Noah Thederahn

Artificial Intelligence Agent for Solitaire Play of *Twilight Struggle*

*“Now the trumpet summons us again—not as a call to bear arms, though arms we need; not as a call to battle, though embattled we are—but a call to bear the burden of a long twilight struggle, year in and year out, rejoicing in hope, patient in tribulation—a struggle against the common enemies of man: tyranny, poverty, disease, and war itself.” John F. Kennedy, Inaugural Address, January 20, 1961*

**Overview**

For my senior thesis I intend to create a program which will act as an AI opponent when playing the board game *Twilight Struggle.* In *Twilight Struggle,* two players pit themselves against one another as either the United States or the Soviet Union, spreading geopolitical influence across the globe in a simulation of the events that made up the Cold War. The reason I chose *Twilight Struggle* as the focus of my project is because it is the number one ranked game on BoardGameGeek.com and has been since December 2010. This is what initially drew me to the game and gives me reason to believe that an application of this nature would be positively received by the gaming community.

Gameplay in *Twilight Struggle* centers on players alternating the play of cards obtained from a shared deck; these cards can either be played to enact a unique historical event, which can have a wide variety of effects on the game, or they can be played for a numerical point value which can be used to take one of three different actions to change the current political influence values on the board. This type of gameplay mechanic is called a card-driven game.

The first and most common action that will be taken is adding an amount of political influence points, equal to the numerical point value of the card, to various countries on the board. One can place influence points in countries they already have influence in or countries adjacent to those they have influence in, with both countries having influence in various countries at the start of the game. Another common action is to use the point value of a card to perform a coup in a single country. The goal of a coup is to lower your opponent’s influence in a country and possibly replace it with your own. The last way to use the point value of cards to affect countries is realignment. One can attempt realignment in a number of countries up to the point value of the card, the goal being to lower your opponent’s influence in said countries.

There are also scoring cards which give players points based on the amount of control they have over a certain region of the world. Score is a single sliding value; one player gaining a point is equivalent to the other losing a point, with the extreme ends of the track being an immediate victory for each player. These cards are rare however, with more being added as the deck’s size is increased with additional sets of cards during the course of play. Full rules for the game can be found on the publisher’s website[[1]](#footnote-1).

In terms of project design, my plan is for the program to be an independently running executable with a graphical user interface (GUI) that will run on the Windows operating system. The program will function in a way that requires the user to have at minimum all the cards required to play the game. All dice rolls will be done by the program itself for simplicity, and all relevant board information will be displayed as well; though a physical map will be recommended to determine country adjacency. Interaction between the player and the application will involve the player making various selection choices based on the current state of the game or pre-game setup environment. The actions taken by the AI player will be represented by text-based updates stating what was done. The UI will be continuously updated after both player and AI actions to correctly reflect the current gameplay state.

**Related Works**

Although this game is highly popular, there are only a limited number of methods to play by oneself. There are only three fan-made rules variants on the game that allow solitaire play of the game; all of these are either new or modified rules to follow, or a logic flowchart for the AI player to follow. This fact makes my intended project unique in the field of solitaire variants since mine would automatically make all the decisions for the opponent because it is the only digital variant. In addition, none of these variants seem to quite fit the niche that I intend to make my project fill since they all require modifying the basic gameplay structure to a certain degree, while I intend to make my program retain the same sequence of play that would be present with two human players. However, despite the fact that these variants differ from my own, they are still useful in the creation process of my project by giving me direction and ideas on how the strategy of my AI will work.

Other works that are of relevance for this project are the solitaire systems developed by Mark Herman for the card-driven games he has designed such as *Churchill* and *Empire of the Sun*. Mark Herman is a board game designer who is most famously known for his work on *We the People*, the first card-driven board game (a type of game in which players choose and take their actions though the use of cards they are given). Although it was not designed by Mark Herman, *Twilight Struggle* also uses this type of game mechanic; therefore the methods used for his solitaire systems are what I plan to base the design for my AI agent off of. Specifically, I will be using the game *Empire of the Sun*, a wargame set in the World War II Pacific Theater. This system works by giving the player a flowchart of high-level conceptual objectives, and then directives on how to proceed with implementing those objectives. Certain hierarchies of card play are given depending on the situation in question, but the ultimate choice in some matters is still left up to the player to decide for the opponent. For example, it is up to the player to choose which units their opponent will be activating in any particular military offensive. While *Twilight Struggle* is a much less complicated game, it still uses the same core mechanic as *Empire of the Sun*, and the solitaire system in the latter operates by using a type of decision tree. For this reason I feel that a decision tree is the best agent type to use in my project.

**Project Breakdown**

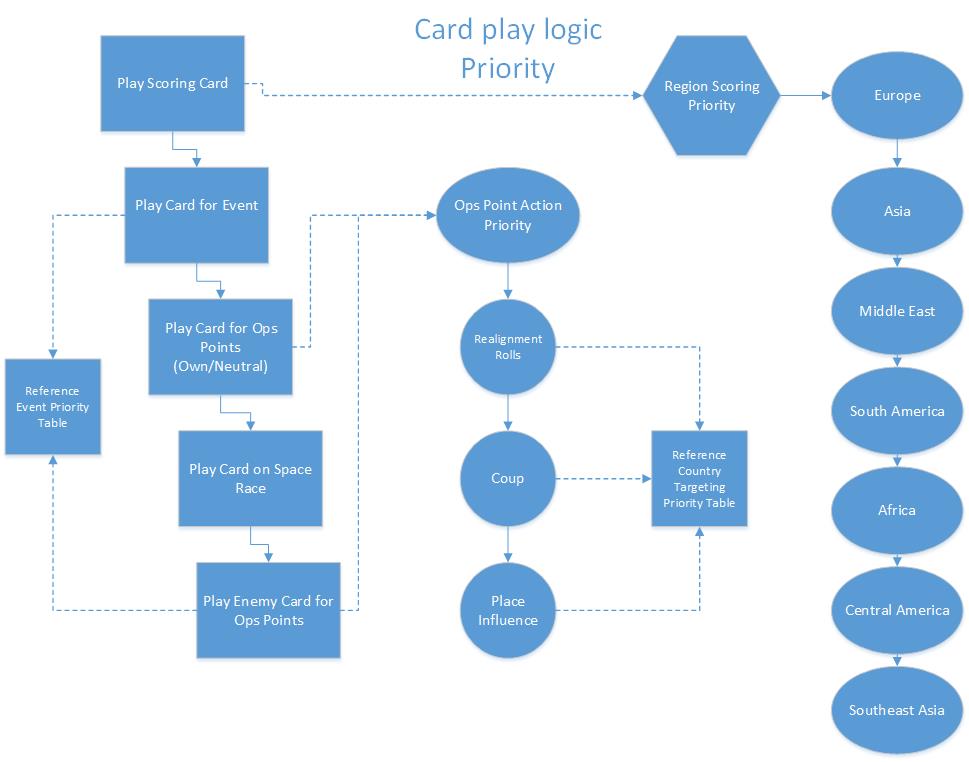
My goal in this project is to create a fully functioning executable that will run on Windows operating systems. The specific type of application I plan to use will be a Win32 application, which can run on all versions of Windows since 95. The type of AI agent that this program will use for its decision making process is a logic tree. Some of the major components of this project are the components of the GUI, the knowledge database that would contain the logic tree, a series of objects that would represent the information of the current game-state, and a database of all the necessary information on the one-hundred and ten cards used in the game.

**GUI**

The main reason I am choosing to use a Win32 application is that it is the easiest application type to make that will run a GUI on Windows. Making this project a Win32 application will set several different aspects of it firmly in stone. The first, and arguably most important, is that it will be coded in C++, which is a requirement of this application type. Another important aspect of the Win32 application type is the fact that it will run on all recent versions of Windows without compatibility issues. Being a 32 bit application it will only be able to access up to 2GB of memory, but with the relatively low level of complexity my program will be running I do not believe this will be an issue. I estimate this to take approximately three to four weeks to fully implement this portion of my project in a working order.

**AI Agent**

The second major component to my application will be the AI agent which actually plays the game. The type of agent I will be using is a decision tree. Implementing such a structure will be done using a node tree, with each node being a specific decision to be made. A high level conceptualization of the logic flow is illustrated in the following chart:



Each box represents a high level decision to be made, with dotted lines going to high level condition checks of their parents. What this diagram does not show is the necessity for each box to make a check, which will determine if conditions to take their action have been met before going on to the next box. Also not shown is the table that will prioritize card events linearly for each faction. For example, the conditions to play card “A” will always be checked before the conditions for card “B”, which will be checked before the conditions for card “C”, etc. Though this linear hierarchy may change to something slightly more elegant in the future, this will require further researching into the strategy the AI will use. The table that is responsible for prioritizing countries for certain actions may follow a similar flow to the prioritization for country scoring, but this needs further research before it is finalized. Since this is the most important part of my project, and the most in depth, I believe it will take me four to six weeks to implement in its entirety.

**Game-State Representation**

One lesser component for my project is the representation of the game-state. Representing the game-state should be a relatively simple affair, since most of the information can be broken down into simple variables of either Boolean or numerical value. Representing these with various interrelated objects would be the most straightforward approach, with the most complicated portion being how to represent the adjacencies of countries. The countries could be implemented with a series of interconnected nodes, with a list object able to index to any particular country. I estimate this taking me one to two weeks to fully implement.

**Card Database**

Creating the database for the cards is another relatively simple aspect of my project. Since each card has a unique ID number in the game, this makes it extremely easy to catalog and reference them in an organized and logical way. I plan on putting all relevant information into a list so that each card’s information can be accessed by indexing its ID number. Each card entry will need a function to check if the conditions to play its event are met, and then execute said event if the conditions are true. This will mean a total of at least one-hundred and ten unique functions, but each one should be no more than a few lines long, since most cards only modify a few different variables each. I estimate this taking me only a week to complete and possibly another week to integrate it in with the rest of my project.

**Technologies and Skills**

While the majority of the components in my project can be completed with the knowledge I already have, there are still some essential things I will need to learn in the course of developing this project. The most important thing I will be learning is how a Win32 application operates. I have been following some tutorials that teach the basics of how these applications work, but I am not as familiar as I need to be with Win32 applications in order to complete my project. I am familiar with the basic operating structure of these programs however, and feel that a minimal time investment into researching them further will yield sufficient knowledge. Another area which I will be learning about during this project is using Microsoft’s IDE. Microsoft Visual Studio is freely available, and required to compile to a Win32 type application. I have only minimal experience using it, and since it is what many of workplaces will be using, learning its nuances will be a valuable job skill. I do not believe it will be nearly as difficult as learning how a new application type works however. In conclusion I am confident that I will be able to complete my project on time.

**Timeline**

The following table is an estimated timeline of my progress in making my project. The times given are only an estimate but do represent my overall goals.

|  |  |
| --- | --- |
| Week 1 | Finish tutorials on creating a Win32 application. Begin initial framework on program. |
| Week 2 | Code basic structure of application. Have a running application shell done by end of week. |
| Week 3 | Finish basic application functions. |
| Week 4 | Implement game state representation. Finalize logic flow used by decision tree. |
| Week 5 | Begin implementing decision tree. |
| Week 6 | Finish implementing decision tree. Test decision tree for logical correctness. |
| Week 7 | Implement decision tree into Win32 application |
| Week 8 | Create card database. |
| Week 9 | Implement database into application. Bug testing. |
| Week 10 | Final testing to ensure everything works correctly. |

**Grading Breakdown**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grade | GUI | AI | Game State | Card Database |
| A | Application dynamically displays all game state information at all times. | Agent performs “B” level decision making with minimal downtime “thinking”. | Game state implemented. | Querying from the database takes minimal time |
| B | Application dynamically displays cards in user’s hand. | Agent’s decision making skills are on par with a human opponent. No illegal decisions are made. | Game state implemented. | Database implemented and functional. |
| C | Application runs independently with user making selection from various context driven buttons. | Agent follows intelligent decision making. Illegal decisions are rare. | Game state implemented. | Database fully implemented. Not all functions work correctly. |
| D | Application runs independently with input and output in a text only format. | Agent makes choices in a somewhat logical order. Illegal decisions are still common. | Game state not fully implemented. | Database implemented though cards used mid-game. Not all functions work correctly. |
| F | Application runs at command line level only. | Agent makes choices seemingly at random. Many illegal decisions are made. | Game state not implemented. | Database only partly implemented. |

**Conclusion**

I believe that I can complete this project in an estimated ten weeks. I believe this project is fully within my capabilities as a programmer and as a student of Hanover College. In terms of skills, I have the necessary experience using the language, but I will be learning how to create an entirely new type of program that I have never worked with before. I will be learning practical skills using Microsoft’s IDE and interacting with the Windows API. I will be filling what I see as a glaring gap in the ways that *Twilight Struggle* can be played by one person. And I will be creating something which people besides myself will be able to use and enjoy. For these reasons I believe that my proposed application is of sufficient breadth and substance to be a Capstone Project.

1. http://www.gmtgames.com/p-557-twilight-struggle-deluxe-edition-2015-reprint.aspx [↑](#footnote-ref-1)