CS6750 - Assignment P3

Nolan Piland
njpiland@gatech.edu

1 QUESTION 1

1.1 Principles & Invisible Interfaces

Our principle of affordances can be described as an interface design having a property which, by nature of its design, tells you how to use that interface. This is a strong statement, "by nature of its design, tells you how to use that interface". This implies that the design is intuitive and is something that doesn't need to be explained. By developing a design which is intuitive, the user shouldn't spend any time thinking about how to interact with the interface, helping establish an invisible interface.

Consistency is yet another principle which lends itself to developing invisible interfaces. Consider a button with an "x" icon. You most likely thought of this as a "close button". The close button is an example of something which has become so consistent across computing platforms that we blindly click on them expecting to close content. By maintaining this consistency, we're able to establish behaviors that the user doesn't have to reevaluate when presented in a different context, lending itself to building an invisible interface.

Integrating mappings into a design also helps to build invisible interfaces. Mappings help establish relationships between elements in a way that is familiar to our user by using real-world conventions in a natural logical order. Doing this, we help our user establish a mental model of our interface which matches real-world concepts. An example of such a mapping is a video game character moving in the direction a controls stick is tilted. In establishing this relationship between the control stick and the character, the user is able to move the control stick and not have to think about how that output will affect the game state.

1.2 Principles & Participant User Views

Equity, as outlined by Story, Mueller, and Mace (1998) in "The universal design file: Designing for people of all ages and abilities", states that "[t]he design is useful and marketable to people with diverse abilities". One such population of individuals who have diverse abilities are blind individuals. Much of today's

computerized designs rely on visual stimuli to interact with an individual. For blind individuals, this communication breaks down as they can't see what's on a screen. In applying the principle of equity to our designs for blind individuals, we might ensure that screen readers are able to properly parse and communicate our interface to users. The principle of equity forces us to consider the design in the context of our potential users' abilities, forcing us into a participant view of our users. In the case of blind users, the context would be interacting via a screen reader on a device.

Ease (or "Low Physical Effort"), as outlined by Story, Mueller, and Mace (1998) dictates that "[t]he design can be used efficiently and comfortably and with a minimum amount of fatigue", is yet another principle which emphasizes the participant view of a user. Let's consider the application of this principle to the interface of text input on a smartphone. Traditionally, this is performed via an on-screen keyboard. In most contexts, this is fine. However, there maybe scenarios where an on-screen keyboard is impractical, such as while cooking and your hands are dirty. In looking for a way to input text on a smartphone which enhances the ease of the task, one might look to a dictation interface. However, in a noisy room, the interface which enhances the principle of ease may be the traditional on-screen keyboard. In these cases, we're looking at the context of the user for ways to enhance the principle of ease in each situation.

2 QUESTION 2

An example of an interface from everyday life that is intolerant of errors the user commits could be a stove-top cooking surface.

2.1 Intolerant Interface

There are a few errors a user might commit while using a stove-top cooking surface. A user may leave their food on for too long at too high of a temperature, potentially burning the user's food. Additionally, a less organized user may have paper towels too close to the cooking surface creating a fire hazard. A user may also experience water boiling over while cooking starchy foods, creating a burnt mess to clean up later. In all of these cases, the stove top simply lets the errors happen and doesn't take any proactive measures to help the user avoid these errors.

2.2 Constraint Redesign

We could add some logical constraints to a stove to help the user avoid some of these errors. For example, the stove could use a motion sensor and run detection algorithms with the knowledge of if a timer is set on the stove to know if a user is neglecting their meal and warn the user to check on the food to ensure the food doesn't get burnt. Similarly, cameras could also be used as a sensor for this interface to ensure that burners are clear before letting the user turn on a hot surface, helping ensure that fires aren't accidentally started due to a cluttered work space.

2.3 Mapping Redesign

In offering a redesign which integrates a mapping as a strategy to eliminate user error, let's revisit the task users are looking to accomplish when using a stove. In general, the task is to cook or re-heat food. To offer a mapping-based redesign which helps avoid errors, like leaving the food on too high for too long, one might consider adding buttons or a companion app which lets users select pre-created heating and timing profiles for the food they want to cook. The idea here being that we want to establish a mapping between our user's goal and the interface with which they do it. A user could measure out the portions they want, set it on the stove, select a profile for the meal they want, and let the stove manage the heat and time settings. In this redesign, the user would also need to be notified of alerts or when they need to either stir or rotate food.

2.4 Affordance Redesign

Considering the mistake one might make where a cluttered work space creates a fire hazard at a stove top, there are already some existing affordances in place to help avoid this error. On most ranges, there's some indication of what area will heat up, whether its an electric coil, a gas burner, or a design imprinted on a glass top; these all let the user know where the heat will be on the cooking surface. However, that doesn't necessarily apply to the work space around the cooking surface. An inattentive user may leave a roll of paper towels around the work space which could then rolls onto a heated surface creating a fire hazard.

One way we could enhance the affordance which lets the user know where the heat will be coming from is by designing the work space around the cooking area in a way that discourages clutter around that area. For example, if we de-

sign the cook top such that things placed on a surface other than the cooking surfaces aren't level, the user will know that they aren't supposed to put something there.

3 QUESTION 3

For analyzing slips, mistakes and designed difficulty, let's examine the NES title: *The Legend of Zelda*.

3.1 Slips

The Legend of Zelda provides players with an inventory of tools to equip from in addition to a sword which is always equipped (shown in *Figure 2*). Players can select their additional tool from a grid in the pause menu. Intuitively, one might think they can select the tool using all buttons of the d-pad on the controller. However, the grid only navigates from left-to-right, top-to-bottom and wraps. So to get from the upper-left tool to the bottom-left tool, the user must cycle through all tools on the top row, pressing right on the d-pad, as opposed to simply pressing down on the d-pad to get to the tool on the bottom left. The slip can occur when the user is presented with the grid of tools and knows to select the tool below the currently selected one, but he or she forgets the up and down d-pad buttons are disabled in this grid menu.

A way to update the interface to help avoid this slip is pretty simple, allow the up and down button on the d-pad to select the tool above or below, respectively, the currently selected tool. This would help establish a stronger mapping between the menu and the d-pad and avoid any accidental mis-inputs.

3.2 Mistakes

Players of *The Legend of Zelda* are presented with an opportunity to make a mistake on the first screen they're presented with once gameplay starts. On that first screen of gameplay (*Figure 1*), the user is presented with the option to navigate North, East, West, or into a cave. In the cave, the user is given a sword which helps him or her defend his or her self from monsters. In any other direction though, the user will encounter monsters and find being able to progress through the game remarkably difficult. Nothing necessarily compels the user to explore the cave more than curiosity, but by that logic, a user may also explore the overworld. The mistake here is that a user may skip exploring the cave and

progress through the game without being able to defend his or her self in an effective manner, though they know they need to be able to do so somehow.

A potential way the interface could be changed to avoid this mistake is to add some sort of obstacle to exploring the overworld until the user has received the sword. The basic concept could be some brush that prevents the user from leaving the first map frame but can be struck down or removed once he or she has received the sword.

3.3 Designed Difficulty

A place where *The Legend of Zelda* can be perceived as difficult by design is knowing where to go next. For navigation, the user is equipped with a small display of the overworld map which simply displays their position relative to the world borders. No world features are tracked on the map. To find the dungeons player needs to clear before reaching the game's finale, the player must explore the overworld on a large-scale to find hints and clues as to where the next dungeon is and what they need to do to open its path. This exploration at a large scale is difficult and a player may find themselves having to back track to make sure they're on the right trail or revisiting NPC's to see if hints may apply to newly unlocked areas of the overworld. It's difficult and a lot to keep track of as a player, but it is a part of the game that makes the overworld so immersive.

4 QUESTION 4

4.1 Good Representation

An interface that I use which I feel presents a good representation of its underlying content is the GUI file system manager on Mac OS: Finder (*Figure 3*).

One way Finder presents a good representation of its underlying content is by making the relationships between directories, files, and parent directories explicit. The relationship between directories, other directories, and the files they contain is hierarchical in nature. Finder's column view supports this model of a hierarchy between directories and files. As you navigate throughout the file system, the hierarchy is preserved and additional columns are added per sub directory to support the idea that you are interacting with files contained within directories.

Of course presenting the hierarchical relationship is only one part of what makes

Finder a good representation of a file system. This presentation would be worthless if we didn't know the difference between directories and individual files. Finder uses different icons to show what items are directories and what items are files to help bring the relationships between directories, sub directories, and files together. Finder represents directories as what they are more commonly know as: folders. Subsequently Finder represents files as documents with an icon associated with the program that one would use to open it with. This representation helps bring together the idea that files (or documents) and folders can be placed inside folders.

4.2 Poor Representation

On the other hand an interface which presents a poor representation of its underlying content is the *ls* command on the terminal (*Figure 4*).

In examining the output of the ls command, we're presented with a very flat representation of the file system. We're only shown the contents of the directory that we're currently in. We're not being explicitly made aware of the hierarchy between the directory we're currently in and parent directories. In fact, this output may even confuse the user. What is ".."? Where am I currently in the file system? These questions are unanswered by what we see from our ls output. We're not presented with a view of the file system that promotes the idea that we are interacting with a hierarchical entity.

Another area the ls output fails to present a good representation of the underlying content of the file system is by providing too much information (albeit, you can suppress some of it). For starters, it lists the file / directory permissions, owner user, owner group, file sizes, modification dates, and hidden files / directories. Considering a power user who may want to interact with files and directories at a very low level, this is valuable information. However for traditional users, this is too much information. This output could be limited to only show files and directories which only the user is able to interact with. There isn't an explicit need to show the owner group and user if the user can't interact with that file or directory at all. Similarly for overall permissions, if the user isn't able to interact with the file, then there should be no need to show the file in the output of ls. This information doesn't provide meaningful information about the relationships between files and the directories. So as a tool for helping to manage files and directories at a basic level, the ls command simply provides

too much output.

5 REFERENCES

[1] Story, Molly Follette, Mueller, James L, and Mace, Ronald L (1998). "The universal design file: Designing for people of all ages and abilities". In:

6 APPENDICES

6.1 Figures

Figure 1—*The Legend of Zelda*'s overworld. Source: Nintendo Switch Screenshot.

Figure 2—*The Legend of Zelda*'s pause menu. Source: Nintendo Switch Screenshot.

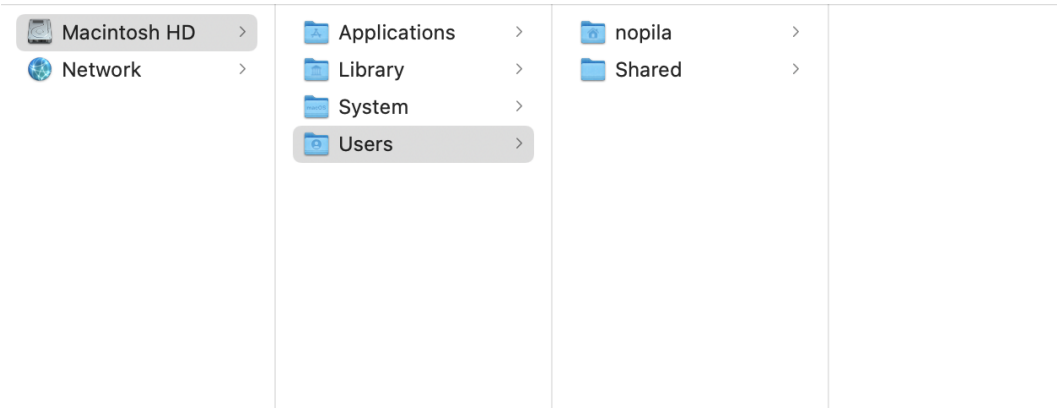

Figure 3—Finder's column layout. Source: Screenshot.

```
sh-3.2$ ls
total 0
drwxr-xr-x   5 root    admin   160B Jan  1  2020 .
drwxr-xr-x  20 root    wheel   640B Jan  1  2020 ..
-rw-r--r--   1 root    wheel     0B Jan  1  2020 .localized
drwxrwxrwt  16 root    wheel   512B Jun 20 15:17 Shared
drwxr-xr-x+ 83 nopila  staff   2.6K Jun 20 15:20 nopila
sh-3.2$
```

Figure 4—Sample output of the command *ls*. Source: Screenshot.