和读取文件的命令？桌宠的动画素材怎么解决？诸如此类。对此，我们学习了 API 调用、流式输出、Spine 等软件和技术，在 AI 的辅助下完成了相关的技术细节，手绘了相关的素材并制作动画，虽然实际效果与设想相去甚远，但是最终结果是让我们所以成员都满意的。

总的来说，我们小组成员由于都是第一次进行大型项目合作，缺乏相关经验，所以没有在早期做出十分明确的项目代码结构和人员分工，而且大家对 git 的使用也不是非常熟练，导致早期代码的同步出现了较大的问题，大家之间也会出现互相修改代码、调整功能的行为。虽然总的结果是好的，但是仍然出现了效率较低的问题。

而且在具体的代码方面，我们对 PyQt6 的学习也是走一步算一步，需要什么功能再去查询相关文档或借助 AI，导致了代码结构混乱、类之间没有保证单一职责原则的问题，甚至在代码后期加入新功能时出现了更多的 BUG，如实例循环初始化等。应该有更加严谨的代码结构设计和软件设计的相关知识。

这次小组合作可以说是一次宝贵的经历，为初次合作的我们积累了宝贵的合作经验和程序开发经验，也认识到了项目初期进行合理的人员分工的重要性。