CPS 3601 PROJECT #1 REPORT

iTer: An Interactive Map based on 3 input approaches

WENZHOU-KEAN UNIVERSITY

2021-11-22

WENZHOU-KEAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Team Members:

Qiu Yang (David)

Wang Xiao (Maverick)

Xu Yuxuan (Jacky)

Xu Wenhe (Eric)

Zhu Yunfeng (Mark)

Contents

Abstract

本项目采用组合的概念，探索性地组合了三种不同的输入法来实现一个交互式应用程序。用户可以通过触摸屏（平板电脑/手机）、连接到电子设备的地球仪或语音（国家或地区名称）来选择某个国家/地区。输出将是相关多媒体信息（照片、热图等）的可视化。首先关于第一个输入方式，是现实中用户可以直接触摸的显示设备，即“控制面板”。开发团队使用 Arduino 将原本复杂的显示机制简化为可通过可操作按钮直接点击的触控。在可以全面显示所有国家的地球版本中，开发团队以 Unity 3D 为主要开发引擎，编写了一系列 C#脚本来帮助软件运行。我们尽可能完成了项目要求，包括路线展示和互动屏幕。由于开发时间有限，我们取消了原先将两款产品整合为一个客户端的计划。取而代之的是，我们将第二个能够语音识别的系统（使用虚幻 4 引擎制作）做成了无障碍版本，增加了按键的语音提示和跳转后的语音提示，让有一定识别困难的用户也可以随意体验这款产品。在最终展示环节，用户可以通过直接操作装有 Windows 系统的可触控设备、语音输入、以及使用 Arduino 面板来与软件互动，实现包括：国家旅行路线，信息展示，地图绘制等功能。

This project adopted the concept of combination, exploratoryly combining three different input methods to develop an interactive application. The user can select a country/region through a touch screen (tablet/mobile phone), a globe connected to an electronic device, or voice (country or region name). The output will be a visualization of the relevant multimedia information (photos, heat maps, etc.). The first method as input is the display device that can be directly touched by the user in reality, that is, the "control panel". The development team uses Arduino to simplify the originally complex display mechanism into a touch that can be directly clicked through an operable button. In the version of the globe that can fully display all countries, the development team uses Unity 3D as the main development engine and wrote a series of C# scripts to help the software run. We have achieved as complete the project requirements as possible, including route display and interactive screens. Since development time is still not enough, we canceled the original plan to integrate the two products into one client. Instead, we made the second system capable of voice recognition (made with the Unreal 4 engine) an accessible version, adding voice announcements of buttons and voice prompts after jumps, so that users with certain recognition difficulties can also Feel free to experience this product. In the final presentation, users can interact with the software by directly operating touch-enabled devices equipped with Windows system, voice input, and using the Arduino panel to achieve functions including: national travel routes, information display, and map drawing.

Overall architecture

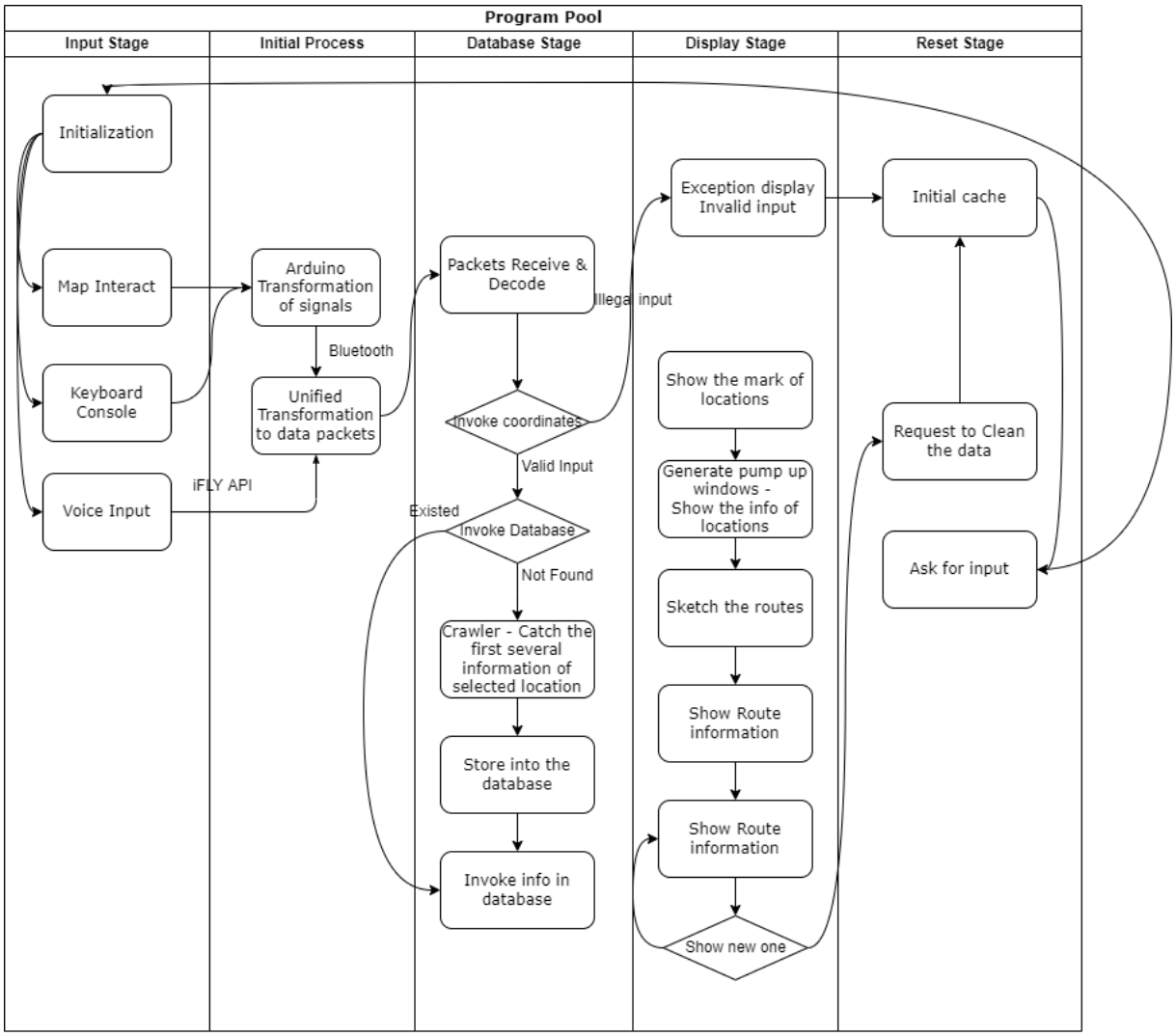


Figure.1 Initial Flowchart

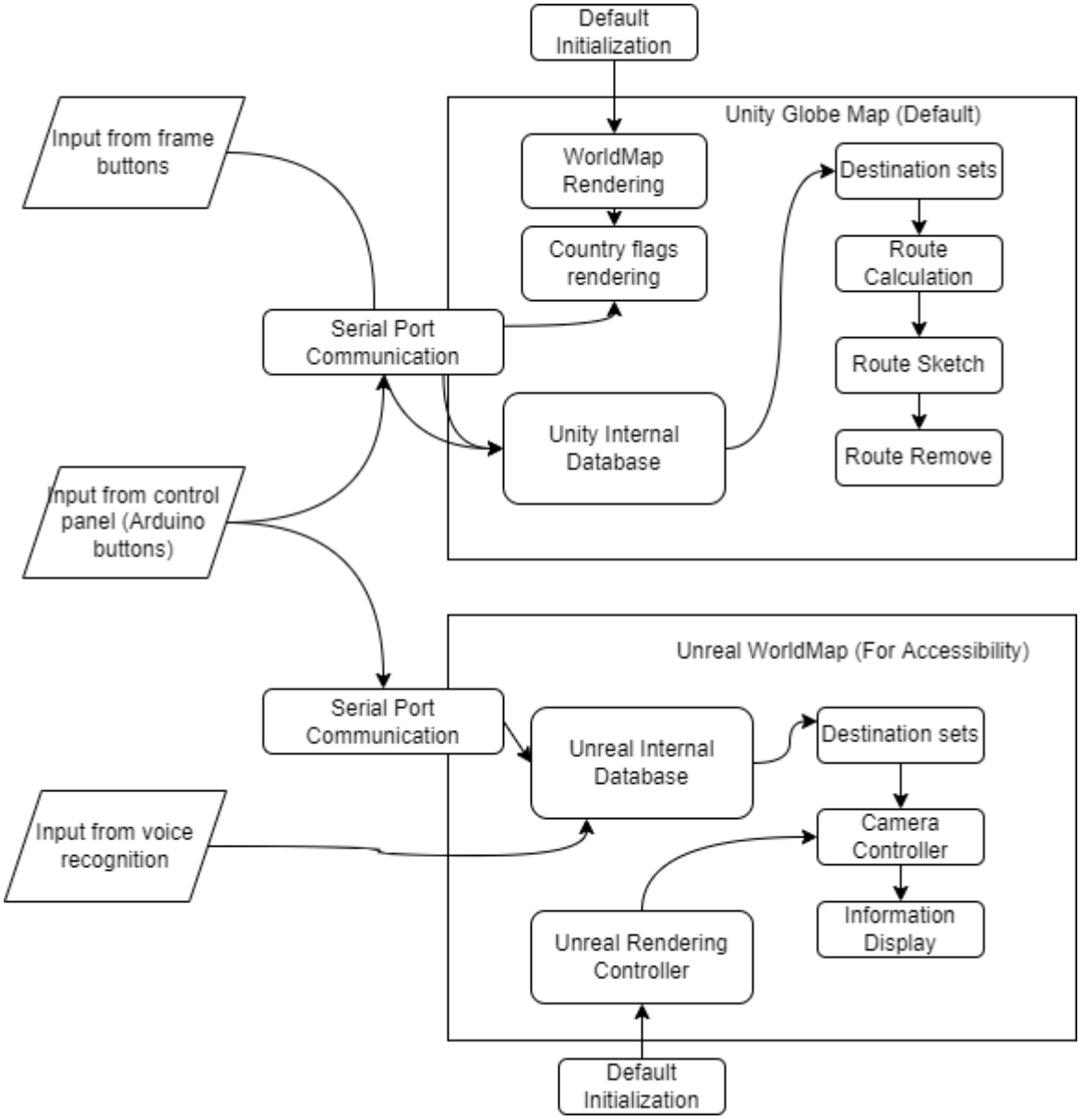


Figure.2 Revised Architecture Chart

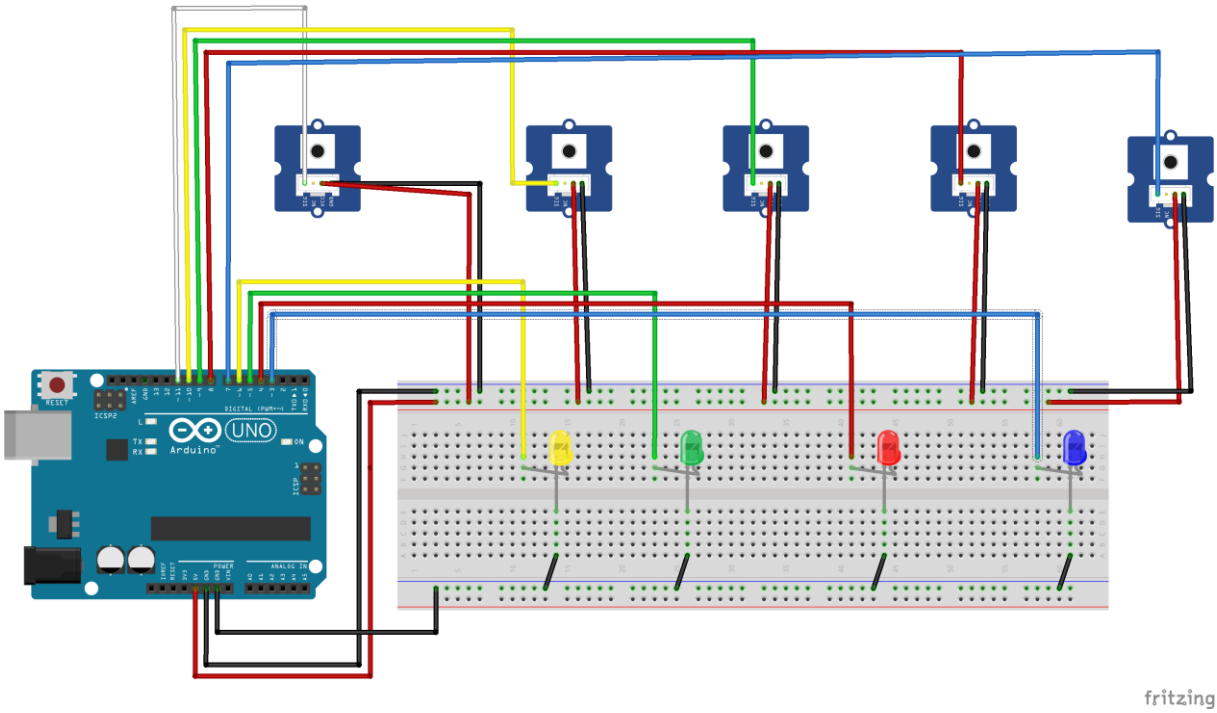
Hardware devices and implementation

The whole harware part is based on Arduino board and related modules, implement country/city selection, footprint display, signal transmission, and serial communication functions.

1. Hardware

- Arduino Board UNO R3 x1
- Breadboard x1
- 2.54mm Dupont Wires (Male to Male, Female to Female, Male to Female) xN
- Push Button xN
- LED diode light xN
- World Map x1

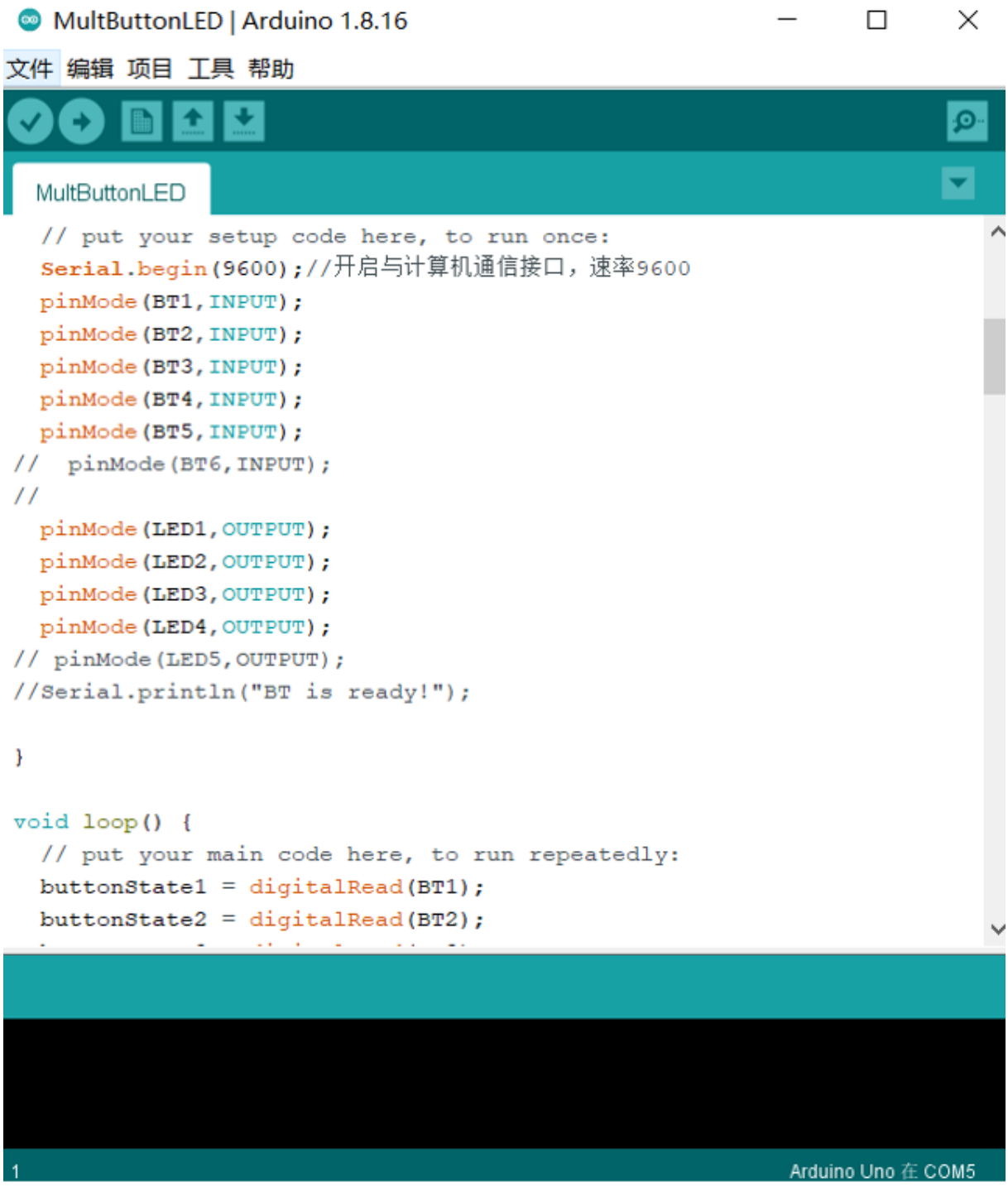
2. Circuit



Connect N wires to the board. Connect the two long vertical rows on the side of the breadboard with the red and black wires to provide access to the 5 volt supply and ground. The other color wire represents the pin port link for signal input and output. Connect the pushbutton and led light to the circuit in turn.

When you press the pushbutton, the corresponding led light will light up to display the footprints, and the signal will be transmitted through the Arduino board and serial port (COM5) to complete the human-computer interaction.

3. Code



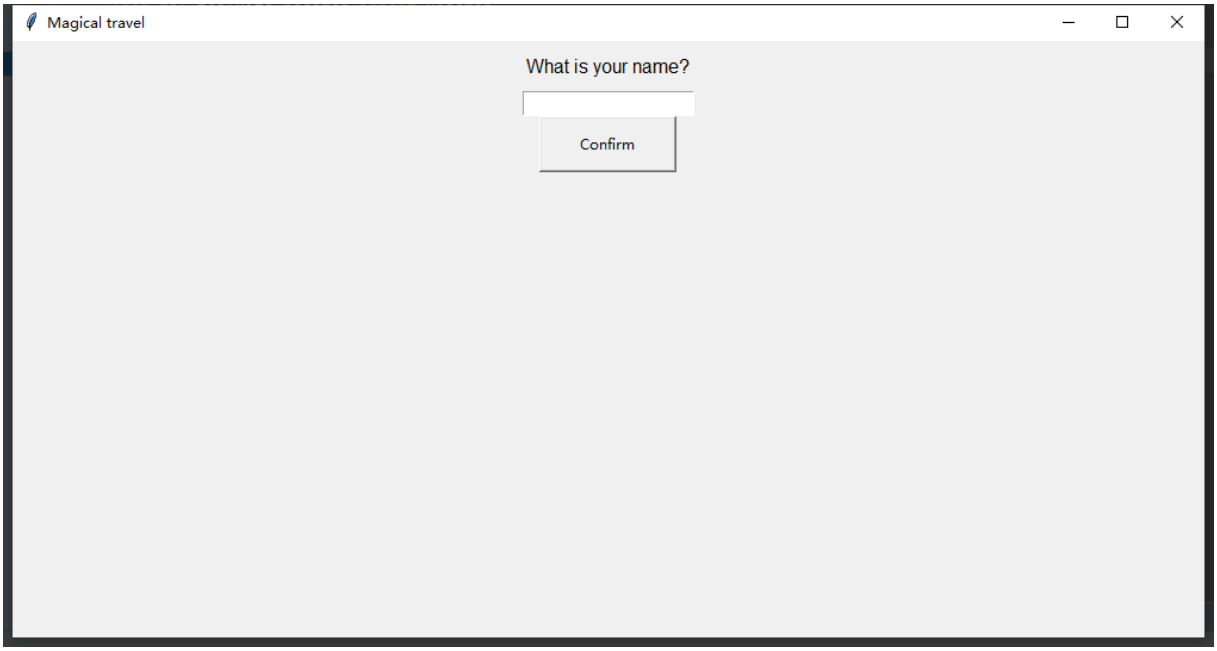
The main code is to use C# and write in the Arduino IDE, in the SetUP(), we set which Pin is used in input or output, and connect each pushbutton with the attached LED light.

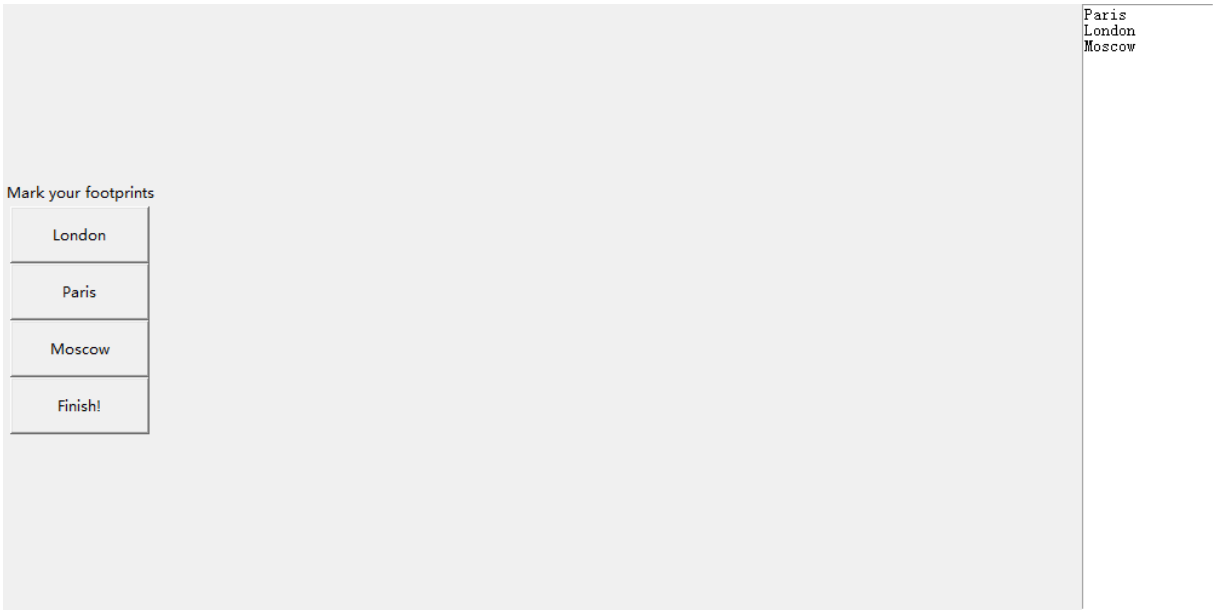
Then, start the Serial communication part. when the Arduino board connects to the PC, we get the Port name through the device manager, in the PC, we can get the data from board by Serial Monitor. Such data is necessary for our interaction. we set serial monitor events in software parts (Unity and Unreal Engine4), such signals can decide the information shown by tablet and PC.

4. Snapshot

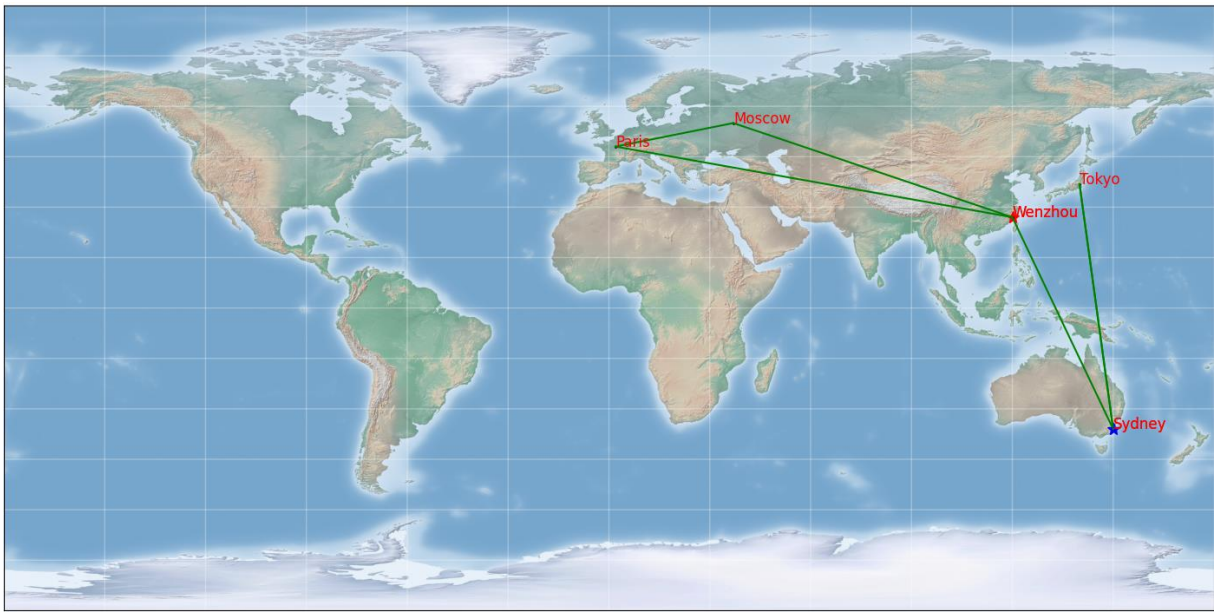
5. UX consideration

To design our user interface(UI), we first used tkinter in Python to implement. In that we have not implymented the hardware part yet at that time. In the first version of UI, we only have functions with users' name input, cities selections and route desplay in a 2D map. The UI is shows below:



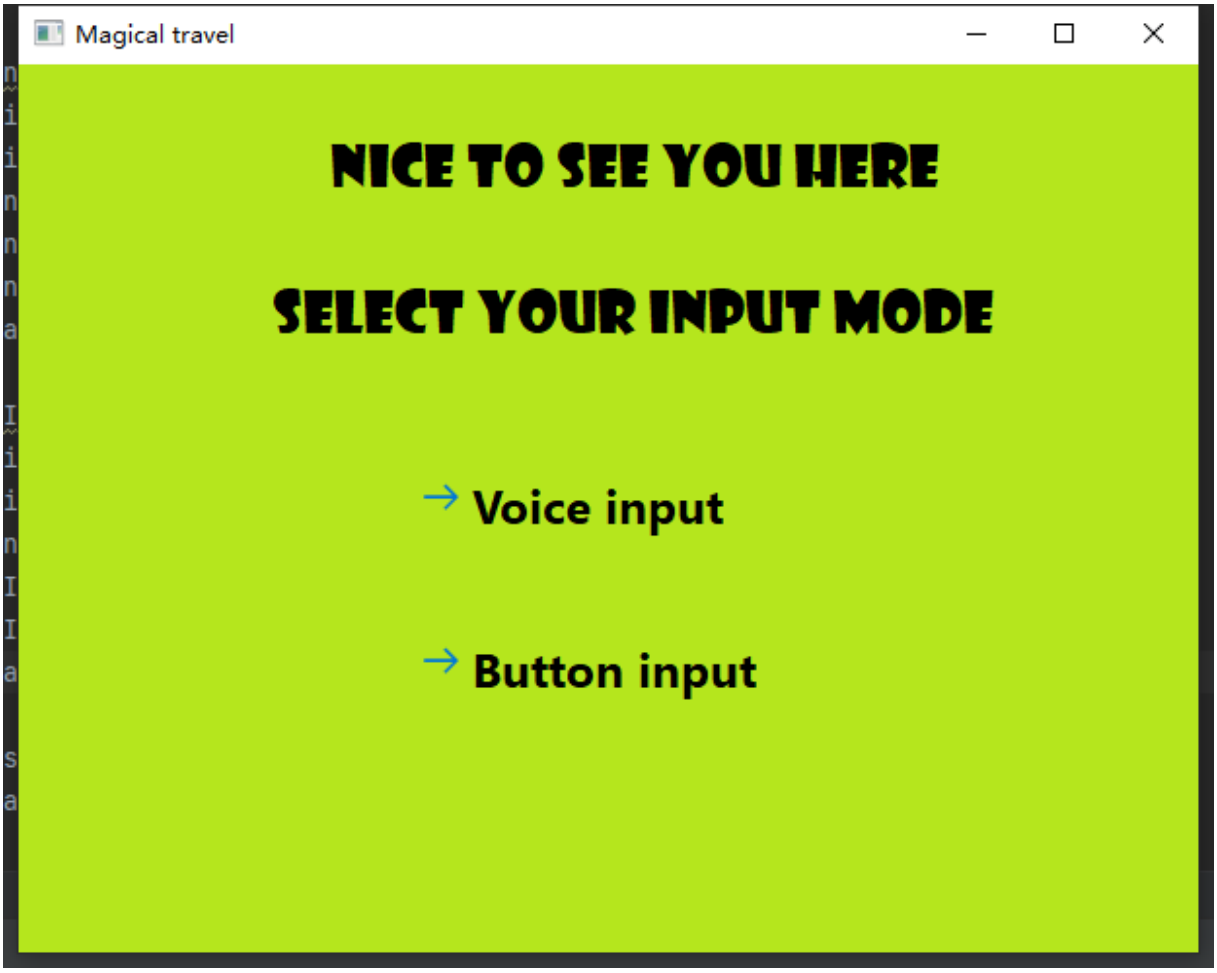


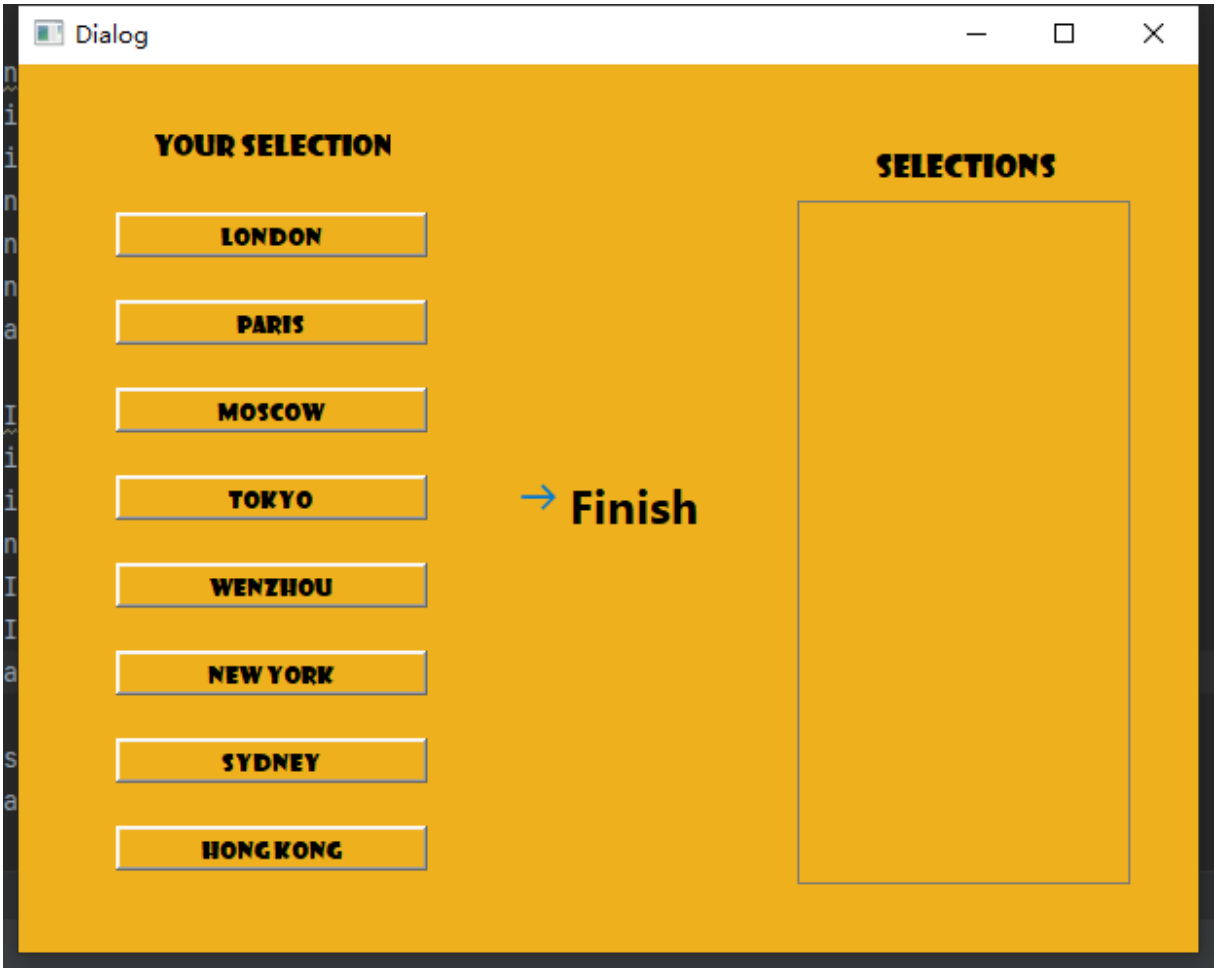
The 2D map is implemented with Basemap in Python, which basic on matplotlib, so it does not look nice. In the map, when user's input, it can locate each city and conncet then with lines in order. Meanwhile, the initial city and destination city can be marked with stars in different colors. The map is showed below:



6. Initial Attempts

After first attempt, we tried to polish our UI. As tkinter cannot inport background and design appearance for icons, we decided to use PyQt5 to implement our new UI. With various functions in PyQt5, we changed our buttons appearances, font and background colors. We also deleted the name input part in that input names every time will waste users' time and not functional at all. The appearance of the UI is showed below:


Magical travel
NICE TO SEE YOU HERE
SELECT YOUR INPUT MODE
→ Voice input
→ Button input



We did not changed the map part. After the work of hardware part, we realized that first, make UI with Python is still limited, we cannot implement animation and other awesome visual functions;second, we canned receive the sign from hardware with Python code; third we failed to embed vioce input function into Python UI. As a result, we decied to use games engines to make our UX part.

Worldmap Globe Edition (Default)

Introduction

In response to the need to display and record WKU faculty's travel paths around the world, we developed an application using Unity. The application is currently in its infancy, and the version number is V1.0.

Description

2.1 Product Functions

World Map Globe Edition is a digital model of the earth. We use digital technology and methods to organize the temporal and spatial change data of the earth and its environment according to the coordinates of the earth and store them in the computer to form a global digital model. This allows people to quickly understand the planet we are on. The main functions are as follows:

World Map Globe Edition is a digital model of the earth. We use digital technology and methods to organize the temporal and spatial change data of the earth and its environment according to the

coordinates of the earth and store them in the computer to form a global digital model. This allows people to quickly understand the planet we are on. The main functions are as follows:

2.1.1. Real-time latitude and longitude monitoring

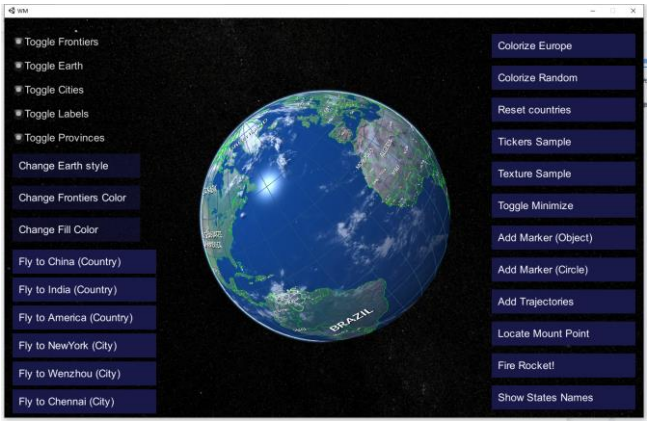
We divided the earth with a grid of latitude and longitude, and set a crosshair composed of red longitude and red latitude to locate the user's current position and transmit it to the monitor at the bottom of the screen in real time.

2.1.1 Latitude and longitude monitor

2.1.2. Earth display mode conversion and free viewing mode

We set a slow rotation speed and starry sky background for the earth to simulate the real earth's position in the universe. In order to enhance the overall interest of the software, we set the free rotation of the lens to realize the user's manipulation of the earth. Users can drag the earth to rotate it to the area they want to observe, and we will also provide corresponding sunlight to light up this area. In this way, the user can clearly observe the divisions of the day and night hemisphere (and the ring light in the backlight zone). Being able to rotate the earth freely can make users more like to use the software.

2.1.2.1 The day and night hemisphere



2.1.2.2 The day and night hemisphere

2.1.3. Country and detailed province information

Our software has a complete database of regional division information of all countries in the world, and can realize the internal division of all countries (such as states in the United States, and provinces in China). Users can observe the division of different regions through strong contrasting colors. While the software gives the latitude and longitude information, it will display the neighboring countries (or the neighboring provinces of each province) at the position of the front sight of the mouse. Users can also select different viewing modes by checking the buttons on the upper left.

2.1.3.1 Province nearby

2.1.3.2 State nearby

2.1.3.3 Country nearby 1

2.1.3.4 Country nearby 2

2.1.4. Area jump and path display

In order to simulate the real situation of cross-border (or cross-city) airplane travel, I inquired a lot of data and information about passenger planes flying. After setting the minimum ground clearance and the shortest path between the two cities, use these two data to calculate and draw the flight arc. The flight trajectory will stay on the digital earth for 40s for observation, which is convenient for the faculty to choose a suitable travel path. In selecting the target location, our database is detailed to the level of the city, and the software will accurately locate the selected city and teleport . Of course, we also provide teleport options for more vague countries. When teleporting, the country area will be highlighted so that the user can easily observe the selected country.

2.1.4.1 Flight line

2.1.4.2 Highlight of the country

2.2 Operating Environment

2.2.1Software environment:

Windows 10 and above.

2.2.2 Hardware environment:

Dual-core 3GHz (and above) + 1G RAM memory (and above) + 80G hard disk capacity (and above)

Design and Implementation Constraints

3.1 Challenge

3.1.1.Interaction with hardware devices

We use the Arduino motherboard to make our hardware control panel. At first we decided to use Bluetooth to realize data interaction with PC, but the situation is not optimistic. The serial port communication performed by Bluetooth is slow and prone to garbled characters. So we decided to use USB to TLL to connect directly to the computer to achieve data transfer. The data output of the Arduino board is through the serial port. We initially saved the data in a txt file and used the software for iterative monitoring. However, this method is inefficient and prone to failure. After Professor Relic's guidance, we directly called the computer's serial port and obtained the data through the C# method, and finally completed the transmission through the hardware control area.

```
public GameObject target;//目标操作物体
private SerialPort sp;
private Thread recvThread;//线程
string s = "";

public GameObject rocketPrefab;
```

```
void Start() {
    //SerialPort
    Debug.Log("ZhengChang");
    sp = new SerialPort("COM3", 9600, Parity.None, 8, StopBits.One);
    //串口初始化
        if (!sp.IsOpen){
            sp.Open();
        }
    recvThread = new Thread(ReceiveData); //该线程用于接收串口数据

    recvThread.Start();
```

```
private void ReceiveData(){
        try{
        //string s = "";
        //以行的模式读取串口数据
            while ((s = sp.ReadLine()) != null){
            //s = sp.ReadExisting();
            //Console.WriteLine(s);
            Debug.Log(s);
            print(s); //打印读取到的每一行数据
            }
        }
        catch (Exception ex){
        Debug.Log(ex);
        }
}
```

```
if(s.Equals("2")){
Debug.Log("success");
FlyToCity("Chennai");
int countryIndex = map.GetCountryIndex("India");
map.BlinkCountry(countryIndex, Color.black, Color.yellow, 4, 2.5f, true);
s = "";
}

if(s.Equals("8")){
Debug.Log("success");
FlyToCity("Wenzhou");
int countryIndex = map.GetCountryIndex("China");
map.BlinkCountry(countryIndex, Color.black, Color.green, 4, 2.5f, true);
s = "";
}
```

3.1.2.Simulation of air flight trajectory

It is very difficult to simulate the flight trajectory in the air. After querying the corresponding information, I set the minimum ground clearance and the coordinates of the two cities, and then used these two data to calculate the flight arc and draw the line. We need to store the latitude and longitude coordinates of the current location and the target location, Set two Vectors to represent the starting location and the final location respectively. Then I set up a function to calculate the shortest arc between two points at a certain height above the ground. Finally set the width and duration of the line, and the flight trajectory is formed.

```
/// <summary>
/// Adds a line to the globe with options (returns the line gameobject).
/// </summary>
/// <param name="Color">line color</param>
/// <param name="arcElevation">arc elevation relative to the sphere size (0-1 range).</param>
/// <param name="duration">drawing speed (0 for instant drawing)</param>
/// <param name="fadeOutAfter">duration of the line once drawn after which it fades out (set this to 0 to make the line stay forever)</param>
public LineMarkerAnimator AddLine(float latitudeStart, float longitudeStart, float latitudeEnd, float longitudeEnd, Color color, float arcElevation, float duration, float lineWidth, float fadeOutAfter) {
    Vector3 start = Conversion.GetSpherePointFromLatLon(latitudeStart, longitudeStart);
    Vector3 end = Conversion.GetSpherePointFromLatLon(latitudeEnd, longitudeEnd);
    return AddLine(start, end, color, arcElevation, duration, lineWidth, fadeOutAfter);
}

/// <summary>
/// Adds a line to the globe with options (returns the line gameobject).
/// </summary>
/// <param name="start">starting location on the sphere</param>
/// <param name="end">end location on the sphere</param>
/// <param name="Color">line color</param>
/// <param name="arcElevation">arc elevation relative to the sphere size (0-1 range).</param>
/// <param name="duration">drawing speed (0 for instant drawing)</param>
/// <param name="fadeOutAfter">duration of the line once drawn after which it fades out (set this to 0 to make the line stay forever)</param>
public LineMarkerAnimator AddLine(Vector3 spherePosStart, Vector3 spherePosEnd, Color color, float arcElevation, float duration, float lineWidth, float fadeOutAfter) {
    CheckMarkersLayer();
    GameObject newLine = new GameObject("MarkerLine", typeof(LineMarkerAnimator));
    newLine.transform.SetParent(markersLayer.transform, false);
    newLine.layer = markersLayer.layer;
    LineMarkerAnimator lma = newLine.GetComponent<LineMarkerAnimator>();
    lma.start = spherePosStart;
    lma.end = spherePosEnd;
    lma.color = color;
    lma.arcElevation = arcElevation;
    lma.duration = duration;
    lma.lineWidth = lineWidth;
    lma.lineMaterial = GetColoredMarkerLineMaterial(color);
    lma.autoFadeAfter = fadeOutAfter+40;
    lma.earthInvertedMode = _earthInvertedMode;
    lma.reuseMaterial = true;
    return lma;
}
```

3.1.3. Model rendering of the digital earth

How to make a beautiful digital earth is also a big challenge. First create a high-definition spherical 3D model. Then find different levels of tile maps (tile maps: in simple terms, the number of tile maps at different levels is different, the higher the level, the more the number of tile maps, the more detailed the map displayed. The map of a certain tile level The tiles are composed of 4 tiles cut from the lower-level tiles, forming a tile pyramid). So the most important thing is how to obtain tiles of different levels. If you can connect to the Internet, I can obtain these tile map resources online by applying for the BingMap Key, and then load them locally. In this case, the rendering of the digital earth is basically completed.

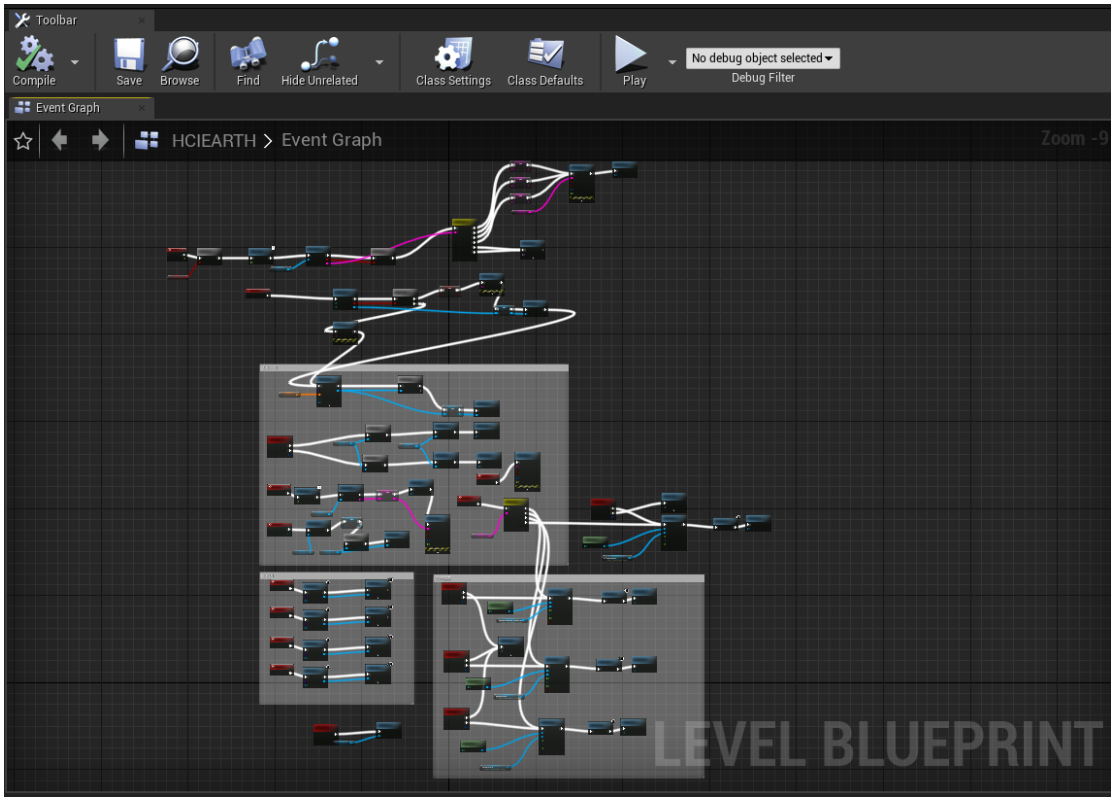
Realmap Edition (Accessibility)

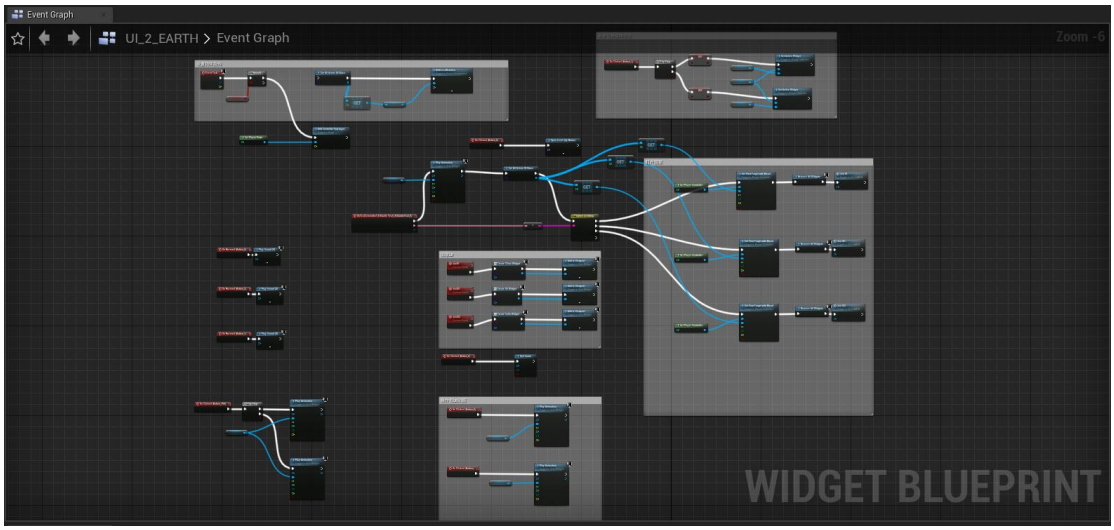
Introduction

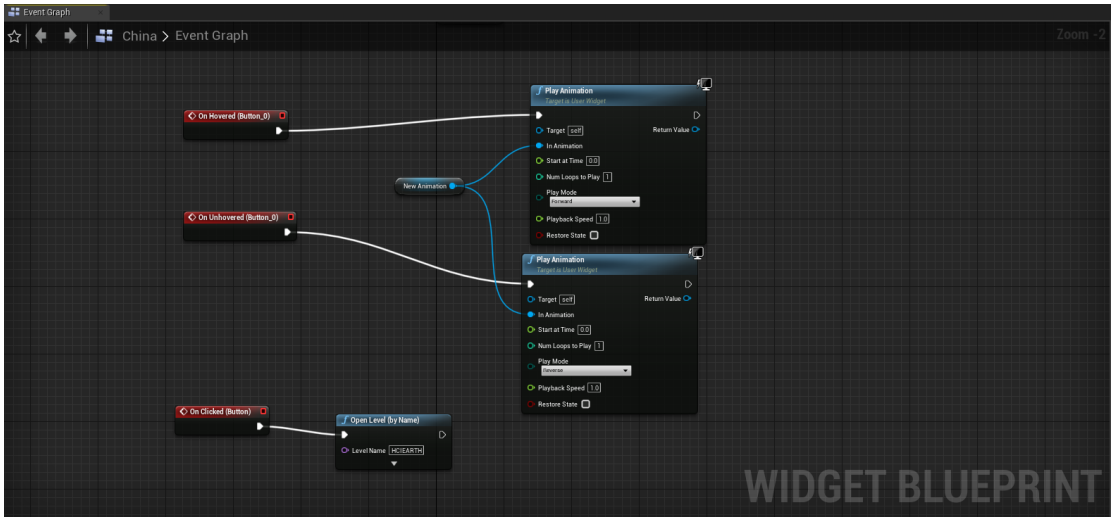
This Real Earth is made by the Unreal 4 engine, which uses a lot of blueprints and C++ scripts to realize the display and logical interaction of an earth. In this software, users can view the panorama of the earth and certain countries. In addition, users can view specific country information and location through text, voice and hardware input.

Implementation & Challenge:

In order to realize the functions of the application, we have created many blueprint classes for each level and object during the development process:

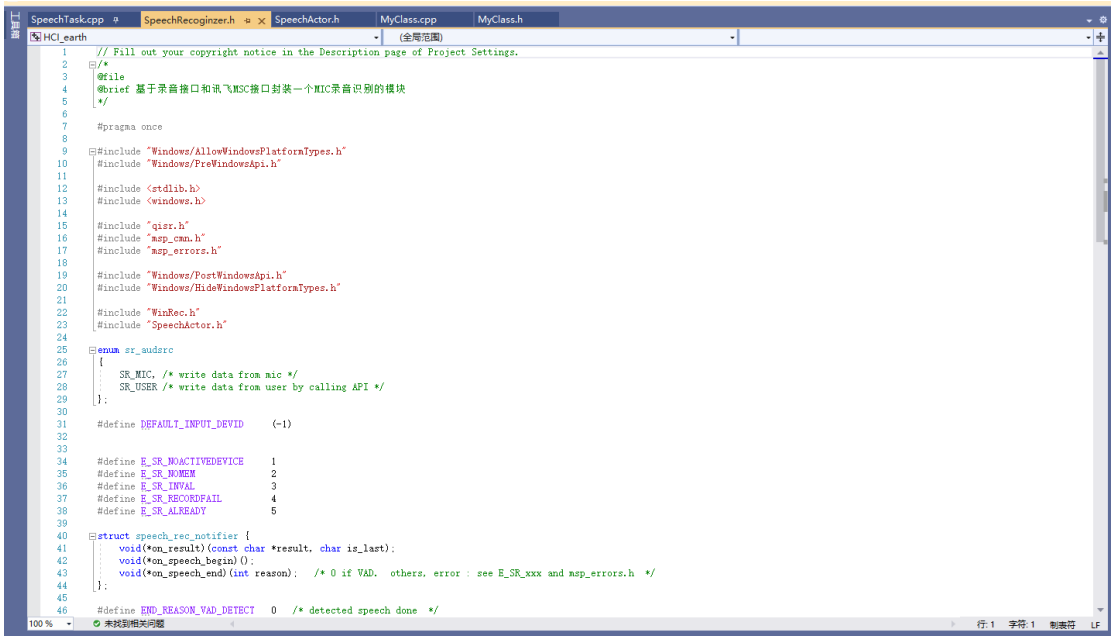




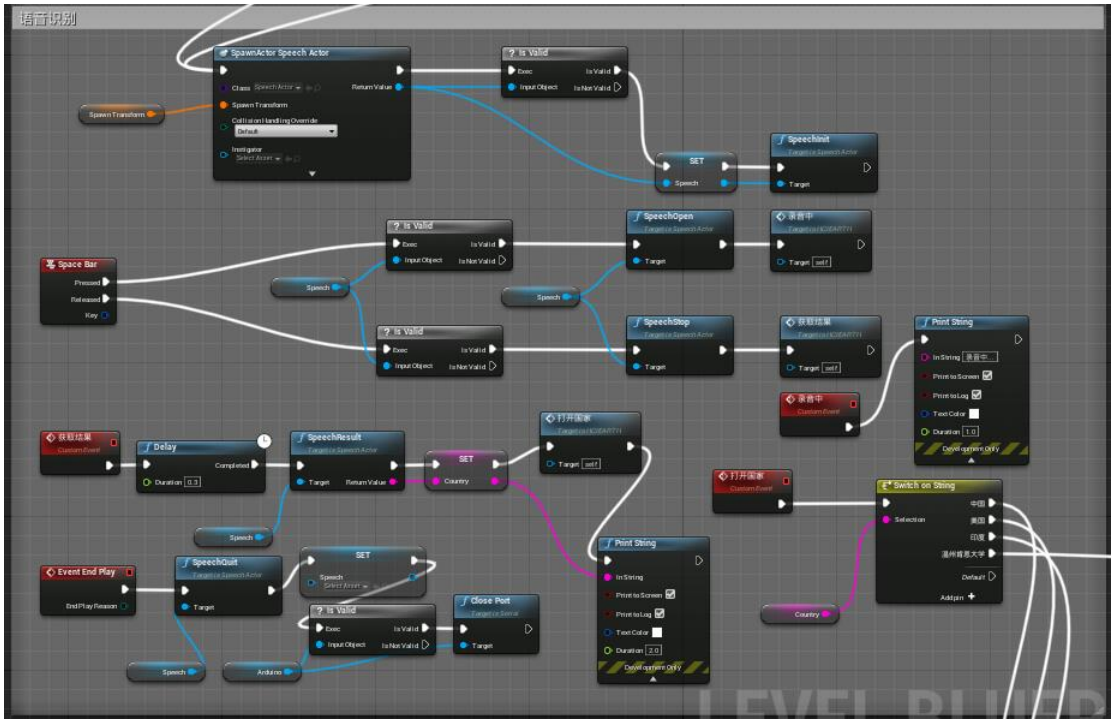


Only the three most important blueprint classes are shown here. In these classes, we have realized the way of displaying and interacting with the earth, and the receiving and processing of three kinds of inputs. Other blueprints that only contain simple logic are not shown here. During the development of Unreal 4, the biggest challenge was voice and hardware input.

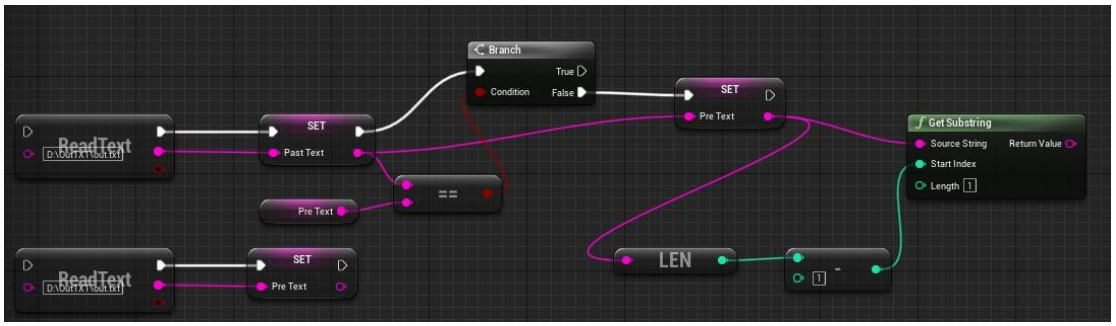
The challenge of voice input: First of all, it is a long work to find a C++ plug-in for voice recognition that is suitable for the Unreal 4 engine. After that, we create a plugin file in the root directory of the project and import the downloaded speech recognition plugin into it. Due to configuration issues, the imported plug-in cannot run successfully in the project. Troubleshoot the problem step by step through the error log, and finally adjust the plug-in by changing the environment of the APIID and reference address.



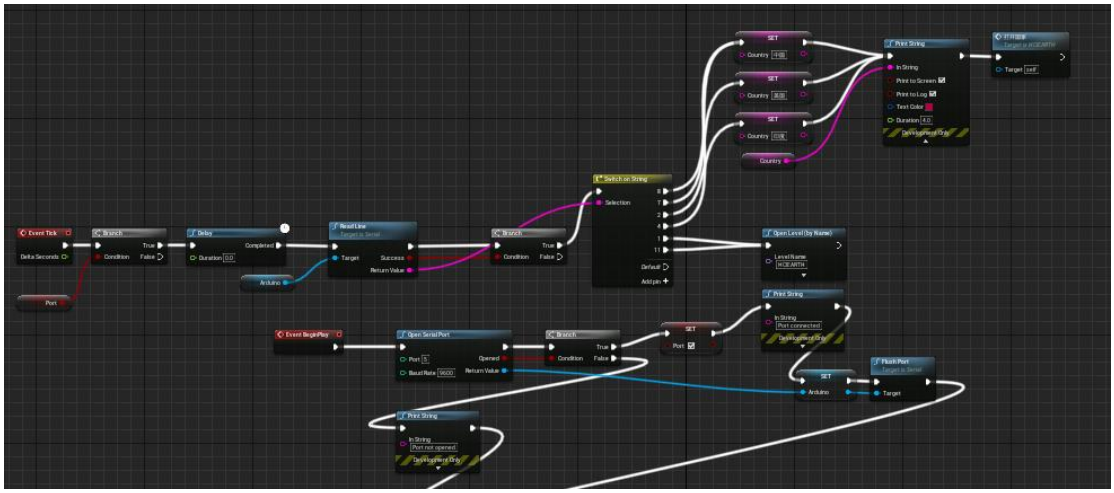
After that, through the blueprint call and logical arrangement, the function of language recognition was realized in the project. The following figure shows the logical blueprint of voice input:



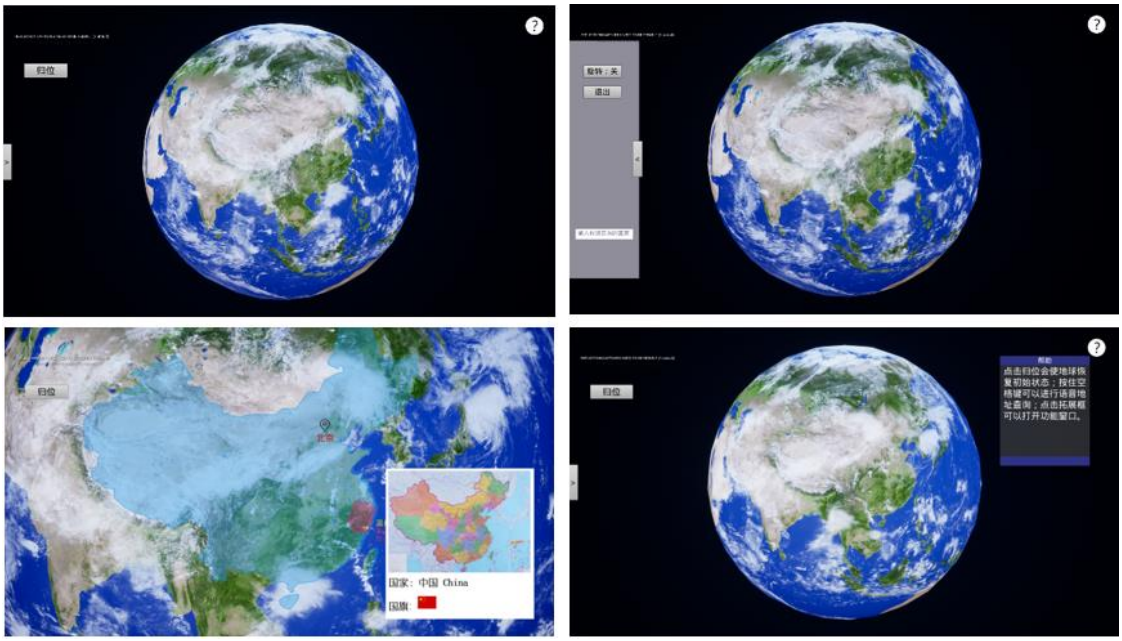
The challenge of hardware input: The input of Adrnoid hardware is a new txt file. You need to write a command to monitor the txt file in Unreal 4:



The function of this command is: the program will monitor the specified txt file (out.txt) every frame. When the content of the file changes, the program will automatically detect the last input signal and process it. However, during the test, we found that the hardware serial port will always occupy the reading of the txt file when it is opened, so the program cannot read and monitor the changes of the txt file. Later, we directly read the Arduino signal through a plug-in, without indirect transmission through txt. So we found the UE4 Duino plug-in for Unreal 4 and completed the interaction between Arduino and Unreal Engine with the help of an older version of Unreal Engine. The following figure shows the logical blueprint of hardware interaction:



After completing the basic functions of the program, the team optimized the UX of the program. First of all, except for the reset program, which is the most important and commonly used function, all other functions are put into a window, so that the program interface will look neat and tidy. When the user opens a country, the interface will display the country's information, national borders and capital. In order to prevent users from not knowing the available functions, a help button has been added to the upper right corner of the main interface. When the user clicks the button, it will display the complete software usage guide. For the voice input function, when the user presses the space bar, the upper left of the screen will display recording, so that the user can understand that the voice signal is being received normally. When the user moves the mouse to the function button, the corresponding voice content will be broadcast, so even the visually impaired people can use our software to explore the earth. The texture used by the project team is an 8K high-definition large image, so the model can be displayed clearly even on a larger resolution screen.



TIPS: The source file is too large to upload through the blackboard, so we upload all the source files to the following Google Drive folder: CPS3601 Project1 Iter, you can access it through the following URL:
https://drive.google.com/drive/folders/1_-yrhxiQGnSK1A6r9QaBV3eiCtH1jJrr?usp=sharing