作业报告

（一）程序功能介绍

PKU 地图助手是基于 Qt 开发的一款以燕园地图为基础的辅助游客游览的程序。其中包含了以下功能：

1. 选择指定地点后出现该地点的近景图片
2. 选择两个地点后，在地图上标出最短推荐路径，并显示骑行与步行所需时间
3. 寻找距离最近的某类地点
   eg.寻找距离最近食堂、大门
4. 为游客规划精品游览路线

为完成以上功能，本组通过 UI 设计的方式为使用者提供多种交互方式，保障灵活且精准的指导。

（二）项目各模块与类设计细节

1.Node 类

```
class node
{
public:
    int x;
    int y;
    QString name;

    node();
    node(int a, int b, QString s);
    QString findNearestDining();
    QString findNearestGate();
};
```

整个程序通过对 node 类的不同调用实现不同功能，其中 node 类包含：

- 类成员：
  x,y（int） 记录不同地点在显示框中的横纵坐标
  Name（string） 记录不同地点的名称
- 成员函数：
  QString findNearestDining（） 判断距离当前地点最近的食堂
  QString findNearestGate（） 判断距离当前地点最近的大门

```
QString node::findNearestDining()
{
    int timeNong = abs(this->x - NongYuan.x) + abs(this->y - NongYuan.y);
    int timeShao = abs(this->x - ShaoYuan.x) + abs(this->y - ShaoYuan.y);
    int timeJia = abs(this->x - JiaYuan.x) + abs(this->y - JiaYuan.y);
    if (timeNong < timeShao && timeNong < timeJia)
    {
        return "农园";
    }
    else if (timeJia < timeShao && timeJia < timeNong)
    {
        return "家园";
    }
    else
    {
        return "勺园";
    }
}
```

为地图上多个地点创建了 node 类的对象 并使用映射 map 实现了地点名字 qstring 到对应对象 node 的一一对应 方便后续匹配

2. 其余函数

- Qstring CalculateTime（）

```
QString caculateBikingTime(node place1, node place2)
{
    int distance = abs(place1.x - place2.x) + abs(place1.y - place2.y);
    double time = (double)distance / 210 ;
    QString stime = QString::number( time, 'f', 1) + "(分钟)";
    return stime;
}
```

主要分为计算单车时间和步行时间两个函数。在这两个函数中我们分别估计了单车和步行的平均速度，并使用曼哈顿距离作为对两点间距离的估计方式

- Void MainWindow：： showViewImage（）

```
void MainWindow::showViewImage()
{
    QString viewText = ui->ViewcomboBox->currentText();

    if ( viewText == "无"){}

    else if ( viewText == "未名湖")
    {
        QWidget w;
        w.setWindowTitle("未名湖");
        w.setStyleSheet("QWidget { border-image: url(:/new/prefix1/WeimingLake); }");
        w.resize(800,1000);
        w.show();
        QEventLoop loop;
        loop.exec();
    }
```

该函数用于实现查看近景功能。我们对每个地点都是用类似框架，重新打开一个窗口显示该地点近景，并在点击取消键时终止新窗口

- Void MainWindow：： showRoutine（）

```
void MainWindow::showRoutine()
{
    QString routineText = ui->TravelcomboBox->currentText();
    if (routineText == "精华路线（1h）")
    {
        QWidget w;
        w.setWindowTitle("精华路线");
        w.setStyleSheet("QWidget { border-image: url(:/new/prefix1/EssenceRoutine); }");
        w.resize(1200,1400);
        w.show();
        QEventLoop loop;
        loop.exec();
    }
```

该函数用于实现查看精品规划功能。具体实现方式与查看近景类似。

- Void MainWindow：： paintEvent（QPaintEvent* event）

该函数用于实现在输入指定地点后在地图上画出路线的功能。在此函数中，划线功能由 qt 中的 painter.drawline 实现。同时调用了先前已经实现的 calculateTime 的功能，这样就可以在划线的同时同时显示最有规划路径的预计通行时间。

```cpp
void MainWindow::paintEvent(QPaintEvent* event)                    ⚠unused parameter
{
    QPixmap pixmap(":/new/prefix1/PKU_map.png");
    QPainter painter(this);
    painter.drawPixmap(0,0,pixmap);    //绘制地图

    QPen pen(Qt::red,6);
    painter.setPen(pen);
    //painter.drawLine(0,0, 700, 1130);
    QString startText = ui->GocomboBox->currentText();       //获取起始地
    QString endText = ui->TerminalcomboBox->currentText();   //获取目的地

    if (startText == "无" || endText == "无"){ return; }

    painter.setFont(QFont("Arial",20));
    painter.drawText(places[startText].x, places[startText].y, places[startText].name);    //在地图上标明起始地
    painter.drawText(places[endText].x, places[endText].y, places[endText].name);          //在地图上标明目的地

    ui->WalkTimeEdit->setText(caculateWalkingTime(places[startText], places[endText]));    //显示步行时间
    ui->BikeTimeEdit->setText(caculateBikingTime(places[startText], places[endText]));     //显示骑行时间
```

（三）小组成员分工情况

本小组在前期共同讨论程序实现框架与细节，后主要由王宣棠设计 ui 面板，杨浩辅助邹宜轩完成代码编写部分。最后由王宣棠撰写作业报告，邹宜轩录制程序演示视频，杨浩负责视频后期。

（四）项目总结与反思

在该次项目合作中，我们体会并实践了实现整个程序的全部过程。从一开始的构想，到最后细节的落地、成品的呈现，这其中不可谓没有挑战与摸索。有先前计划的功能（如放大缩小地图）最后无法实现，也有部分功能完全改变了实现和呈现的方式。在这一次实践和学习的宝贵经验中，本组反思自我、自省成果，看到了不足也看到了成功，在今后的学习中，诚以为要更脚踏实地，夯实基础，才能在程序设计的实践中自由灵活的实现各样的功能，更好的进步。感谢助教一个学期的努力，收获颇丰。