Warhammer 40000 Legiones Astartes Age of Darkness Army Selector/ List Printer

Ian Nevills
James Laubach
Nicholas Reid

Abstract

The goal of this work is to condense the army selection book (hitherto: Legions Astartes Book) and the rules necessary for play. Specifically, condensing the Core Rules Book into an easily navigable format to allow for the quick construction of army lists/builds for veteran and novice player of Warhammer 40000 alike.

1. Introduction

Warhammer 40k is an expansive and very complex tabletop game and its rules can be daunting even for veterans of the game. For novices, navigating thick tomes of factions upon sub factions of units and their variants to forge their chosen army list can be quite difficult. Our goal is the condensing of the singular faction book Legion Astartes, "Space Marines" within the setting of the "Horus Heresy" expansion of the base 7th edition Warhammer 40k. This will streamline the army creation and therefore the game play by providing an army-list document that provides reference to rules. With that in mind, however, we are not seeking to make it so that any person could use our program in lieu of buying the books in question. Our plan is for our program to allow the user to search/page through a catalogue of units pertinent to the faction and question and select units within the bound of standard army templates in the book. Also enforce the point limit imposed in the normal game play and output a formatted and easily navigated document that provides unit information, such as basic stat lines, to streamline game play.

1.1. Background

Warhammer 40k Legions Astartes Age of Darkness Army List

Warhammer 40k Core Rules Book 7th Edition

Warhammer 40k is a tabletop science fiction strategy game set in a dark and distant future. Within 40k there are many different factions with equally different game-play fact and origin stories. Warhammer 40k is a game literally buried in lore and reasons for the how and why its factions behave and interact the way they do. In fact, a more accurate description may be a tabletop science fantasy game given that anyone familiar with Tolkiens novels would recognize and anyone familiar with other fantasy tabletop games such as Dungeons and Dragons. With that in mind, each of these races is at first familiar by name but very different then their traditional archetype. A further description though is beyond the scope of what we intend to cover at this time. We have chosen the Space Marines as our subject for this project as they are probably the most widely recognized and have become the poster children of the game. Because of their prominence they are most likely the faction you would introduce new players into the game via the vast number of novels and games based around them. Space Marines in the context of Warhammer 40k are genetically engineered soldiers that are one-part warrior monk, one-part Roman legion, one-part crusading host and one-part Arthurian legend wrapped in a ton and a half of power armor. They are the ultimate soldier and the peerless defenders of humanity, sent into the harshest fighting and the most important conflicts. The actual game-play is much less stellar. At their core, the Space Marines are a force dedicated to somewhat durable heavy infantry formations with a wide selection of weapons and upgrades and a well-rounded stat-line. Everything else in their lineup are either walkers intended for direct support of those infantry or various vehicles and tanks that almost universally double as battle taxis to ensure the infantry reach where they need to be and provide a large enough wall of guns to ensure they can do their job when they get there. The Horus Heresy is what we are basing this project off of. The Horus Heresy is one-part expansion and one-part prequel to the main Warhammer 40k setting, and is the scope of our project. The Horus Heresy is a civil war taking place in the 30th Millennium, ten thousand years before the main setting. We chose this to be our scope for one simple reason and that the Horus Heresy is the book we had on hand and it sounded like a reasonable challenge for our team.

1.2. Challenges

Our challenges are information based. The more information we want to include into the program, the complexity and time required to complete the program grows exponentially. There is a lot of information in Warhammer 40k that must be

processed such as building an army and recognizing factions, unit types and unit upgrades that if we tried to insert this information in the program as well, the program would take us much more time than we have available to us. There are many different connections to every aspect of this game such as faction interactions, unit strengths and weaknesses, and upgrades. Every piece of information that we could put in the program would require much more time to implement.

2. Scope

Our bare bones level of functionality that we would call complete is when choosing a unit, you would see the unit name, index numbers, the unit base stats and point values. Also, the ability to add a unit to a list which ties into the next requirement, the ability to save and then print the list into a text file format. A stretch goal of ours would be the implementation of the point value system that limits the number of units you have in-game and tell you, and stop the selection of new units, when you have exceeded the point limit. If you didnt have this system in place, the players would have no end in the army building and thus create outrageously large armys that would be impossible to interact with in the game. Another stretch goal of ours would be the ability to select the appropriate upgrades and how they affect your unit, update the point value and an index to the page that the upgrades are on. The upgrades are unit specific and have different point costs which could Also another stretch goal would be the ability to read back an already created army to work with it. Allowing you to, instead of creating a completely new army, you can edit your already created army.

2.1. Requirements

2.1.1. Functional.

- User needs to have container to hold information.
- User needs to be able to add and delete information as needed.
- User needs to be able to create units and access related information.

2.1.2. Non-Functional.

- UI that the user interacts with is easily navigated and simple to learn.
- Ensure that the database is accurate to the source material.

2.2. Use Cases

1) A user should be able to open a drop-down menu and double click a button to add a unit.
2) A user should be able to open a drop-down menu and click on a unit to see its stat information
3) A user should be able to look at the textbox to the right to see the contents of your army list.
4) A user should be able to click a button to select number of points available to them.
5) A user should be able to see the amount of points still available to you.
6) A user should be prevented from adding more units if that unit costs more than you currently have
7) A user should be given basic instructions on how to use the program on startup via message box.
8) A user should be able to select an index, and click a button to delete a unit.

2.3. Interface Mockups

3. Project Timeline

2/15/18 —– Presented proposal to the class

2/28/18 —– Initial Database (Unit names, index, point values) finished and undergoing conversion to Access format.

3/2/18 —– Initial database fully converted to Access, UML finished, timeline finalized, design patterns chosen.

3/7/18 —– Text describing stat-line figures and copyright information established, sample list format 1 ready for review. Prototyped classes,

3/15/18 —– Next check in ready, UI demonstrating partial functionality, at least able to access initial database

3/22/18 —– One or two main classes fully implemented and functional, with list builder fully functional with initial database

3/29/18 —— Adding information to initial database to form final database (Unit names, index, point values, composition, statlines), begin altering three complete classes to suit

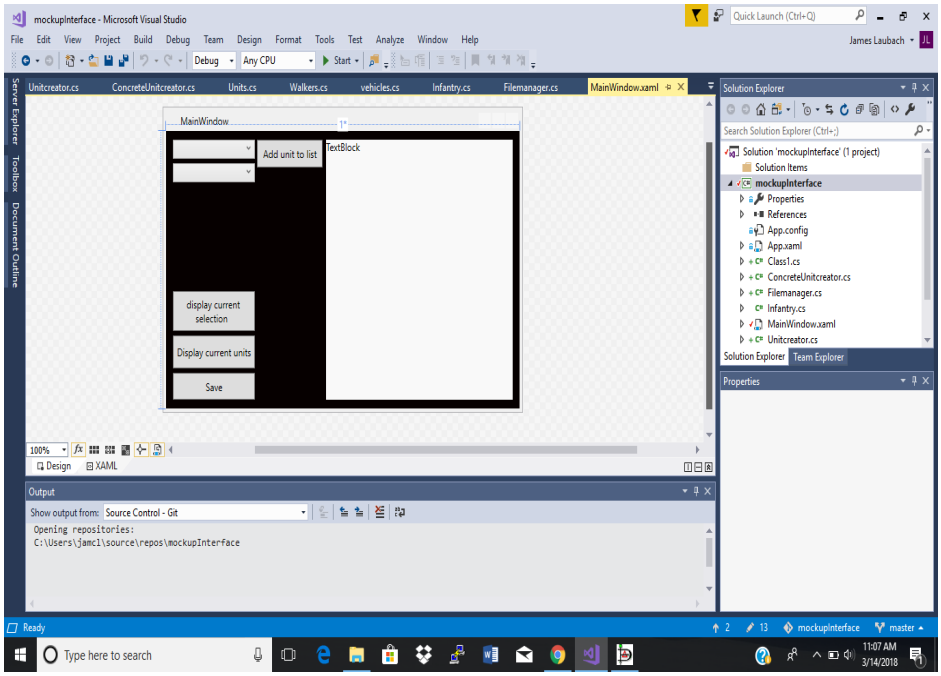

Figure 1. A picture displaying an early mock-up of our UI , in this early iteration the upper drop-down menu would allow the user to select which class of unit (infantry, vehicle, or walker) and then the bottom menu would display the appropriate units of that class. The user could then decide to display the stats of that unit (in the Textbox) and add the unit. The user could then choose to display the current list and save the list.

4. Project Structure

After some research we arrived at the conclusion that we had three basic classes of unit. One of our classes, infantry, also has a sub class where data is displayed differently as well. For our purposes here, the actual differences in data displayed are unimportant. So, we decided to arrange the three main classes in a factory pattern as sub classes of the class Units. The UnitManager class acting as our assembler that the client interacts with via the buttons. The client does not interact with any other class, only UnitManager.

4.1. UML Outline

Infantrys sub class (hitherto known as DualLine) is written as a decorator on the Infantry class that is called to record sub-unit information that makes up various infantry squads. Infantry records the overall squad stats, where DualLine adds the ability to record squad-member and squad-leader stats, both of which are separate and distinct from the overall. The class UnitManager accesses all the other classes. All the Unit classes, Walker, Vehicle, Infantry and DualLine are all accessed by the UnitManager. FileManager is the only class that can access the database but FileManager can be accessed by UnitManager. For all intents and purposes, UnitManager does everything required in this program because it is the central hub to the program.

4.2. Design Patterns Used

We used a factory pattern in conjunction with a decorator to store our distinct unit information on three different classes and one sub class. Infantry,, Vehicle, and Walker act as products called through a UnitManager factory that is acted upon by the UI. DualLine is a decorator that adds the ability to record multiple statlines to Infantry's

5. Results

Our progress in this area was slow due to the time it took us to finalize which goals we could accomplish and which should be out of the scope of our project. As it stands now, we have achieved several of our objectives. We have finalized our UI and have a working list builder. Our work is almost finished with the user being able to read the information of a unit with a single button click. Within our UI and UnitManager, we have the ability to add units, subtract units, display

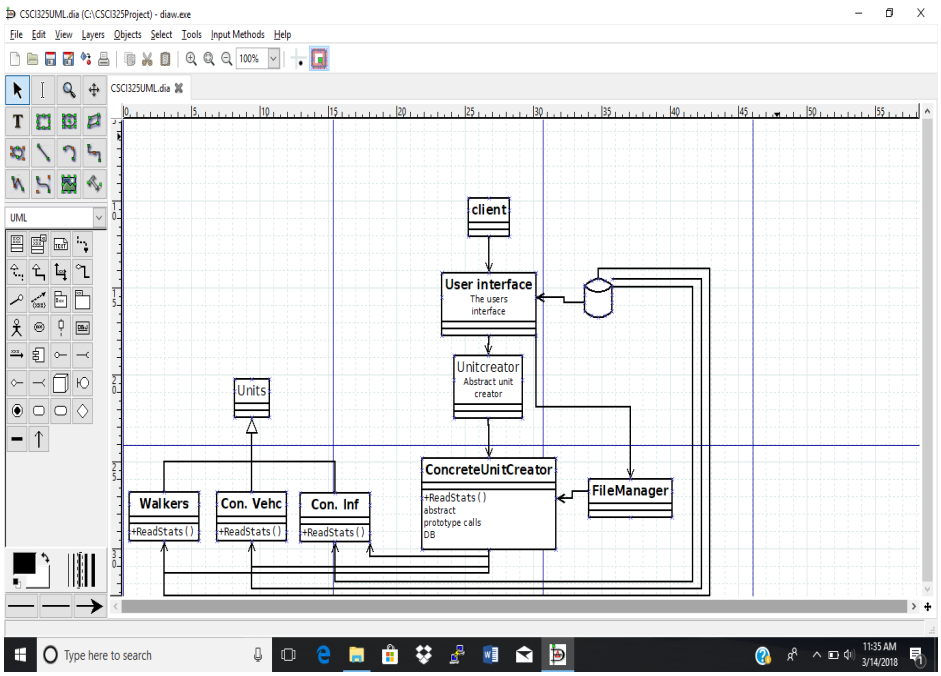

Figure 2. A UML that describes the interactions on a broad level between the client, our user interface, and our various classes that supply functionality

individual unit stats, and to print the entire list of selected units. We had thought that we may make this program compatible with different databases corresponding with different books and thus different factions, however, we do not know how to procedurally create buttons to fit the other books so we dont have to manual write every unit button again. Currently our extra features are the ability to see a textbox with the data of the unit that you selected and your list of units you have chosen to be in your army.

5.1. Future Work

In the future we may try to modify the program to work with the eighth edition rules when they come out as we are currently on the seventh edition. Also as a stretch goal that may not be possible is the ability to create the buttons procedurally to allow us to use different versions simply by reading in the database.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.