HCI CS6750 – Assignment P3:

Michael Lukacsko
mlukacsko3@gatech.edu

1 QUESTION 1

1.1 Creating Invisible Interfaces, 3 Principles

1.1.1 *Principal 1: Simplicity*

The first principal that supports the creation of an invisible interface is simplicity. Simplicity, when utilized correctly, provides only relevant information to the user, and thus makes the gulf of execution smaller because it reduces the cognitive load required by users to execute a task. The underlying theory of this principle is that the interface is designed with the user in mind. As such, irrespective of the user's level of experience, an interface is easy to understand and use. Moreover, for an interface to achieve simplicity, the output must also be easily interpreted. This in turn shrinks the gulf of execution because the user only receives feedback/output that is relevant to the task and easier to decipher. Hence, when output is relevant and easy to interpret, the gap to understand the current state of system is small.

1.1.2 *Principle 2: Consistency*

The second principle that supports the creation of an invisible interface is consistency. When a user is well versed in one interface, it is important to reuse those design elements in other interfaces. Not adhering to this principle can increase the cognitive load on a user because they would have to rethink and remember what they previously knew how to do. When done effectively, both the gulf of execution and evaluation are reduced as the user knows what actions to take to perform a task and can easily interpret the interfaces output after completing the task. A perfect example of this is copying and/or pasting a few words in a word document. Because Ctrl-C and Ctrl-V are universal shortcuts used across all word editing software, the gulf of execution is small because a user would instantly know how to accomplish the task. Furthermore, the gulf of evaluation is small because the user can easily map the result of their action back into their own mental model.

1.1.3 *Principle 3: Affordances*

The third principle that supports the creation of invisible interfaces is affordances. When interfaces are designed to bridge the gap between design and use, an affordance is used. In other words, when a user encounters an interface with affordances, they intuitively know what to do. Buttons are for pressing, and knobs are for turning. When used correctly, affordances bridge the gulf of execution because a user can easily identify their intention to act, the sequence of actions, and the execution of those actions. Likewise, the gulf of evaluation is very small as well when affordances are used correctly. It is very easy to determine the state of a system after a button has been pressed or a knob has been turned.

1.2 Creating Interfaces That Emphasize the Participant View, 2 Principles

1.2.1 *Principle 1: Equity*

Creating a system that is available to both novice and expert users and allows each user to use the system in the same way integrates the principle of equity. A perfect example of this is the iMessage application on an iPhone. Every user who interacts with this app has the flexibility to speak-to-text or use the integrated keyboard. This caters to those who might not be able to use the keyboard, such as the blind, but also for users who might not be able to physically utilize the keyboard given the context of their actions.

1.2.2 *Principle 2: Flexibility*

Like the principle of equity, the principle of flexibility can be used to create interfaces that emphasize the participant view. Depending on the user's preferences and abilities, flexibility enables the user to interact with an interface from many different contexts. Using keyboard shortcuts to copy and paste (Ctrl-C and Ctrl-V) as opposed to highlighting some text and right-clicking to select copy and paste enable users to accomplish the same task from two different contexts. Where one user might be more concerned with efficiency and speed, a different user will want ease and simplicity.

2 QUESTION 2:

An interface from my regular life that is intolerant of any errors I might make is the command line interface (CLI) when running Linux, Ubuntu in my case, on my PC. Personally, I prefer using a CLI over a graphical user interface (GUI) because it consumes less memory, is more efficient, its simple, and I feel cool doing so. That being said, using the CLI can be disastrous if used incorrectly. When using a GUI, most actions that remove or modify a folder or a file will prompt the user to confirm that the action they initiated is one they want to make. This is not the case when interacting with a system via a CLI. By entering the command "rm <yourTextFile.txt>" your "yourTextFile.txt" is gone and unrecoverable. The only way to recover a file after removing it is to restore a snapshot/backup of the Linux instance. For many, including myself, this is an easy error to commit, and the penalty is permanent.

It is generally assumed that novice users stay as far away as possible from the CLI and that anyone using the CLI is an experienced user. This is true is most cases, but as with all things there is a learning curve. After all, everyone using Linux had to use it for the first time at some point. Because of this, I think one constraint that could be added to improve the interface to avoid errors is a configurable setting that will only allow users to perform certain actions contingent on their level of experience and expertise. By introducing this constraint, a user with less experience is limited to what they can do that might be done in error and is also irreversible. If, as an example, a novice user enters a command "mv /home/users/* /dev/null", the interface would let them know they are about to move the users folder in its entirety to a blackhole (null) where it cannot be recovered. Next, there would be a prompt – "Are you sure this is what you want to do?" The same would be true when using the remove command or any others command that has significant repercussions. Adding this constraint would lessen errors made by novice users who might be entering commands they are not familiar with. Turning this constraint off or adjusting the setting for experienced users would allow them full control of the system.

Like the constraint above, improved mapping could be used to reduce unintended actions and subsequent errors. If performing an action that affects multiple files, the user might be prompted with something that shows all the files they are about to interact with. This would inform the user of the result of their

action before the change is made permanently. As an example, if a user wants to delete all files within a specific folder that contain "CS6750" in its name, they could use a command like "rm *CS6750*". After entering the command, a user could be shown all the files that contain "CS6705" and ask them to confirm the action before permanently removing them. Thus, this could potentially catch a user error before it is made.

Lastly, improved affordances could be user to avoid errors as well. For users of all levels, this could be the addition of color to certain commands. If a user is about to remove a file or make a change to a system file, the command could be changed to red before execution. If a command is entered with a low level of risk, it would be displayed in green. Incorporating this affordance would inform the user of the risk is associated with their desired action by using commonly known color scheme and thus aid in the reduction of user errors.

3 QUESTION 3

One game I am familiar with where slips can occur is the ago old game of tic-tac-toe. Players take turns placing an X or O on a 3x3 board. The winner is declared when one player gets three X's or three O's in a horizontal, vertical, or diagonal row while also blocking the opposing player from getting three of their shapes in a row.

3.1 Slips

A common slip in the game of tic-tac-toe is when a player misses a move and thus enables their opponent to win the game by getting three of their shapes in a row. In this case, the action-based slip is caused because the player knows that they are supposed to block their opponent from winning but commit the slip when they are preoccupied with the goal of getting their three X's or O's in a row.

To prevent an action based slip of this type, the interface of the board could highlight the boxes where shapes occupy a space. This would make it easier for the player committing the slip to see what moves are available and decide how to best place their shape to block the opponents progress.

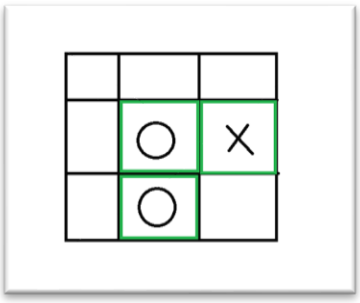

Figure 1 — Tic-tac-toe with occupied spaces highlighted. Source: Author

In the figure above, it is easy to see where an X could be placed that would also block the opponent from winning.

3.2 Mistakes

In the game of tic-tac-toe, rule based and knowledge based mistakes are probably the most common. In a case where someone who has never played the game of tic-tac-toe makes a move that enables their opponent to win, this mistake would be rules based. Here, the user incorrectly assesses the state of the board. From Figure 1 above, the player can see that their opponent has 2 O's on the board. The next move should be placing an X in the center of the top row, however, if the objective of the game is unknown, the player might place an X anywhere except the center of the top row. Again, referring to Figure 1, a knowledge based mistake could occur if the play just incorrectly assesses the state of the board. Here, if the player doesn't recognize that their opponent can win with an O being placed in the center of the top row, a knowledge based mistake could easily occur. Placing an X anywhere other than the center of the top row, in this case, would be a knowledge based mistake.

To prevent a mistake of any type from being made, the interface would include a red color where the opponent can win.

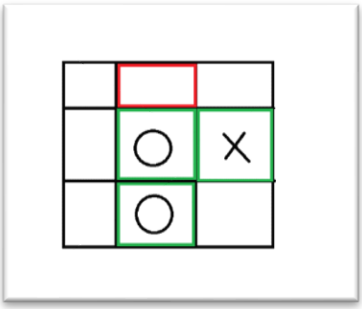

Figure 2 — Tic-tac-toe with opponents next move highlighted. Source: Author

Figure 2 above shows an example of this improvement. By incorporating this, the player knows that their opponent can win if the next move doesn't block it. This improvement would aid in the prevention of users making a knowledge based mistake. Where a user might make a rule based mistake, the interface could be improved such that the board highlights all the possible next moves. Here, the user needs to know the goal is to get three of their shapes in a row, but it also shows all valid next moves that can be made. Figure 3 below shows this.

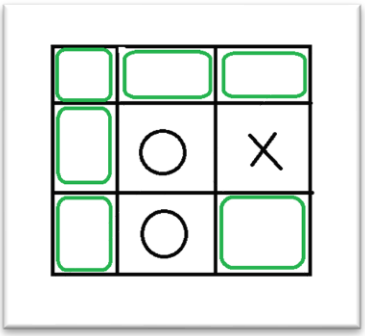

Figure 3 — Tic-tac-toe with X player available moves. Source: Author

3.3 Challenging but not a Slip or Mistake

The game of tic-tac-tow is simple in theory, but there is some strategy. It is even entirely possible to never lose and guarantee a win or a draw every time. A list of moves can be found here that represent the best strategies. Because of the multiple scenarios, the next best move is not often evident and can be missed. This can at times result in a draw and, even worse, a loss. When an X or O is wrongly placed in anticipation of the opponents next move, it is really neither a mistake nor a slip.

4 QUESTION 4

4.1 Good Representation of Underlying Content

An interface from my everyday life that uses good representation of its underlying content is the swing my wife and I used for our oldest daughter and will soon be using for baby number 2. The Image on the left from Figure 4 below shows what this interface looks like. There is a power button on the top, a + and – button to control speed, a + and – button to control volume, and 4 buttons that play different types of music and one that will rotate the mobile.

What makes this a good representation of its underlying content is the visual icon next to each button representing either what it does or the sound it will make if pressed. The power button, located in the top center, has an indicator light signaling whether the device is turned on. Also shown in the middle of the controller is the speed that the swing is set to. As the speed of the swing increases, so do the number of lights in the bar. For the music features along the bottom, one label has a moon and plays more relaxing music intended for sleeping or nighttime. The button with the sun on it plays more upbeat music and is intended for daytime use. The third button plays animal sounds, and the last button shows the mobile with arrows depicting it spinning in a circular motion. Another principle that denotes good representation of its underlying content is the fact that the whole controller excludes extraneous details. Each button serves a specific purpose, nothing more and nothing less. All the information that is present is relevant to using the swing effectively and, with any luck, soothing the infant who occupies it.

Figure 4 — Left: Infant swing controller. Right: Fan remote controller Source: Author

4.2 Not Good Representation of Underlying Content

An interface that I would argue does not use good representation of its underlying content is the remote controller pictured above that controls the fan and light in my daughter's bedroom. While the remote is simple in its design, if the fan is already running, it does not tell me what speed the fan is set to when I walk into my daughters' room. Additionally, the light on the fan is dimmable. The task of dimming/increasing the lights brightness can be accomplished by pressing and holding the button in the center down. Again, there is no indication of what the brightness is set to nor if pressing and holding the light button will dim the light or increase the amount of light. In these two examples, the relationships are not explicit and, as such, create a poor representation of its underlying content.

A second reason that this fan remote controller is a poor representation of its underlying content is that the controller does not do a good job of exposing natural constraints. Specific to the light's brightness, there is no indication if the light has reached its maximum or minimum level of brightness. As a user of this controller, I have to hold the button down long enough that the light starts to dim and then quickly let go. At that point the light will be at its brightest level. Inversely, to get the light as dim as possible, I must hold the light button down just until it starts to brighten. If my reflexes are good, the light will then be at or just slightly above its dimmest setting.