Real-Time Strategy Streamlined & Enhanced User Interface (RTSSE UI)

Bryan Blackford, Andrew Gee, Tarin Smith, Adrien Young
UC Santa Cruz
{bblackfo, agee1, tamasmit, akyoung}@ucsc.edu

1. INTRODUCTION

In the heat of battle, with forces on both sides wearing thin, the need for reinforcements becomes clear, but the battle still requires attention to maneuver troops around for tactical advantage (micro-management). In a typical real-time strategy game (RTS), the player would have to click the map to return to base, click on their unit producing buildings, queue up reinforcements, and put those units into a group before sending them into battle. RTSSE UI is a set of widgets for the RTS "Conflict Terra" which allows the user to quickly call for reinforcements, without leaving the battlefield.

RTSSE UI is a proof-of-concept system, which could eventually be used by new and experienced RTS players to streamline and make their games more battle focused, which is where the fun and tactical possibilities of an RTS shine. For new users to the system, a tutorial will introduce the usage of the system. Along with the production grouping, RTSSE UI also features movable and resizable windows, for custom user layouts. Our system was implemented using the Spring SDK, using Lua scripted custom widgets and a select set of existing widgets, and may work with Spring games other than Conflict Terra.

A screenshot of the final RTSSE UI with production group window at bottom right.

2. REQUIREMENTS GATHERING

Competitive analysis:

A picture of the Starcraft II (SC2) UI

A: An inspirational production queue, showing all production for both players.
B: Tiny text that is unreadable over YouTube in standard quality.
C: Tabbed unit groups, created by player assignment in SC2. We'll automate this with production groups

Data gathering method:

Method: "Behavioral Mapping" and "Shadowing"
Story: we were able to determine that several of our target personas were more comfortable with an RTS experience that was more battle-centric than micro-management. While this is an ideal that encompasses the entire game design project, the interface is certainly a place to start conveying this to the player
Results: The user reacted positively to consolidating/hiding UI elements for minimal/simplified interface (revealing more battlefield) reducing screen-clutter, as well as the idea of making production queue accessible from anywhere (no need to return to base to manage queue).

Method: Extreme User Interview
Story: I asked a friend of mine that has played Starcraft II about what he liked about the current Starcraft UI, what he doesn't like about it, and what he would like to add to the UI.
Results: The data I got revolves mostly around the macro management of the game. He has a tendency to switch his camera view back to his base to check if his production queues are correct. He suggested adding some visuals that shows what units are in production. In general, he would like the macro management to be more accessible during battle so multitasking is less of a burden. He also doesn't enjoy having to hotkey new units to an existing group. He finds it repetitive and is sure there is a better way to assign new units to existing groups than to manually hotkey them.

Method: Narrative
Story: I had a friend play an RTS (Age of Empires 2) while he talked through his thoughts. I asked him to focus on the game's GUI, controls, and economic strategy.
Results: The useful data from my observations largely pertains to automation and multitasking. Early on, he noted how automatic rally points reduced the need for him to micromanage his workers while he was busy with something else, and that the age research indicator was a welcome addition in the game's expansion, as it made it easy to track his progression towards important game progression milestones. He also noted other automation improvements in the expansion, such as automatic farm replanting (previously, players had to manually order workers to replant) and automatic worker job assignment. He explained that these features allowed him to concentrate on the fun aspects of the game without having to repeat the same rote mouse clicks to run his economy.

Method: Guided Tour
Story: Watched Youtube videos of Starcraft 2 replays to determine what improvements could be made to the interface. The UI for replays displays some information that is normally private to each player, so the viewer can see what is going on with both players at the same time. Any UI changes should be reflected in the replay interface. Also, the UI elements were sometimes hard to read over YouTube.
Results: A more low-resolution-friendly replay interface would be an improvement. Any UI changes in the player interface should be considered for addition to the replay interface. Hidden information for both players should be visible in replays.

3. REQUIREMENTS ANALYSIS

Personas.

Method: After interviewing some intermediate players of the RTS genre, I created this persona to represent the users of that category.

Eddy Wong, 20, UCSC Molecular Cell Development major.

Has experience playing a lot of popular RTS games such as Warcraft 3 and Starcraft 2. His time playing RTS games has gone down since his current laptop cannot run a lot of high-end games. He resorts to playing games on the lowest graphical settings when he has some leisure time.

Attributes:
Dislikes rallying system in terms of hot-key assignments in most RTS (repeat hot-key assignment every time you want to add a unit to group).
Prefers to micro-manage units in battle instead of managing economy/unit production.

Key Goals:
Have unit production visible while managing a battle.
Minimize the amount of times a player has to use hot-key assignments.

Method: After asking some potential users who played more casual games about the RTS genre, we gathered the data to create this persona. During the questioning, we explored why they shied away

from the RTS genre and what might convince them to try the genre.

Joe Schmo, 17, High School student

Normally plays simple games like Angry Birds. Rarely steps into competitive features of a game. The multiplayer games he usually plays are the ones that involve cooperation.

Attributes:
Uses computer regularly, mostly for school and web surfing.
More familiar with console game controls.

Key Goals:
Wants to get into the RTS genre but hasn't played any games with that much complexity (his limit so far is Warhammer 40k: Space Marine, which is actually a pretty simple game)

Scenarios
1. Group factories: To enable the functionality for the next two scenarios, the user first assigns their unit factories to production groups.
2. Production during battle: In this scenario, the user is managing battlefield maneuvers and they need to build reinforcements. This is a typical behavior for intermediate players and above, and it is important that this be done efficiently.
3. Grouping recently built units: After producing some units, the user wants to group the units together so the group can be selected with a single keypress. This is a frequent behavior by users at all skill levels.

Use Cases
1. Group factories: User clicks factory to group, user clicks "+" beneath the desired group in the production group window.
2. Production during battle: User clicks the number on the production group window for the group which needs reinforcement, click units on build window to queue reinforcements.
3. Grouping recently built units: Units produced from factories which had been grouped are added to the group automatically.

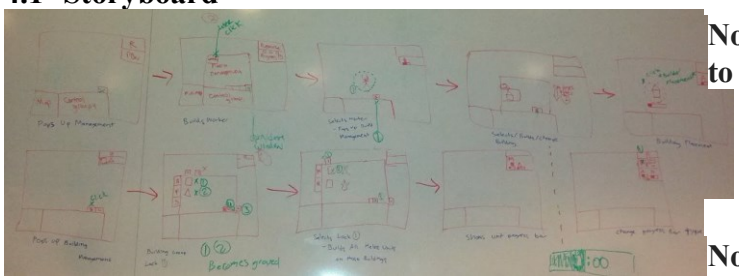
Requirement summary.
Functional:
- (high) Shared production queue window
  - Helps to group all production functions in a single window.
- (high) Building grouping - produced units group
  - Minimizes the need to hotkey units to specific groups.
- (med) Production queue accessible without need to return to base
  - Minimizes need to switch the camera view from base to battles.
- (med) Progress bar in queue
  - Provides confidence that production of units are correct.
- (low) New replay interface elements for new game functions
  - Helps users learn advanced techniques from pro players.

Non-Functional:
- (high) Game feels more battle-centric than micro-management
  - Provides a most exciting feel to the game by emphasizing combat.
- (high) Accessible macro-strategy
  - Lessens the time spent managing resources and more time managing battles.
- (med) Consolidating/hiding UI elements for simplified interface (revealing more battlefield)
  - More screen space for the user to gather information from.
- (med) Decreased multitasking load (automation, combination)
  - Attract players that are looking for a less complex game in terms of RTS.
- (low) UI elements readable in mid quality YouTube video

- Lets players watch the replays in detail.

4. LOW FIDELITY PROTOTYPES

4.1 Storyboard



Our storyboard was developed by 3 gamers who ranged from new to experienced with RTS games. We first identified the key features of the system, and with the context of an existing RTS UI, Starcraft 2, they drew up the screen frames that illustrated a process for achieving our primary requirements. The frames are annotated with labels for the windows and where user clicks occur. They are presented as a sequence of events with abstract symbols representing the game assets (we had not yet chosen the base game for our mod).

4.2 Inspection

We used the heuristic evaluation technique for inspecting out user interface. The reason we chose this over the cognitive walkthrough was to determine what features we might be missing after our initial design phase as well as how accessible the interface was. Our focus for the RTS interface was to make certain aspects of the game to be easier to use, this led to a conclusion to use the heuristic evaluation in order to gauge how easy and how complete the interface was.

Usability Problems:

No clear exit buttons from menu

- Add a button to the menu or replace the Unit/Building tabs on the bottom with an X after the menu is open.
- When tabs are closed you see [Building][Units].
- When Unit tab is open you see [Building][X] or [exit].

No cancel unit selection in build order.

- Can just have the player select the unit in the progress bar to remove them from build order.

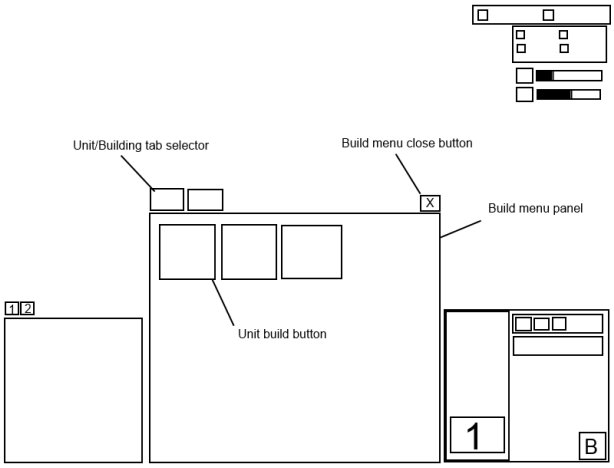
No hotkey options to navigate through menu or to build units.

- Have built in hotkeys for player and have the option to change those hotkeys in a menu.

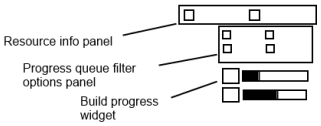
No help button to explain the purpose of the menus and the progress bar.

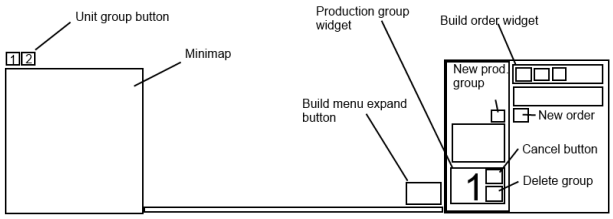
- Include a tooltip when the mouse hovers above a button.

4.3 Wireframes



The above wireframe displays the build menu. We included tabs in the menu to help the user filter through the different buildings. The menu would also pop up pictures on units that would be available to build on the main screen. We left the tabs out of the build menu in the final build due to how clunky it would have been for the user to always be clicking on small tabs to go through a build menu.





This wireframe entails everything else outside of the build menu. We had included several features in this wireframe that are not apart of the current final build. Some of the features that are not present include the progress queue filter panel, unit group buttons, build order widget, and the build menu expand button. The reason why is because the features were superfluous to how our game is structured and that the size of those features were too small and making them any bigger would have taken away a lot of valuable screen real estate.

5. USER TESTING

Method:

The method we used to test our user interface was to just have a small number of users play two games of Conflict Terra. The first game would use the standard interface provided by the game and the second game would be using our interface's features. This way we can have the users compare and contrast their experiences and see if our interface was an improvement over the standard interface that was provided.

Users.

Edmund Kong, 20, Male, intermediate player of StarCraft 2.
Danny Wong, 21, Male, intermediate player of StarCraft 2.
Ryan Dawson, 21, Male, beginner player of RTS genre.

Testing setup.

We had the users play the game in a quiet room on campus in the afternoon. We brought three laptops that already had the game install and proceeded to set up a match for a three player free for all match. After the two games were finished, we gave the users a sheet of questions to answer about the user interface. Some questions we asked involved how difficult our interface was to learn, whether they preferred the original interface or ours, rating the individual interface features, and much more. We did our best to leave the users alone when they were playing and just observed how they used the interface.

Analysis.

We analyzed the data by looking at the qualitative data that the users gave us. We also gathered data from the comment boxes that were at the end of our questionnaire. The users seemed to have liked most of the features but they also liked the idea of having the option of moving and resizing the interface windows.

Results.

Users seemed to have really liked the option of freely moving around the windows to their preference instead of having us anchor windows in certain parts of the screen. In light of this new data, we decided to have all interface features to be moveable by the user so they can decide where they would prefer the windows to be.

6. FINAL VERSION

The final version added:

- Resizable windows with automatic button rearrangement
- All windows (including map and resources) movable
- Fixed bugs (error output)

The final version is different from the high fidelity because of the official feature that all interface windows can now be moved. The high fidelity use to have the map and resources bars anchored to corners of the screen, now the user can move those around as well as resize them to their liking.
If we had more time to develop the interface, we would have added a button that would have the build menu appear and disappear to clear the screen

space when you don't want to see it. We also wanted to have visible tabs on the screen to help users select grouped units. We might have also included improved video replay interface elements.

We managed to fulfill 4 out of the 5 functional requirements that we set for ourselves. The lone functional requirement we didn't meet was a low priority requirement that involved the replay interface. The non-functional requirements were pretty much the same as the functional requirements with the only thing not accomplished was implementing the features pertaining to replay videos.

Another screenshot from RTSSE UI showing a different arrangement of windows from the picture on page 1.