Project documentation for Group H22

There are two major parts for our program, there are the inner code for our game and the GUI for the game. The inner code will handle almost all behind logic for the documentation. And the GUI is the part that our program interacts with users.

1. The design of code (sorted by the class name):
   1. Continent: This class use the stl::list to store every country’s pointer in the continent and using iterator to search for country and to count score for the continent
   2. Country: Using list to represent the neighbor of the given country. Also in this class we introduce the concept of comparable for the country for the AI part of the program. We implemented the operator of comparison for this class, to determine the weight of decision of a given country in our AI algorithm.
   3. MainWindow:

This is the class for the major window that the player will be interacted with. Every country button in the ui is controlled by two maps, The USSR\_button\_map and USA\_button\_map, these two maps has the key as the string of name of the country this can hugely save the coding cost whenever the usage of button is make. Also, to generate the part of code for allocating all the button, a Python program is also written to automatically generate such codes. This map will be used a lot in the AI part and also handle the part for the on\_click functions of the countries. To play the background music we also add the QMediaPlaylist data structure and used the random play method for the music to add the chance of randomness.

In the MainWindow class, the two pop up window will be shown, one is the coup dialog and another is the event pop up window, they will be shown in the certain time (for specific time, see the comment for detail).

* 1. MainWindowAI:

This is the class that inherits from the MainWindow for the AI play mode. Since if we choose the signal player mode then the pointer of MainWindow will pointed at the AI window, not the Base class window. Since the similarity of the AI window and the MainWindow we use the inheritance to implement the MainWindowAI. In this class, we use the concept of decision tree to decide the move of the AI. The first layer of decision tree is to judge which condition is the current game in, this is handled by the historical trend i.e. the turn system. Then goes into the certain the decision\_tree judge the continent of the given continent.

After entering the decision tree, we used a priority queue to update for the current situation of the world and the AI. The queue stores the country’s decision weight, when doing decision, the queue will pop and the weight (getting from the operator >, <, == we just implemented) and then the AI will handle for the action of given country, to simplify the decision sequence, the buttons is get from the map to give the updates for GUI.

Also, since the AI is occupied for one player, the function of on\_btn\_next\_clicked() override the base class such function in the base class is declared virtual.

* 1. World:

This is the class where main content of the game is stored. We store the country of the world as public data member and manage them using vectors. There are two vectors, one stores all the continents one stores all the countries. Also, the superpowers are stored as public enum in our program. Also in this class, we use the iostream to handle the save and load. We override the constructor of the World, one by default setting and another by given the filename to find such files.

We merge the idea of graph into our World class since the way that we manage the country is fundamentally satisfy the definition of graph, can add vertices and also add or find the connection between vertices.

* 1. MenuWindow:

This class is for the menu window of our program, in this class, we use Liskov to handle the different case of AI (signal player) or non-AI mode of our game.

1. How different parts of the code are integrated to make the program functional:
   1. The first step for the player to do is to decide whether they will load the game that they played last time if they do not, initialize the world with default constructor, or initialize the world with the constructor of filename (using file dialog to choose the file). We will initial all the countries and all the continent in the constructor of the world using helper function (if has influence in the save file, initialize them together with the world). In the menu window the user’s decision will also determine what kind of main window we will use (MainWindow or MainWindowAI) then it will initialize the chosen class. Then close the menu window.
   2. Then enter the initialization of MainWindow, we will initialize all the object of the window and if in not loading case, also initialize all the basic influence in this step. If in the load constructor, the MainWindow will initialize all the button status of the world.
   3. After enter the MainWindow the event popup window will pop when the player change also whenever you click a button, firstly we will using the searching function in the World to find the given country’s pointer then using the Country pointer to update the influence change after the button is clicked. Then update the GUI for the clicked button.
   4. If the coupe mode selected when click the button of the current player side, the coup dialog window will show up, to let the user to choose the point that they want to use for the coup. The dialog window will return the number of points players will use to do the coup.
   5. If the next button is clicked, the next() function in World will call and handle everything of the world’s change. If the save is clicked, the program will handle the selected filename to the World class and the save function in World will handle for the save.

Here is the high level view of the class integration in a logical graph:

图示

描述已自动生成

1. How the code can be reused:
   1. The way that we implement the World can be used in all other games that represents the history, such as the game of EU4 and Civilization the mechanic of our World class is similar to many other board games. After some little change can be directly used in other program.
   2. The pop-up event can also be reused in any other games that requires pop-up window
   3. The CoupDialog that we use in this game is useful in the other program that requires input of users during the runtime of the game. Because of the dialog mechanism requires the users to finish the input to continue the program.
   4. The idea of managing AI window which is the window of single player is the derived class of the main class and all loops for the AI is in the derived class, is useful in the game design that requires similar mode of the game, such as different degree of difficulties can be different derived class of the basic main window.
   5. The system of our country system (the way we manage the countries) is fundamentally the implementation of graph, which only need slight change to be a general graph data structure to use in all other kinds of places that requires graph