数据结构 PJ-OpenStreetMap 实验报告

洪成勋 22300240021

一、 概述

1. 项目语言、环境及工具：C++，Visual Studio 2022，pugixml 库，easyx 库；
2. 代码量：1 个.h 文件，8 个.cpp 文件，约 1300 行。

二、 基本功能完成情况

1. 从 OSM 网页下载了成都市主市区（三环路及以内，70000+个点）的地图数据，并构造出了一张地图。使用 C++的 pugixml 库读取数据，easyx 库进行地图绘制，完成了必要的 UI 和程序运行效率的改善；
2. 实现了地图查看的一些基本操作，包括用键盘控制地图视角上、下、左、右移动，放大/缩小，以及用鼠标滚轮控制放大/缩小的功能；
3. 完成了求两点之间最短路径的 Dijkstra 算法和 A*算法，通过（可选功能项）在图中绘制出算法过程中所遍历过的点，以及寻路的过程来比较了二者之间的效率差异，并在地图上展示出最短路径。

三、 扩展功能完成情况

1. 加入了启动界面的动画；
2. 增加了右上角小地图（缩略图及视图展示）；
3. 增加了左下角比例尺显示和模式显示；
4. 增加了全新的地图模式——末日模式（Doomsday Mode），并结合 Welsh-Powell 算法与地图五色定理实现了平面（地）图的面着色贪心算法（该部分会在后文详细介绍）；
5. 增加了“帮助”功能，可在控制台窗口打印地图使用说明及个人 logo。

四、 项目各功能展示（截图）及使用说明

1. 启动动画（动效）

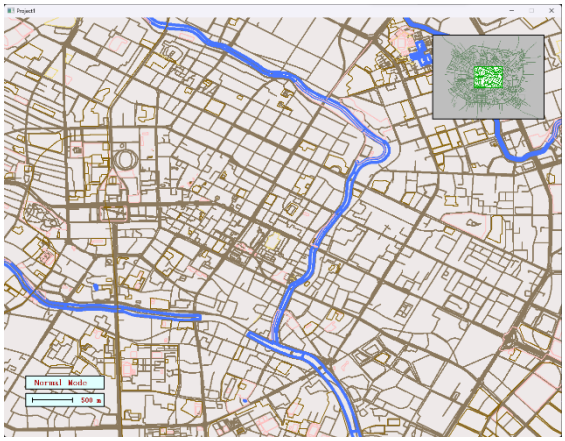
*立体星空横移，字体渐显，按任意键继续

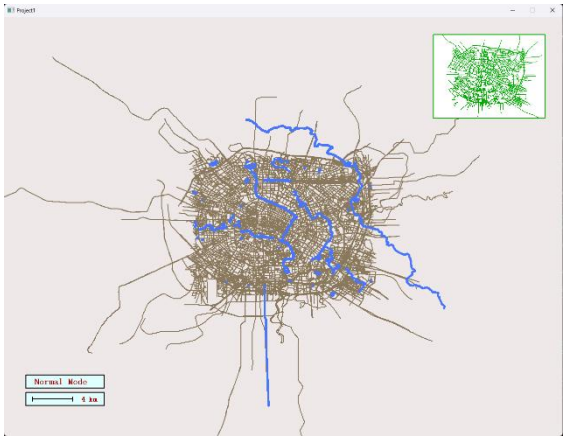
2. 地图绘制与视角变换

*正常模式（左）以及末日模式（右）

左下角显示比例尺，右上角显示小地图。蓝色为河流，棕色为城市道路。

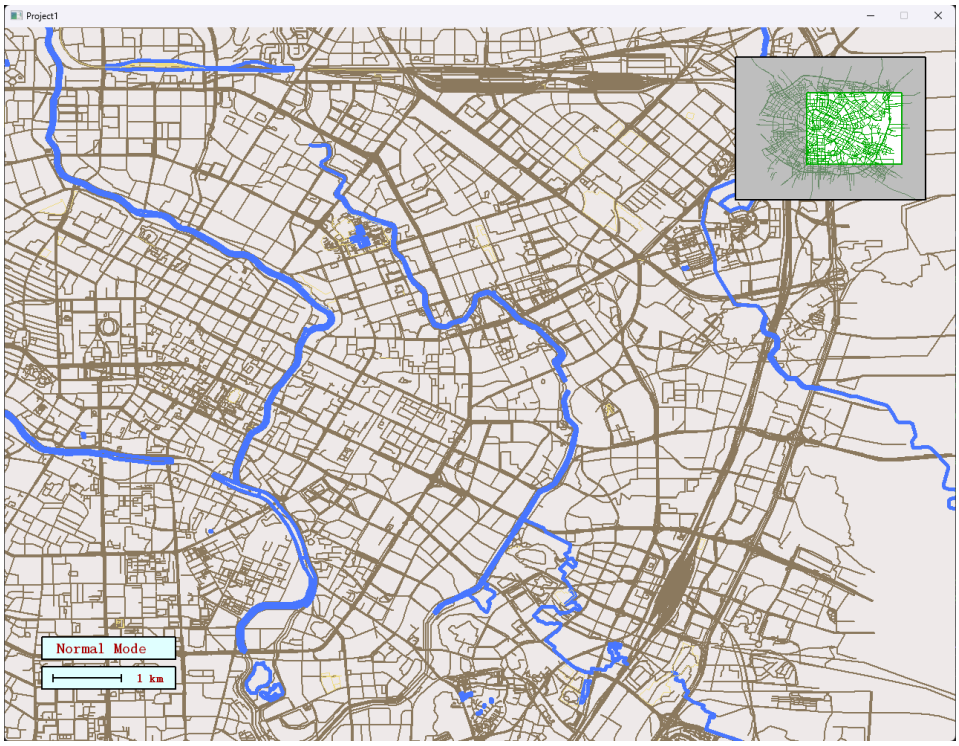
键盘 Q 键/鼠标上滚轮：视角放大（以下功能以正常模式演示），键盘 E 键/鼠标下滚轮：视角缩小。比例尺随之改变，小地图显示可视区域，绘制线条粗细、地图细节随之改变（如放大到一定倍数开始出现建筑、公共基础设施等细节）





*放大（左）、缩小（右）后的地图视角

键盘 W/↑、S/↓、A/←、D/→键：控制视角上下左右移动。



*放大、平移后的地图视角

键盘 Esc 键：恢复原始视角。

键盘 Tab 键：切换模式（正常模式/末日模式）。

3. 算法展示

A. Dijkstra 算法

正常模式下，鼠标左键单击以在地图上选择 Dijkstra 算法的起点和终点，选择完毕后会绘制出两点间的最短路径。

键盘 Ctrl 键：切换展示模式（标记出算法遍历过的节点，以及算法的寻路过

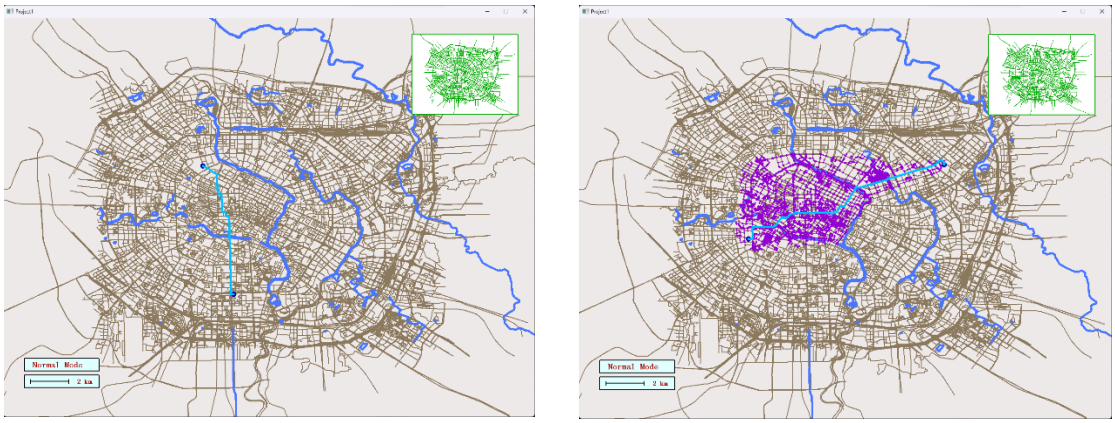
程，通过前驱表+Sleep 来实现，适用于正常模式的 Dijkstra 和 A*算法）。

键盘 Space 键：清除算法已绘制的路线和点。

*非展示模式（左）及展示模式（右）下的 Dijkstra 算法

B. A*算法

正常模式下，鼠标右键单击以在地图上选择 A*算法的起点和终点，选择完毕后会绘制出两点间的最短路径。



*非展示模式（左）及展示模式（右）下的 A*算法

C. Welsh-Powell 贪心算法

该部分会在后续的拓展功能介绍中进行讲解。

4. *末日模式（Doomsday Mode）

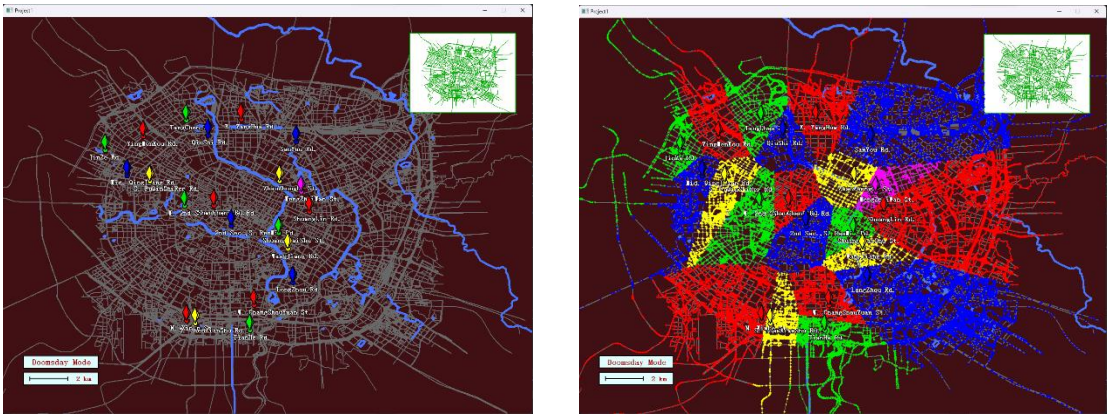
这是我自己添加的一个模式。该模式的背景是，由于在我的家乡成都，地震等地质灾害频发，城市中设立了很多公共紧急避难场所。于是我在 Google Map 及 Tencent Map 的网站上下载了一些避难场所（Shelters）的位置、名称等信息，并融合进了我的地图之中。通过不同的配色、地图各点与它们的位置关系以及结合 Welsh-Powell 算法对不同区域进行贪心的面着色，来进行最近避难所推荐，同时也验证了著名的“五色定理”的正确性。

```
vector<Shelter_t> shelters = { Shelter_t(30.6072,104.0481,"S. ShenXianShu Rd."),      Shelter_t(30.6083,104.0440,"N. XinLe St."),       Shelter_t(30.6956,104.0236,"YingMenKou Rd."),
                               Shelter_t(30.7033,104.0438,"TangChen Rd."),            Shelter_t(30.7036,104.0699,"E. YangHua Rd."),     Shelter_t(30.6962,104.0541,"QiuShi Rd."),
                               Shelter_t(30.6930,104.0960,"SanYou Rd."),              Shelter_t(30.6745,104.0881,"ZhaoZhongCi St."),    Shelter_t(30.6695,104.0984,"MengZhuiWan St."),
                               Shelter_t(30.6627,104.0431,"W. 2nd Sec.,1st Ring Rd."),Shelter_t(30.6630,104.0569,"ShaoCheng Rd."),     Shelter_t(30.6600,104.1022,"ShuangLin Rd."),
                               Shelter_t(30.6532,104.0651,"2nd Sec.,S. RenMin Rd."),  Shelter_t(30.6497,104.0878,"ShuangHuaiShu St."), Shelter_t(30.6157,104.0758,"W. ChangShouYuan St."),
                               Shelter_t(30.6891,104.0056,"JinZe Rd."),               Shelter_t(30.6773,104.0159,"Mid. QingJiang Rd."), Shelter_t(30.6743,104.0266,"S. FuQinShiRen Rd."),
                               Shelter_t(30.6039,104.0739,"TianHe Rd."),              Shelter_t(30.6264,104.0945,"LongZhou Rd."),       Shelter_t(30.6423,104.0921,"WangJiang Rd.") };//Shelters' location
```

*一些避难所经纬度、名称信息，源自 Google Map/Tencent Map

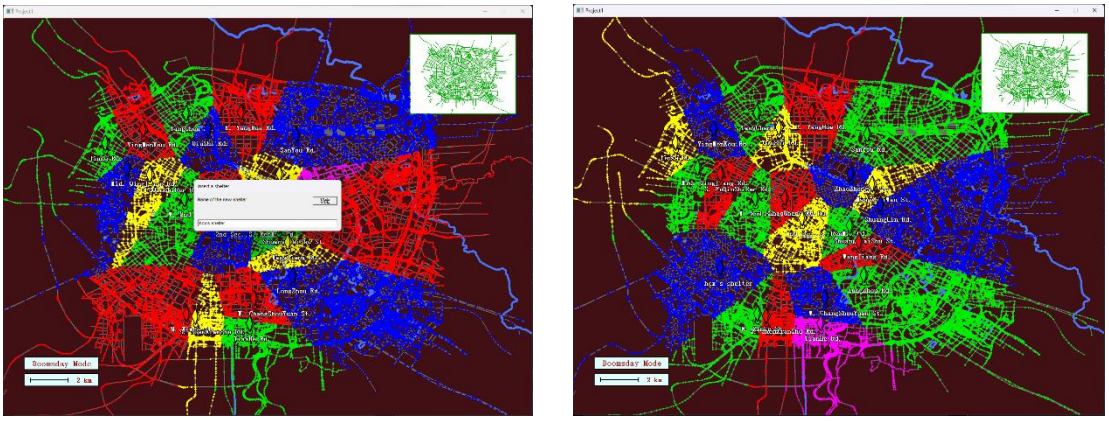
同时，我还增加了自设避难所的功能，用户可以在地图上新建自己的避难所信息，并为其取名。随后算法会根据新的避难所位置重新对着色方案进行规划，详见后文演示。

键盘 P 键：在地图上显示已有的避难所位置与名称信息。
键盘 O 键：对地图上区域进行分割，使每个地图节点与距其最近的避难所着同色。



*已有避难所信息（左）及地图分割的着色展示（右）

鼠标左键：在地图上点击添加自己的避难所，此时会弹出一个输入框，可输入避难所的名字。随后地图上会增加一个自建的避难所图标，且着色方案随之改变。（这样的避难所目前可以无限添加，但不排除添加达到一定数量后，对它们着色所用的颜色数大于 5 个）

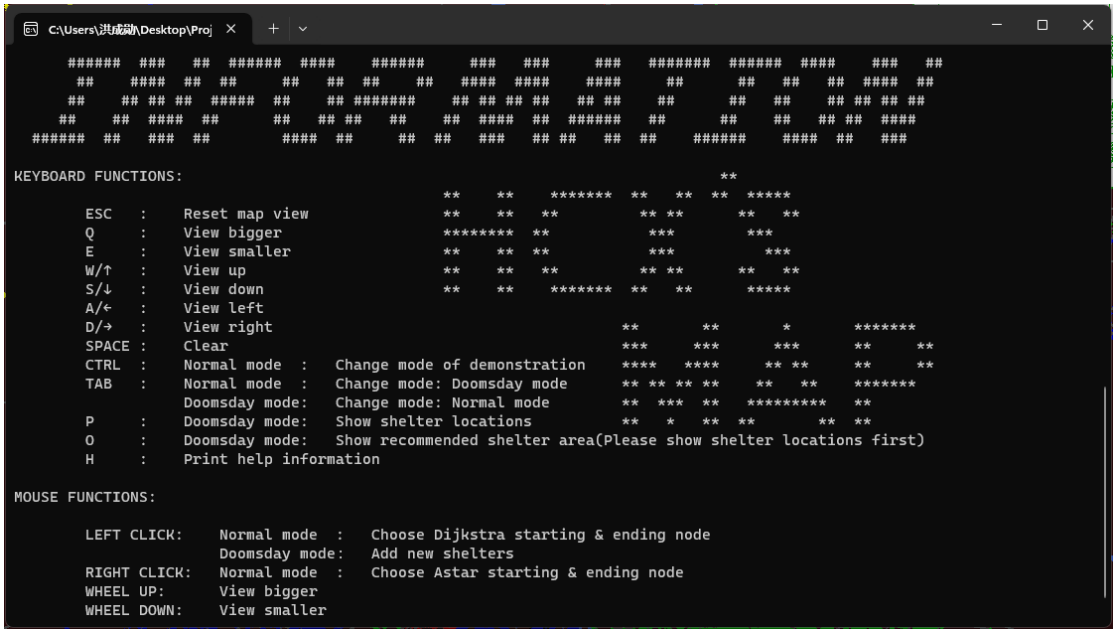


*自定义新建避难所名称（左）及更新后的地图着色（右）

算法思路：
在我的算法实现中，首先计算各点到各避难所的距离（这里最终使用的是 Euclid 距离；Manhattan 距离会更快但精度降低严重，而实际路程距离精度最高但耗时显著提高），并标记出每个点距离最近的避难所（及其颜色标号）。随后利用邻点关系和对偶图的性质将地图的面着色问题转化为其对偶图的点着色。然后利用 Welsh-Powell 算法，核心思想就是将颜色进行标号（下标 0~4），从度数最高的点出发，并每次寻找其邻点中度数最高的点，贪心地选取标号尽量小的颜色对其进行着色。如此便可使所用的颜色数尽量少且相邻的面不会出现同色的情况。

5. 打印帮助信息

键盘 H 键：可在控制台窗口打印操作手册以及个人 logo。



*地图操作指引

五、 其它亮点分析

1. 启动动画

```
ExMessage* ms = NULL;//
TCHAR text[40] = _T("Hello! Welcome to hcx's city!!");//
int incr = 255 / 160;//
int cnt = 0;//
LOGFONT f;//
gettextstyle(&f);//
f.lfHeight = 80;//
_tcscpy_s(f.lfFaceName, _T("Comic Sans MS"));//
f.lfQuality = ANTIALIASED_QUALITY;//
f.lfWeight = FW_ULTRABOLD;//
settextstyle(&f);//
int text_width = WINDOW_WIDTH * 0.618;//Loading scene

while (1)
{
    if (peekmessage(ms, EX_KEY, true)) break;
    if (cnt < 160)
    {
        cnt++;
        settextcolor(RGB(incr * cnt, incr * cnt, incr * cnt));
    }
    outtextxy((WINDOW_WIDTH - text_width) / 2, WINDOW_HEIGHT / 2, text);//
    for (int i = 0; i < MAXsSTAR; ++i) Movesstar(i);//
    for (int i = 0; i < MAXbSTAR; ++i) Movebstar(i);//Text & Stars
    Sleep(20);
}
cleardevice();
int r_incr = GetRValue(COLOR_BKGROUND_NORMAL) / 80;
int g_incr = GetGValue(COLOR_BKGROUND_NORMAL) / 80;
int b_incr = GetBValue(COLOR_BKGROUND_NORMAL) / 80;
settextcolor(COLOR_BKGROUND_NORMAL);
outtextxy((WINDOW_WIDTH - text_width) / 2, WINDOW_HEIGHT / 2, text);
int r = 0, g = 0, b = 0;
BeginBatchDraw();
while (1)
{
    r += r_incr;
    g += g_incr;
    b += b_incr;
    if (r ≥ GetRValue(COLOR_BKGROUND_NORMAL) || g ≥ GetGValue(COLOR_BKGROUND_NORMAL)) break;
    setbkcolor(RGB(r, g, b));
    cleardevice();
    outtextxy((WINDOW_WIDTH - text_width) / 2, WINDOW_HEIGHT / 2, text);
    FlushBatchDraw();
    Sleep(8);
}
```

*利用 RGB 色彩参数的微调以及 easyx 库提供的 BeginBatchDraw，FlushBatchDraw 和 EndBatchDraw 函数来实现了字体的渐显效果、画面的融合。利用随机数种子和状态机实现星空映像。

2. 数据处理

```cpp
int Euclid(ll start_id, ll end_id)//Calculate Euclid distance
{
    return pow(nodesinsight[start_id].x - nodesinsight[end_id].x, 2) + pow(nodesinsight[start_id].y - nodesinsight[end_id].y, 2);
}

int Manhattan(ll start_id, ll end_id)//Calculate Manhattan distance
{
    return abs(nodesinsight[start_id].x - nodesinsight[end_id].x) + abs(nodesinsight[start_id].y - nodesinsight[end_id].y);
}

void Closest(int n)//Find closest node according to the click point
{
    int mindis = 10000000;
    int dis;
    for (auto& pair : nodesinsight)//Find closest node using Manhattan distance
    {
        if (graph[pair.first].empty()) continue;
        if (n == 1) dis = abs(m_x1 - pair.second.x) + abs(m_y1 - pair.second.y);
        else dis = abs(m_x2 - pair.second.x) + abs(m_y2 - pair.second.y);
        if (dis < mindis)
        {
            mindis = dis;
            if (n == 1) n_1 = pair.second;
            else n_2 = pair.second;
        }
    }
}
```

*提供了不同的距离计算函数来针对不同功能作距离计算

3. 小地图

```cpp
void Overview()//Draw overview
{
    int ov_mid_x = (ov_left + ov_right) / 2;//
    int ov_mid_y = (ov_top + ov_bottom) / 2;//Overview central coordinate
    int area_width = ov_width * 4 * ratio;//
    int area_height = ov_height * 4 * ratio;//Overview visible area central coordinate

    int left = area_mid_x - area_width / 2;//
    int top = area_mid_y - area_height / 2;//
    int right = min(left + area_width, ov_right);//
    int bottom = min(top + area_height, ov_bottom);//
    left = max(left, ov_left);//
    top = max(top, ov_top);//Overview visible area edge

    setfillcolor(COLOR_OVERVIEW_BACKGROUND);//
    solidrectangle(984, 40, 1240, 232);//Background(dark)
    setfillcolor(WHITE);//
    solidrectangle(left, top, right, bottom);//Background(light)
    setlinestyle(PS_SOLID, 1);
    for (const auto& way : ways_highway)//Draw only highway
    {
        if (way.stype != "trunk" && way.stype != "primary" && way.stype != "secondary" && way.stype != "tertiary") continue;

        int ox1 = -1, oy1 = -1, ox2 = -1, oy2 = -1;//Nodes on overview
        string os1, os2;
        for (size_t i = 0; i < way.nodeRefs.size() - 1; i++)
        {
            ox1 = ov_mid_x + static_cast<int>((nodes[way.nodeRefs[i]].lon - 104.0824) * 256 / 0.25);//
            oy1 = ov_mid_y + static_cast<int>((30.6582 - nodes[way.nodeRefs[i]].lat) * 256 / 0.25);//
            ox2 = ov_mid_x + static_cast<int>((nodes[way.nodeRefs[i + 1]].lon - 104.0824) * 256 / 0.25);//
            oy2 = ov_mid_y + static_cast<int>((30.6582 - nodes[way.nodeRefs[i + 1]].lat) * 256 / 0.25);//Calculate coordinates on overview

            if (ox1 <= ov_left || ox1 >= ov_right || oy1 <= ov_top || oy1 >= ov_bottom || ox2 <= ov_left || ox2 >= ov_right || oy2 <= ov_top || oy2 >= ov_bottom) continue;//
            if (ox1 >= left && ox1 <= right && oy1 >= top && oy1 <= bottom || ox2 >= left && ox2 <= right && oy2 >= top && oy2 <= bottom) setlinecolor(GREEN);//Visible line color
            else setlinecolor(RGB(105, 139, 105));//Invisible line color
            line(ox1, oy1, ox2, oy2);//Draw the line
        }
    }
    setlinestyle(PS_SOLID, 2);//
    setlinecolor(BLACK);//
    rectangle(ov_left, ov_top, ov_right, ov_bottom);//Black frame
    setlinecolor(GREEN);//
    rectangle(left, top, right, bottom);//Green frame of visible area
    setlinestyle(PS_SOLID, linewidth);//Reset linestyle
}
```

*对坐标进行了重新计算，并用白色和灰色作背景，制造出可视区域亮、不可视区域暗的实时效果，并用绿框来增强提示

4. 详细的注释

*代码中插入了大量详细的注释（由于发现用 VSCode 打开时中文注释会变成乱码，所以整个Project都采用了英文注释,其中可能出现的语法错误和指代不明敬请通融）

5. 面向对象特性

```cpp
#include "PJ.h"

int main()
{
    Init_window();//Window initialization

    Loadmap();//Load data

    Drawmap();//Draw the map

    Init_graph();//Initialize graph

    Multifunc();//Function changing
    return 0;
}
```

*主函数十分精简，在整个 Project 中实现了很好的函数和数据的封装性、程序的模块化，体现了面向对象的特性

六、 实验过程记录

1. 12 月 13 日，利用从 OSM 下载的数据和 easyx 库实现了 UI 并成功绘制出地图；
2. 12 月 15 日，优化了地图的配色、线条等整体美观度，添加了视角变换功能；
3. 12 月 17 日，实现了 Dijkstra 算法；
4. 12 月 19 日，实现了 A*算法，增加了演示模式；
5. 12 月 20 日，增加了小地图和比例尺功能；
6. 12 月 22 日，实现了帮助信息打印；
7. 12 月 23 日，优化了邻接表的建边逻辑，数据读入方式以及存储结构；
8. 12 月 25 日，实现了着色 Welsh-Powell 算法。将代码文件从一个冗长的.cpp 文件分成了 1 个.h 文件和 8 个.cpp 文件，并整理了宏、全局变量定义和函数声明；
9. 12 月 26 日，实现了避难所的自定义，完善了 Doomsday Mode 并增加了模式显示；
10. 12 月 28 日，实现了启动动画界面。为代码添加了英文注释；
11. 1 月 1 日，再次优化地图配色和启动动画外观；
12. 1 月 3 日，实验报告的攥写。

七、 实验中遇到的困难及解决

1. 寻路算法出 bug 概率太高，出现卡顿、闪退或死机
   解决方案：增加跳出和刷新机制，避免 bug 出现时程序死循环和无法恢复的情况；优化 xml 数据读取逻辑、邻接表建边逻辑、前驱表输入逻辑。目前 bug 概率显著降低，且可以手动恢复。
2. 绘图速度太慢
   解决方法：增加地图哈希，将已绘制过的地图视角存入 unordered_map 中，在再次加载时直接 putimage；优化节点经纬度转化为坐标的计算方式，减少线条绘制量（在不同缩放倍率下绘制不同数量种类的地图信息）；将不同色数据分类绘制，减少大循环中重新 setlinecolor 的次数。
3. 键盘无法在控制台窗口最小化的情况下进行操控
   解决方法：将 kbhit()函数更换为 getmessage()函数，并置于主函数末尾用 while(true)来实时监视键盘/鼠标操作。
4. 小地图窗口、比例尺窗口等的位置、尺寸信息在优化时需多次更改，十分麻烦
   解决方法：通过宏定义来替换代码中冗余、重复的数据项，从而达到简化代码、优化可读性和批量修改的目的。

八、 项目运行方法及使用说明

详见前文讲解、帮助信息打印与代码文件中的注释。

九、 实验总结

本次 PJ 极大提升了我对代码的编写、组织、分析和优化能力，加深了我对 C++库、算法、各开发环境的认识和理解。同时，在完成过程中助教老师们给予了我莫大的帮助和鼓励，在此也想感谢各位助教老师的及时、准确的答疑。

十、 参考资料

[1]EasyX 文档-基本说明，https://docs.easyx.cn/zh-cn/intro

[2]虚拟键码对照表与 ASCII 对照表的整理，
https://blog.csdn.net/andrew_wx/article/details/6864504

[3]OSM 地图道路分类信息，http://udu.org.cn/post/8.html

[4]RGB 颜色查询对照表，https://www.sojson.com/rgb.html

[5]Visual Studio 环境安装配置教程，
https://zhuanlan.zhihu.com/p/71110525