Our-Lovely-Campus 一张北大的小地图

邓文骐　罗以楠　蒋诗勗

1. 初衷

我们为北大校园绘制了一张活点地图——这里没有教学楼、图书馆或东门的冰冷坐标，只记录了那些被宏大叙事遗忘的微小悸动。黑天鹅的优雅弧线、苍鹭掠过的水纹、壁虎在墙缝间的倏忽闪现、甲鱼在红湖底划出的隐秘轨迹... 这些鲜活的生灵才是这张地图真正的主角。

我们的地图能让你和这些小动物” 隔空互动”——点点屏幕喂喂黑天鹅，看看壁虎怎么溜达。用着用着，你会忍不住想：” 要不真去湖边看看？” 这就是我们的小心机：用电脑上的简单程序，勾着你去发现校园里藏着的真实乐趣。

熟悉，从来不是认知的终点，而是万物向我们展露新相的起点。当代码构建的苍鹭振动翅膀，现实中的它是否也在同步起飞？每个点击都在提醒：真正的探险不在远方，而在我们与这些微小生命无数次擦肩而过时，那个终于愿意驻足凝视的瞬间。

世界永远新鲜如初露，枯萎的只是我们凝视的目光。

2. 项目各模块与类设计细节

2.1. 程序功能

- 用户通过点击”Enter PKU” 按钮进入交互式地图界面
- 支持两种操作方式：
  - 键盘控制：使用 WASD 或方向键移动地图
  - 鼠标点击：直接点击地图进行导航
- 动态图标交互：
  - 悬停效果：鼠标停留在图标上时触发特殊动画
  - 点击功能：打开聊天对话框并显示对应实景图片

2.1.1. 运行时机制

- 所有图标实例存储在 imageviewer.allIcon 集合中
- 基于坐标的碰撞检测系统处理用户交互
- 世界时钟驱动所有动画同步更新

2.2. 项目结构

本项目的程序界面主要模块：

- coverwidget 进入窗口
- navigationwidget 主界面
- imageviewer 视图窗口
- movingicon 可移动的地图图标
- chatdialog 图标信息展示与智能交互

功能模块如下：

- pathgenerator 解析图标移动路径
- coverwidget 实现云雾拨开的效果

我们将所有功能模块拆分为不同文件，便于组员分别实现各自负责的功能，互不干扰。

2.3. 类设计细节

项目最关键的类是继承自 mapicon 的 movingicon 类。

mapicon 实现了：

- 动图效果
- 悬浮检测
- 点击检测

movingicon 在此基础上实现了沿轨迹移动功能。

类实例化需要传入以下参数：

- iconFilePath 图标的名称
- normalNum 正常状态的图片数量
- clickNum 鼠标悬浮后的动图数量

资源管理系统

```
Our-Lovely-Campus/
  rc/
     icon/[name]/
  # 名为name的图标动态效果所需的全部图片

     text/[name].txt
  # 名为name图标的文字描述

     photo/[name]/
  # 文件夹下为名为name图标的实拍图像

     path/[name].json
  # 移动路径数据
```

新增图标只需按上述目录结构添加相应资源 movingicon 的构造函数会自动根据传入的图标名称去相应文件夹寻找并获取：

- 动态效果
- 文字描述
- 实拍图片
- pathgenerator 解析后的 json 轨迹

2.3.1. 类实例化参数

- iconFilePath: 图标资源名称
- normalNum: 默认状态动画帧数
- clickNum: 悬停状态动画帧数

所有图标实例都储存在 imageviewer 的 allIcon 中。鼠标移动或点击后，根据位置判断触发效果的图标。imageviewer 具有一个世界钟向所有图标广播，从而实现动画效果。

---

3. 小组成员分工情况

3.1. 罗以楠

3.1.1. 主要功能

1. 添加背景音乐
2. 图标的移动的初始函数
3. 鼠标点击 + 悬浮的判定
4. 人机问答弹窗的实现和弹窗界面设计

3.1.2. 实现方式

1. 引入了 qt 自带的 multimedia 等模块，实现” 燕园情” 背景音乐的播放，并合理设置 new 和 delete 以分配内存。
2. 通过构建 mapicon、movingicon 等类，通过 QPixmap 实现图标显示，随后通过图片偏移量这一坐标确定了各个图标的位置，并初步通过一段 QPoint 给出了运动路程的离散化表示。
3. 在已有的坐标基础上，通过 QMouseEvent 类，判断鼠标的各种行为，并允许跟踪鼠标轨迹来判定是否悬停。
4. 通过接入 deepseekAPI 的形式，实现了人机问答。并使用可变化的绝对坐标的形式，设置了较为灵活的窗口。通过系统文档为每一个弹窗设定对应的身份。在考虑到既有的 QWidget 难以实现切换功能时，构建了 imagewidget 类来实现图片的切换和伸缩。

3.1.3. 亮点与不足

亮点：

1. 在显示图片时实现了图片预加载，一次性加载所有图片到内存中，避免了每次切换时重复的文件 I/O 操作。引入 m-scaledpixmap 缓存缩放后的图片，优化绘制性能。
2. 合理的对各个模块进行了职责分离，并设定了较为清晰的控制接口。

不足：

1. 对 GitHub 的使用熟练度欠缺，在初步的文件传输上耗费了很多时间。
2. 初步的设想过于宏大，缺乏现实性。

3.2. 蒋诗昌

3.2.1. 主要功能

封面制作，打开地图后云朵动画，动画路径生成函数以及相关文件

3.2.2. 实现细节

1. 封面制作：
   - 图片加载优先从资源系统加载封面图（:/image_database/cover.jpeg），失败时生成灰色背景 + 居中文字” 北大校园” 的默认图
   - 核心交互按钮”Enter PKU” 按钮（200x60px）三态样式：半透明红（默认）→ 高亮红（悬停）→ 深红（按下）通过点击触发 startNavigation 信号
   - 垂直布局 + 弹性空间（顶部 4: 底部 1）使按钮悬浮于底部，图片始终铺满窗口（paintEvent 实现）
2. 地图封面切换虚化效果：
   - QLabel 初始化创建左右云朵，形成半透明背景，初始位置居中折叠
   - 延迟 200ms 启动动画组（QParallelAnimationGroup）包含 4 个子动画：
     - 左云向左平移（3 秒，缓动曲线 OutCubic）
     - 右云向右平移（3 秒，缓动曲线 OutCubic）
     - 双云淡出（1.8 秒，透明度 1→0）
3. 动画路径生成函数以及相关文件：一共实现 5 种路经——矩形，圆形，折线形，对角形，静止
   - 矩形路径通过起点、宽高和步长参数，按照顺时针或逆时针方向生成四条边组成的闭合路径
   - 圆形路径基于圆心和半径，使用三角函数均匀分布指定数量的点形成平滑圆周
   - 对角线路径在起点和终点之间线性插值计算往返路径点
   - 折线路径则连接起点、多个途径点和终点，支持闭合选项形成多边形
   - 静态路径生成固定位置的点，可选持续时间参数实现原地停留效果
   - 所有路径均通过 JSON 配置参数驱动，返回 QPoint 列表，支持自定义步长、方向等参数，适用于各类动态轨迹需求

3.2.3. 亮点与难点

亮点：

1. 通过调用读取 json 文件，使得路径生成函数简洁，利于代码的模块化结构化，方便扩展与阅读。
2. 自适应窗口大小的云朵缩放、自动资源回收机制、顶层渲染保证，以及通过 200ms 延迟触发和复合动画组（位移 + 透明度）实现的精细化时序控制，最终形成既省内存又富有沉浸感的动态过渡效果。
3. 通过 paintEvent 动态拉伸图片至窗口尺寸，完美适配不同分辨率，行代码连接按钮点击信号到 startNavigation，实现业务逻辑零侵入。”Enter PKU” 按钮采用三态动态样式（默认半透红/悬停高亮/按压深红）。

难点：

1. 云朵动画参数调整，多次修改参数以使动画平滑自然，并对图片初始化处理时适当放大以完整覆盖屏幕，达到“拨云见日”的效果。
2. 路径生成的实现难点在于需要高效解析 JSON 配置并动态适配多种几何路径算法（矩形、圆形、折线等），同时确保不同参数组合（如步长、方向、闭合性）下的路径点连续性和精度。其中，圆形路径需处理三角函数计算的性能优化，折线路径要解决多点间插值的平滑过渡，而动态参数（如持续时间）要求路径点密度可调，最终需在保证数学准确性的前提下，输出适配动画系统的 QPoint 序列，并兼顾异常配置的鲁棒性。
3. 封面制作的实现难点在于平衡视觉稳定性与动态交互，需处理图片加载失败时的健壮性回退方案，确保不同分辨率下的自适应布局精确性，最终在有限代码量内完成从资源加载、异常处理到动态交互的全链路闭环设计。

3.3. 邓文骐

3.3.1. 主要功能

1. 组长，提出程序设计想法
2. 分配任务，鼓励组员，调动大家的积极性
3. 搭建并管理 git 仓库，撰写README，整理仓库并优化项目结构
4. 配置.gitignore config.json文件，确保 api 不被上传
5. 搭建可移动的imageview窗口和navigation界面,实现最初版的地图查看窗口
6. 拓展图标的动作、移动效果，完成图标类使其仅根据图标名称便可生成实例
7. 整理所有的数据（rc 文件夹下），逐帧调整动画效果
8. 制作图标的动图素材，提供图片实拍素材
9. 撰写本报告，录制功能介绍视频

3.3.2. 实现细节

(见报告上一节)

---

4. 项目总结与反思

4.0.1. 蒋诗勗

1. 熟练使用 github 各种指令的能力有待提升，负责不同模块的同学之间的协作沟通，代码的共享上传很大程度上决定了项目的完成效率。
2. 代码的可读性：函数的封装，变量命名，整洁关键的注释，这些都可以为你的 cooperator 带来方便。
3. 过度追求技术完美反而容易陷入细节泥潭，比如最初花费大量时间调试云朵动画的缓动曲线，实际上整体观感更在意整体流畅感而非具体参数。

4.0.2. 罗以楠

1. 合理的使用 AI 很大程度上加快了我们上手 Qt 和构建一个基础框架的速度。
2. 在撰写过程中要考虑到小组合作，对函数和变量进行合理的命名。
3. 要兼顾开源和隐私性的问题，在 API 等方面要深思安全问题。
4. 要充分考虑程序的可扩展性和可移植性，尽量让一个类能多次利用，减少代码量。

4.0.3. 邓文骐

1. 用心交流
   - 项目协作的核心在于充分沟通
   - 需要坚持耐心细致地阐述设计思路，不厌其烦地解释预期效果和核心想法
   - 还需要确保团队成员对需求理解一致
2. 渐进式开发
   - 初期避免追求过于复杂的效果，从最小可行功能 demo 开始构建
   - 优先实现核心功能，把只需要花时间一定可以优化的效果往后放——先完成，再完美
   - 在稳定基础上逐步完善细节
3. 效率优化
   - 重复的图片名称修改可以使用脚本
   - 将重复的信号槽连接重构为遍历实现
   - 开发者应当为所有重复性工作寻求自动化解决方案