Starcraft II Build Order Visualizer

Karl He
University of California, Berkeley
Department of Computer Science
karl.he@berkeley.edu

Saung Li
University of California, Berkeley
Department of Computer Science
shadowcwal@berkeley.edu

Abstract

We describe a visualization tool for analyzing build orders for real-time strategy games, specifically for Starcraft II. We provide an overview of the strengths and weaknesses of related work, and detail the implementation of the tool and the techniques we employed. We then discuss the results of the tool and outline a number of further directions we wish to take for this project.

1 Introduction

In real-time strategy (RTS) games like Starcraft II [8], a participant's goal is to destroy his opponent's assets. To accomplish this, he must gather resources, construct production buildings, research new technology, and produce units to attack his opponent [1]. An essential concept that has been developed by RTS players is build orders, in which players perform the above actions in a linear pattern, managing resources and constructing buildings and units to achieve a specific army by a certain time.

Since RTS games generally provide many structures, units, and technologies for players to choose from, build orders vary widely and have strengths and weaknesses against other build orders. Timing is of utmost importance, as a player must have a certain amount of units to counter the other player's units at any given time [2]. Often, players come up with ideas for what units to produce, and must play through multiple games to master the timing of the corresponding build orders. If there are parts of the build orders players want to change, they must modify them and play more games to test them out. This process can be time consuming, and although there is a large community of players sharing build orders with each other, the formats in which they currently do so have several weaknesses that we will discuss. These formats typically do not allow players to quickly grasp the key timings or strengths and weaknesses of a build order.

To enable players to focus on learning and modifying build orders, we address this issue by developing an interactive visualization tool that reveals the timings of build orders and allow users to compare them against other builds. We focus on developing this visualization for Starcraft II, as it is arguably the most popular and competitively developed RTS game, but we believe our techniques can be generalized to other RTS games.

We first review related work on visualizing Starcraft II build orders.

2 Related Work

By far the most widespread form of representing a build order is via what we will refer to as the textual format. Players generally use this format when sharing it on online forums [3]. Here is one such example of the "4 Warpgate Rush" build order in textual format:

9 Pylon
13 Gateway + Probe sent to scout
14 Assimilator
16 Pylon
17 Cybernetics Core
18 Zealot
22 Stalker
24 Warpgate research
24 Stalker
26 Gateway
26 Gateway
26 Gateway
26 Pylon

Here a number represents a supply count, which is a measure of how many units a player has, and the following action is the building, unit, or research to perform at that supply. Workers are assumed to be constructed constantly unless specified otherwise. This format is concise, as it packs most of the information a player needs to execute a build order in a compact fashion. Players can easily memorize build orders from this, and sharing is a matter of copy and pasting such text. The drawback is that one cannot infer the timings of the build order from this format. Players cannot tell when they would finish the build or what they would have at any given time. This makes it hard to tell how well this build would do against others. Players must play games to test out the effectiveness of the builds, and modifying the build would mean having to play more games to test out the changes.

The game itself provides a visualization of the exact build order that players use in their games (Figure 1).

Figure 1: *In the post-game screen, Starcraft II players can view the initial build orders of each player involved. The exact supply and time that each unit was built is apparent. However, the lack of horizontal alignment and the strictly tabular format makes it difficult to compare two build orders, or easily create a mental image of what is happening.*

Figure 2: *The sc2replayed version of the build order comparison is much more intuitive. Research timings are included and actions are horizontally aligned. However, it still suffers many of the same problems as the built-in visualizer.*

Along with supply counts, this built-in visualization also shows the timings of each action so that players can tell what they have done by a given time. The problem is only the start times of actions are displayed, so we cannot see when they are completed. Additionally, if we want to know what players have at a certain time, we would first have to locate that time and then count everything upwards and remember the actions. This makes the side-by-side comparison of build orders difficult as players need to locate the time they want for both builds and then remember everything that has happened up to that point. Also, this does not

show some of the in-game actions such as transferring workers, dropping mules, chronoboost, and more.

www.sc2replayed.com provides another visualization for build orders, showing builds that players have used (Figure 2).

The main advantage of this visualization is that the timings of actions are aligned horizontally so that we can easily see what each player as done at given times. However, the supply counts are missing, and this visualization suffers from many of the disadvantages mentioned in the built-in one, such as not knowing the completion times, the total counts of units up to a given point, and not showing certain actions.

www.sc2calc.org provides a tool where users can input a build order in textual format and it generates a table of the actions performed along with their corresponding timings, supplies, and resources available [4]. Although this provides more information about the state of the game, it is still hard to visualize all the timings at once, as there is a lot of cluttered text. The completion times provided are hard to analyze, as players need to remember them and look forward in time to see what has happened from start to completion time. The tool also does not provide a way to compare two build orders at once.

It is worthwhile to note that of the 3 tools described, the first two pull their information from analyzing replays of actual games. The sc2calc.org tool instead allows users to manually specify the build order using an input system derived from the textual build orders previously described. This system allows for much more interactivity, and shared many aspects with the new tool we were planning to develop. We ultimately decided to build upon their tool rather than rebuild the system from scratch.

3 Methods

Our goal is to convert build orders from a textual format to a visualization that addresses the above problems. More specifically, we want to expose the start and completion times of all actions and make it easy to compare what two different build orders have at any given time. We want users to be able to save and access builds in textual format so they can generate visualizations from them at any time. This maintains the portability of text, as users can still share build orders via copy and pasting.

We strongly believe in the importance of the community in developing build orders, and therefore the importance of being able to easily access the tool. Using the web and Protovis [5] as a medium for our work was the best fit to these goals.

3.1 Input

Since the textual format is most widely used and is easily shared, we develop this tool to take in as input a build order in that format. From here, calculations need to be done to determine the resources that the player has, start and completion times, supply counts, and actions performed. Comprehensive work has already been done on this by Jasper A. Visser, as displayed in his build order calculator tool at www.sc2calc.org.

Impressed by Visser's work, we decided to use it as the parser for our system. As sc2calc.org does not have any sort of API, we instead utilized a PHP page to capture the results of POST requests we send to his tool. When the page generates the table, we fetch the html, parse all the information in the table, encapsulate it in JavaScript data structures that can be easily used by our system. We can now use this data to generate visualizations, using textual build orders as input.

3.2 Saving build orders

We create a tree to visualize all of the stored build orders. The root is the name of the tree, and it has three children, one for each race. The children of a race node are the build orders in textual format corresponding to that race. The name of a build is the name of the node. A user who is searching for a specific build can toggle the race nodes to hide irrelevant nodes and look for the build's name. Hovering over the name of a build shows the build order in textual format next to the cursor to remind users of what it is. Clicking on a build from the tree will highlight it red and fill in the input box with that build and name. Users can then perform visualizations or modify the build and overwrite it. Users can also choose to delete the build from the tree.

Note that we did not implement a database for saving since we are focusing on the visualization aspect of the tool, so refreshing the browser will erase saved builds.

3.3 Unit cheat sheet

One weakness of using textual build orders as the input system is the verbosity of the input method. Visser's system in particular is fairly strict in terms of the input, and does not allow abbreviations of any sort, nor does it make any assumptions about actions the players take that may seem natural. An example of this can be demonstrated by the line "14 Assimilator, and put 3 workers on gas". The Assimilator is a structure of the Protoss faction which allows players to gather the resource vespene gas. It is typically not explicitly stated that a player must order his workers to harvest the vespene and is rather implied by the fact that the Assimilator would not be constructed without the intention of using it.

Despite its weaknesses, textual input still remains the best way to transmit build orders. While we cannot wholly solve the weaknesses, we can alleviate the need to remember exact spellings, as well as the need to input large quantities of text. We accomplish this by providing unit portraits categorized by race that users of our tool can click on to quickly input units into our system. This could be further extended to also perform actions such as transferring workers and swapping add-ons, some of the more common and most wordy actions that can be expressed in Visser's syntax.

3.4 Build order visualization

Our approach to displaying a build order is reminiscent of a Gantt chart, which is used to illustrate the start and end dates of the elements of a project [6]. Our display consists of a vertical timeline in-between the two build orders and time bars representing the start and completion times of performing an action. Our major deviation from how a Gantt chart is displayed is that a Gantt chart organizes the bars into projects. Our visualization requires no such distinction, and it in fact would become a visual impedance to have a new column for each new action that was performed. We instead chose to use the minimal amount of columns necessary. The maximum number of columns we use is the same as the maximum number of actions that are simultaneously performed in the build order.

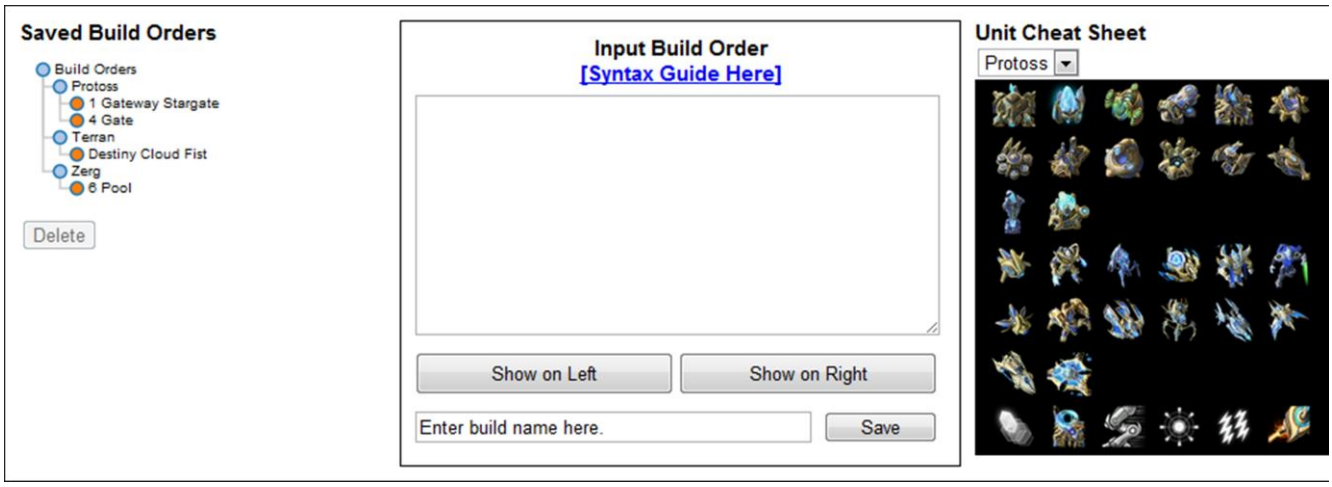

Figure 3: *The input box for our build order visualization tool.*

Figure 4: *The visualization for our build order visualization tool. This example shows the 4gate build versus the 1-1-1 build.*

Additional elements we felt were important to display were the supply counts when each action is performed, as well as cumulative information at any given time in the build order. We felt it was important for supply counts to be visible at all times, since they are a vital part of how build orders are remembered and understood by players. We decided to display the supply count directly on the time bars. Although this adds some visual clutter, the utility it gives is far greater than the cost.

The cumulative information we wanted to display were the units each build order has produced, as well as some measure of the economic power of each build order. We chose to incorporate this via an interactive hover-box that followed the mouse's position on the timeline. This gives intuitive context to the current units and resources being displayed by the hover-box, without unnecessarily cluttering the display.

4 Results

4.1 Inputting Build Orders

The textual format of builds used as input for our tool must be the same as the one used in sc2calc.org. That site provides a syntax guide for the input format [4] that can be used for our tool.

To assist users in inputting units and buildings as text, we provide images of all of them in the "Unit Cheat Sheet" section. Clicking on them will input their corresponding names with spaces before and after them into the end of the text box. This can assist players with remembering and inputting the names of units, and can speed up the process of inputting a build. The drop-down list allows users to toggle between the different races. Organization by races is most logical as no build order will utilize multiple races.

After inputting a textual build order, the user can select "Show on Left" or "Show on Right" to generate a visualization of the build on the bottom left or bottom right, respectively. Doing this will not save the build order. To do so, the user can input a name for the build and save it so that it can be retrieved again. If an input is entered incorrectly, the user is notified of the problem.

4.2 Saving Build Orders

The "Saved Build Orders" stores some example build orders as well as any build orders saved by the user of the tool, stored in a tree structure organized by race. Build orders are saved by name, the contents of which can be revealed by hovering over the name of the build order. Selecting a build order loads it into the input field, and also allows the user to delete the saved build.

4.3 Reading the Visualization

To prevent too many parallel rows of actions, the time bars are collapsed into the minimum necessary columns. Each time bar includes an image of the unit, building, research, or other action taken, and the supply count is displayed below the image. Hovering over the image will display the name of the action or unit in a tooltip. The time bars are color-coded according to whether the action taken is producing a worker, unit, or building, or performing another action such as researching technology or transferring workers. We used the selected colors from Colorbrewer's qualitative-set1 color scheme so that the text and images are still readable and the colors are easily differentiated. By color coding these categories, players can easily focus on one category and compare actions and associated timings in it between the two build orders. A legend for the colors is unnecessary, as players can immediately recognize what they stand for. These time bars allow users to see what is completed and currently in production at any given time.

In the center of the visualization is a timeline representing game time. The timeline is aligned to both build orders displayed, making it easy to compare the timings of each build. When the user hovers over the timeline, a box displays under the cursor showing the time and images of the resources, supplies, units, buildings, and research completed up to that time. The images are organized so that units appear in the first row, then buildings in the second, and then all other actions such as research and transferring workers in the third. Images are shown multiple times according to how many of that action is taken or unit is controlled, so that users can easily compare strengths and weaknesses between builds. The image of workers are displayed once, with a counter next to it, since most builds generally produce many workers. This hover-box feature enables users to see everything that each build order has at any given time.

5 Discussion

We did not undertake a formal usability study of our system, but we allowed several StarCraft players to try out the tool. An important test to perform would be to see how long it takes for players to understand and modify particular build orders using our visualization. We can then compare these times with similar-level players who use other formats such as the textual one. The main difficulty in carrying out this test is searching for players with similar experience and who do not already know the build orders being shown.

Upon sharing our prototype with the community, reception to our system has generally been positive. The most common problem appears to be in initially understanding how to use the system. Even StarCraft players have difficulty using Visser's build order syntax, which uses several symbols and phrases that typically don't appear in textual build orders, and are rather explained in prose beneath the build order. This difficulty is alleviated by our cheat sheet of units, which generates the text necessary for an action. After initial criticisms, we added a direct link to the syntax guide for those unfamiliar with the textual format used. Understanding how to read the visualizations produced also remains an issue, although the several StarCraft players we spoke to quickly picked up how to analyze them.

Upon overcoming the learning curve of inputting build orders, users don't appear to have any complaints about the way the build orders are displayed. Users had highly positive comments on the hover-box depicting what each player has at any given time. Visualizing start and complete times and the color coding of timeline were also found to be both helpful and aesthetically pleasing. The main problem is when users want to modify a build order, they would have to deal with the syntactical issues of the input format again and in-game constraints such as supply limits and prerequisites for buildings, units, and research. However, coming up with build orders and modifying them to perfect timing has always been a difficult task, and overall our visualization system acts as a beneficial tool for analyzing the success of build orders with respect to others.

6 Future Work

Our initial user studies seem to indicate that our tool's major weak points are its initial learning curve and the lack of apparent utility. Future work on this project is therefore centered on giving more immediate utility to the user, as well as making the system more intuitive.

6.1 Workflow

Many users were initially confused when presented with our system. We hope this can be at least alleviated by making the workflow of our system more apparent. This encompasses using the saved build order tree, the input form, and the left/right display actions. Adding more visual cues and tooltips will hopefully remove the need for lengthy explanations or manuals on the tool.

6.2 Clustering

When analyzing a generated visualization of a build order, it is useful to be able to quickly focus on a particular category of

objects such as workers, army, buildings, research, and actions so that they can be easily compared to another build order. We partially addressed this issue by color coding these different categories. However, an improved algorithm which groups similar objects by category could greatly improve readability. For example, in the visualization of a build order, workers could be grouped in the first few columns, units in the next few, then buildings, and then research and actions. Doing so could allow users to easily locate objects within a category. Filtering could be another option, though generally users would still want to have a sense of other categories even when focusing on one of them.

6.3 Dynamic Edits

Currently, users can edit a build order by modifying the text, and then re-rendering the visualization. An alternative feature where users can directly manipulate the timeline to add and remove objects would make it easier to tweak a build order while comparing it with another one in order to perfect a timing. The ability to drag objects into and out of the timeline would improve interactivity and provide the user with greater flexibility in modifying and visualizing build orders.

6.4 Sharing

An important aspect of our project is the ability to bring existing build orders into the system so that users can retrieve them at a future time. We have provided an interface for saving builds, but have not implemented the database for permanent storage due to time constraints.

A further step would be making it easier to share build orders saved into our system, which would allow us to feed them back into the community. Currently, users can copy and paste the text of the build orders, which can then be fed as input for our visualization. Perhaps we can allow users to share URLs that direct them to the visualizations themselves, and allowing them to export and import their saved build orders. Doing so can help the community effectively communicate and develop new build orders.

7 Conclusion

In this paper, we described a visualization tool for analyzing build orders for Starcraft II. We reviewed the strengths and weaknesses of related work, and detailed the implementation of the tool and techniques used. We then discussed the results of our tool, and, finally, we outlined the several future directions we plan to take this project into. Since most real-time strategy games have similar fundamental aspects such as structures, technological advancements, attacking units, and building, resource, and timing constraints, the visualization techniques we used here for Starcraft II can be generalized to other such games.

Acknowledgements

We would like to thank Jasper A. Visser for creating a tool for calculating optimal timings of build orders and writing the syntax guide. In retrospect, replicating even a lesser imitation of his system would have been impossible under our time constraints. We would also like to thank Maneesh Agrawala and his Spring 2011 CS294 class for guidance and Blizzard for creating great games and fostering a wonderful community.

References

1. Real-time strategy. http://en.wikipedia.org/wiki/Real-time_strategy

2. Build order. http://en.wikipedia.org/wiki/Build_order

3. Team Liquid. http://wiki.teamliquid.net/starcraft2/Strategy

4. Sc2calc. http://sc2calc.org/build_order/

5. Protovis. http://vis.stanford.edu/protovis/

6. Gantt chart. http://en.wikipedia.org/wiki/Gantt_chart

7. Colorbrewer 2. http://colorbrewer2.org/

8. Starcraft II. http://us.blizzard.com/en-us/games/sc2/