Small Screen, Big Strategy: Bringing Real-Time Strategy to Mobile Devices

Edward Bramanti

December 6, 2013

1 Introduction

In the past few years, mobile operating systems have become a hub for both independent developers and game studios alike to publish video games that their consumers can hold in the palm of their hand. Mobile game development once used to be a frontier; now, it seems as though a new game comes out every few weeks, and they battle for the addiction of their consumers. While many different genres of mobile games have been produced, from tower defense to adventure to even massively multiplayer, real-time strategy remains a genre mostly untouched by mobile game developers. One's initial reaction may be to believe that lack of interest in real-time strategy (or RTS) games is responsible for few games being published in that genre. Nevertheless, as this design will outline, a lack of intuitive controls and user interface has prevented the genre from thriving on mobile. Many developers have been met with interface design challenges. One example of this is a result of lacking hardware devices outside of the touchscreen (such as a mouse and keyboard, or even a controller with buttons). My user design will show how mobile strategy is the future of challenging, yet rewarding mobile gaming. Empowering the users by utilizing every inch of screen is the key foundation to building a successful mobile RTS interface.

2 Design & Layout

To successfully approach designing a user interface for mobile RTS, my approach will be to play to a touchscreen's strengths, and conform to design principles in mobile operating systems.

2.1 Unit Selection

To select a unit within the proposed design, a user will use just a finger and tap on whatever unit they want to control. While this is simple enough, the challenge arises when a user wants to select multiple units, because of a lack of draggable input using a mouse cursor. I decided to take an approach similar to Halo Wars. Halo Wars employs a reticle, and when a user holds down the A button on the Xbox 360 controller, an overlay appears that shows units in the selection circle. Since the user's controller is the touchscreen itself, the same action will occur when one holds down their finger on the screen. After a small delay, the selection circle will expand from the held touch input, and allow multiple units to be selected at once.
When a unit is selected, stats will be displayed about them centered at the bottom of the screen. If multiple units are selected, they will be displayed in groups by type in the top-left corner of the screen. This will allow a user to both view attributes below their selection, and view their selected units in a status bar at the top of the screen (which, in mobile OSes, is always located at the top of the touchscreen.)

2.2 Minimap

One of the most important information displays to anyone playing a real-time strategy game is the minimap. The minimap is incredibly useful to see where units are located, and where friendly units are deployed. It allows for users to navigate the map quickly; by selecting a location on the map, it allows the camera to snap quickly to that location on the game map. The minimap also provides the user with a way to see where enemies are attacking, and "pings" information to the user when enemy attacks occur.
The biggest challenge of implementing this on a mobile device, as opposed to a computer, is the size of the smartphone screen. I believe that the minimap is essential for the user interface in the RTS genre, but I also believe that cluttering the camera view of the map with too many tooltips will make it difficult for the user to quickly issue commands to their units. Therefore, I propose a two-size minimap. In a normal view, the user will have a very small glimpse of the minimap, as pictured above. It will show their current camera view on the minimap, and a very small representation of unit locations and map. To conquer the issue of screen space, a tap gesture will be employed on the minimap. When a user taps the minimap once, it will cause the minimap to go full screen. This minimap will allow the user to see a bigger, "computer-sized" minimap as opposed to providing a static, small minimap that is close to useless anyways. This will allow a user to see everything going on, and their current camera location behind the overlay. To exit the enlarged minimap, the user will have two options. They can either close the enlarged minimap by selecting outside of the overlay to dismiss. The other option would be to select an area of the map where they would like the camera to snap to. This will allow for the snap-to-location camera functionality that PC RTS gamers have, while scaling the experience to work within the confines of screen space and touchscreen input.

2.3 Unit Movement

Now that unit selection and the minimap interface has been explained, unit movement is based off of a lot of the same principles. After successfully selecting units on the map, you can use either the minimap overlay or within your camera view to move your units. With a tap using either of those methods, your units will be able to move. This will allow unit movement to be simple and easy to remember. After a tap, while units are moving, holding a tap in a different location will allow a user to chain movement commands together, much like Shift + Right Click in Starcraft II.
To attack either a group of units or a base, a single tap on the enemy object will cause an attack command to be issued.

2.4 Camera Control

For camera control, I took inspiration from both iOS and Android Design paradigms. Pinching with two fingers

3 Usage Scenarios

3.1 Creating a Real-Time Strategy Game for Mobile Devices

3.2 Companion App: RTS/FPS Hybrid between Console/PC and Mobile Devices

4 Design Rationale

4.1 Unit Selection

There are a few mobile RTS games on the market today. One notable example is Starfront: Collision, which is available for iOS devices. Starfront's approach for selecting units, or multiple units, is to use a pinch, or two-finger touch, to select a group of units. The biggest problem with Starfront.

4.2 Camera Control

To rationalize my design for camera control, I will address two examples: one illustrating why Starfront: Collision's implementation makes camera control difficult for the user, and mobile OS user interface control for controlling the "camera" of a webpage in a mobile browser.

In Starfront, users are provided with a slider on the right side of the interface, which allows for users to zoom in by sliding up, and zoom out by sliding down. Nevertheless, as mentioned in the Unit Selection rationale, it directly is in conflict with what the gesture does across the OS. A user will use the two-finger pinch gesture, and instead of zooming in, they will be met with a unit selection box. This app is an example of a failure in implementing direct manipulation that corresponds to what most of the mobile OS does in regards to that gesture.

A known paradigm in mobile operating systems is the two-finger pinch gesture; for example, in the Google Maps application on

5 Usability Analysis