CarbonShopper
Interactive Prototype Report

Dennis Doan - User Testing
Anton Osobov - Developer
Phill Pasqual - Project Manager
Neil Rhoades - Design
Kent Wu - Documentation


Problem and solution overview

Protecting the environment is quickly becoming one of the most important topics of today’s world. Yet while reducing greenhouse gas emissions is a must, people continue to follow environmentally harmful habits for reasons such as ignorance, laziness, and the inability to see the impact they are making. CarbonShopper helps users make the most environmentally friendly choice while shopping by providing information about resource use, comparing it to similar products, and helping users keep track of their purchase history in order to improve over time.

Tasks

- **Simple**: The simple task we expect the user to perform is to pull up the information about a particular item. This information is automatically loaded and displayed in the air via our augmented reality technology when the user makes physical contact with a product. A window containing the carbon score, travel, and product information of the item is displayed on a translucent background next to the item.

- **Medium**: Our medium task is the comparison of two different items that the user may be interested in purchasing. This is accomplished in one of two ways, either when the user makes physical contact with two comparable items, or when the user manually selects the compare-to function in the item information screen. When either prerequisite is fulfilled, a bar graph is projected on the ground that gives the user a fast, visual comparison of the two products.

- **Complex**: Keeping track of the user's carbon score is our complex task. By selecting the page from the main menu, the projection display a line graph created from past shopping experiences. Individual nodes would give users an idea of their carbon score during the shopping trip of a specific date, while the overall graph shows trends over time. This provides the user with a clear picture of the changes in their shopping habits over time.

Revised interface design

Usability testing is important to a product, especially so when it is based on a technology that does not exist yet. We learned that what was obvious to us, designers who have spent hours thinking about the constraints and possibilities of augmented reality, may not be clear at all to
first time users. Unfortunately, it soon became apparent that our design was seriously flawed and in dire need of revisions.

The first iteration of our design depended heavily upon physical contact. Namely, the only way to get information about an object was to physically touch it, and to compare two objects meant physically touching both. While we expected this to be intuitive, first time users were consistently confused about how to accomplish these tasks. We addressed this issue by implementing multiple ways to perform the tasks. For example, we added a “compare” button in the item information page, allowing users to manually choose to compare an object with another one without making physical contact with the second item. The compare feature is still automatically loaded when the user touches two comparable objects.

*The user may now compare items without physically touching both*

Another big problems we faced was the fact that our application was not based upon an existing platform. When users were unsure of how to accomplish a given task, many times they resorted to pushing or swiping at the item, the air, or themselves. Essentially, they expected to mirror motions they would use on a tablet over to the AR glasses. To deal with this, we made the navigation of our UI more intuitive. Buttons, sliders, and tabs all give CarbonShopper a mobile app feel, even though it is actually based on an augmented reality vision system. In addition, this helps users learn our program via recognition instead of recall.

Lastly, our low-fi prototype provided users with too much information. In addition to the overall Carbon Score, the user was bombarded with a slew of subcategories. In fact, at one point of our
design the user was given over 20 different numbers, each documenting a minute part of the product’s carbon footprint. Needless to say, this easily overwhelmed our first time users. As a fix, we chose to streamline the Carbon Score down to a single number that was easily visible on the item information page. Furthermore, the rest of the UI was completely reworked into a minimalist, more aesthetically appealing design.
Prototype overview
How the tools helped
We did not use the given tools to prototype our interface. The intended functionality of these tools did not fit our design of augmented reality because these types of interfaces are not exactly common yet. The interface had to be constructed using more robust tools such as Adobe Flash and Adobe Illustrator.

How the tools did not help
Our interface is extremely unconventional in that it is expressed through a medium that these prototyping tools were not specifically designed to create. While they may be useful with prototyping web pages or windowed interfaces, we found that they did not provide us with the functionality that we desired for our application. It would be possible to create a prototype in Justinmind Prototyper or Flairbuilder and import it to Adobe Flash or Adobe Illustrator (our preferred tools), it was faster for us to create it along with the implementation and skip the step of creating it in a prototyping tool. A desirable feature that the prototypers had was that they had templates or stencils that made it fast to create attractive interfaces in minimal time, our interface does not need to use conventional buttons or stencils since it is not intended to be used on a mobile device using iOS or Android.

Overview of implemented UI. What was left out?
The final UI implemented in our medium-fidelity prototype saw much change from its low-fi predecessor. In general, everything was more polished and the design significantly cut down into a sleeker, more minimalist design. One of the most notable
parts cut from the final design was the amount of data presented to the user on the item information screen. This allowed us to not overwhelm the user, especially those new to the program. In addition, We did not implement the glow feature that highlighted environmentally friendly items. This was due to fact that the glow simply impeded the user’s vision far too much, and was more of a distraction.

Wizard of Oz Techniques
Our prototype is heavily reliant on ‘Wizard of Oz’ techniques. Although our interface is fully interactive if you use mouse, our interface is reliant on augmented reality glasses - a technology that is still in its infancy and unavailable to us easily. To simulate this interaction, we will use a projector to project the interface onto the surfaces that our augmented reality counterpart would use. A ‘Wizard of Oz’ would stay behind the curtain, so to speak, to update the interface using a mouse to mimic where the users finger would point to or press. Although in real time it would appear that the interface is being used by pressing on the projected image, someone will be copying the finger gestures using the mouse.

Hand-Coded Features
Another feature that is infeasible to program in time for our interface is the vast database that would be required to access all the information that our product would use. Instead of wiring our application to a database or an API that may lead to complications and much more implementation time than we can afford, instead we were hard/hand-code in the items that we will be expecting the user to interact with. In addition, our interface will be unable to parse visual data the same way that augmented reality glasses would be able to. Ideally, when a user picks up an item, the augmented reality glasses should be able to parse some visual data, such as a bar code or QR code and be able to present the user with the data that they desire. However, our prototype has to be done without access to augmented reality technology, and visual recognition technology is still very unstable and exceedingly difficult to implement in the time frame given. Thus, we will have to hand-code in the items that the user will be interacting with.

Prototype screenshots