Assignment P3:

Mark Nowicki
mnowicki6@gatech.edu

1 QUESTION 1

Creation of an Invisible Interface:

Discoverability: Discoverability might be used to support the creation of an invisible interface because it allows the user to explore an interface and understand its function, preventing them from having to remember or research specific commands. This principle's benefits are most applicable to the "identify actions" phase of the gulf of execution, where discoverability allows the users to more easily identify the actions they need to take to complete a task.

Perceptibility: Perceptibility helps produce an invisible interface as the user remains informed about the state of the system and feels "in control" of the task, even if they are simply manipulating the interface. This principle helps reduce both the interpretation and evaluation parts of the gulf of evaluation, as a perceptible user interface provides relevant information the user needs to interpret and evaluate the interface output.

Simplicity: Simplicity reduces clutter that would otherwise make the user "aware" of the interface, thereby helping to create an invisible interface. Simplicity helps in both the gulf of execution and the gulf of evaluation. In the gulf of execution, it helps with the "identify intentions" and "identify actions" portions. A simple interface more easily allows the user to identify their intentions in the context of the interface because they are not overloaded with options; a complicated interface might make the user confused about their intentions. Imagine a sound board with hundreds of sliders and dials: if the user simply wanted to increase the volume, they might wonder what their intentions are in the context of this interface. Simplicity also helps with identifying actions, as there are simply less actions to take. In the gulf of evaluation, simplicity helps in all three stages because a simple interface allows for more prominently featured interface output, easing interpretation as the user knows that the output is directly related to their input, ultimately enhancing evaluation.

Interfaces that Emphasize the Participant Point of View:

Equity: Equity could be used to create interfaces that emphasize the participant point of view because equity both looks to "accommodate a wide range of individual preferences and abilities" and make sure the interface is "useful and marketable to people with diverse abilities."[1] Equity therefore seeks to understand the context of users outside of their interface interaction and incorporate those contexts into the design. Allowing for the change of language in an interface is one such example that considers how different users might speak different languages outside of the interface.

Comfort: Comfort can be used to create interfaces that emphasize the participant point of view because it ensures that user's traits, such as body size, posture, and mobility, are all reflected in the interface. This design principle considers the user's context outside of the interface interaction because it seeks to see how users might be limited in their abilities and incorporates those considerations into the interface. A door with a lower, automatic "open" switch would be one such example, as the designers consider that some users might not be able to reach high objects or pull things open, so they incorporate that limited ability into their design by providing an alternative.

[1] Lesson 8: 2.5: Design Principles and Heuristics, Video 14

2 QUESTION 2

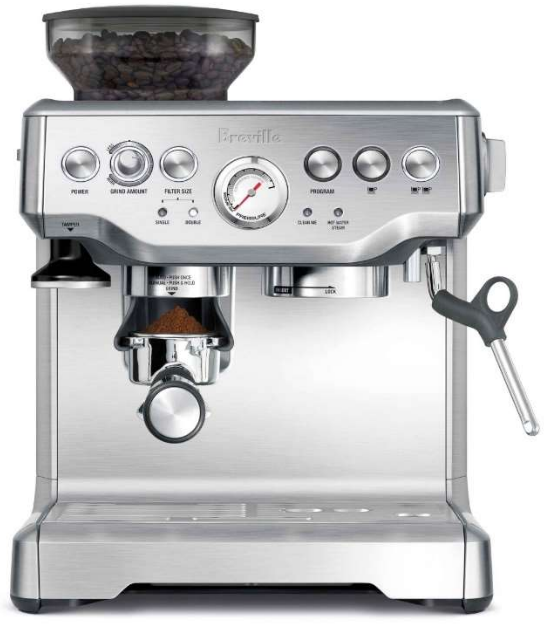

Figure 1: The Breville Barista Express

Select/Describe: Almost every day, I use the Breville Barista Express (shown in Figure 1) to make a coffee drink. The interface includes buttons and dials with labeling underneath each button. In order from left to right across the top, the buttons are power, grind amount, filter size (how much coffee to grind), program (for pre-determined amount of water volume), single espresso shot, and double espresso shot.

Error Response: Currently, this interface does not respond to user errors: for example, if a user accidently presses the single shot espresso button without any coffee grounds prepared, it will simply pour hot water into the receptacle. If a user hits the power button in the middle of espresso extraction, the machine will turn off.

Ease of Error: These errors are relatively easy to make, as the action occurs immediately after the user hits the button without any confirmations. The user is also likely to hit the incorrect button if they are moving quickly.

Penalty: The penalty associated with these errors depend on the error, but it could range anywhere from extracting coffee into the basin (instead of into a cup) to accidently extracting too much coffee.

Constraints: Adding constraints with additional technology and confirmations could avoid errors in the first place. With a screen and an optical sensor, for example, the machine could alert the user that they are about to dispense coffee into the basin and ask them if they want to add a cup or continue with their action. The machine could also confirm the user wants to shut the machine down while it is in the middle of another process, preventing wasted coffee. Moreover, the screen could even replace the physical buttons altogether, preventing the user from pressing erroneous sequences (such as extract espresso, then grind beans) because those sequences would simply not be present in the interface. A screen and a series of confirmations depending on user actions could therefore help to prevent user errors altogether by making them aware of the mistake they are about to make and guiding them away from that mistake.

Mappings: The buttons for single and double shots of espresso representing the button's functions through images is a great example of incorporating mappings. The grind size dial, however, is a very poor use of this design. The dial provides no intuitive way of telling the user how much coffee corresponds to their selection. Instead, a redesign of this grind amount dial could also use pictures: it could mark about how much coffee would provide a single shot or double shot of espresso. By providing visual markers, the user has a ballpark range of how much coffee to grind. By leaving it in this dial format, the format also allows more experienced users to tune the amount more precisely depending on the coffee (also making this redesign is also an example of flexibility).

Affordances: The "program" button design could allow for more affordances. The function of this button is to "save" the user's preferences on other internal settings such as temperature, filter size, etc. Like the other buttons on the machine, its design suggests that the user must simply press it to save the preferences, since a single press activates the button for all other buttons. However, the user must press and hold the button for a significant amount of time to save these preferences. Two possible redesigns exist to incorporate affordances. First, the function of the program button could simply change so that a single press is required, aligning the intuitive understanding of a button and its function. The only problem is that (I surmise) the machine makes the user hold the button down so they do not accidently overwrite current settings. A better redesign would be to add a signifier that they should hold the button down, such as writing "(Hold)" beneath "Program." The addition of this writing would alleviate the conflict between the actual affordance of a button and the perceived affordance of the user.

3 QUESTION 3

One game that I spend a lot of time playing is Blizzard's Overwatch, a first-person team-based hero shooter (meaning the view is in first person, the user plays with a team, they select from characters with pre-defined roles and abilities, and the game involves shooting).

Slip: A common slip is when choosing characters. Before the game begins, each player must select a hero to play. They can also change heroes mid-game at the spawn room. In an intense game, the player must select their hero quickly to get back into the fight. The process of choosing a hero involves clicking on the user's icon and then pressing a "select" button below. This interaction is where a slip might occur, as the user has the character they want to play in mind, but they accidently click on the wrong portrait, press select, and enter the game with the wrong character. The primary causes for this slip are speed and stress: the user wants to help their team as quickly as possible, and the game might be close, so the user might accidently select a different character and press "select". The result of the mistake is that the user needs to go through the character selection screen again. A bad redesign would be to have an additional character selection confirmation—this interface would prevent the player from getting back into the game and might add to the stress. Instead, the interface could simply have the hero's icon appear on the "select" button once the player chooses a hero's icon. By having this icon appear on the select button, they will select a hero icon and then see that icon again before pressing "select." This split-second of additional information would likely prevent slips because the user would not need to realize their mistake and open the character selection screen again—they would simply need to select a different character from the same screen and press "select." As such, the user would enter the game with the correct character.

Mistake: The game's UI includes crosshairs in the middle of the screen, a health bar in the lower left-hand corner and ability buttons (and hotkeys) in the lower right-hand corner. The game's UI does not include any description of how to move, however. A user might struggle to figure out what keys they need to press to move left, right, forward, or backward, and they might make a mistake by pressing the wrong key. The reason for this mistake is that the game does not teach you how to move—it simply places you in the game and lets you figure out how to move on your own. Even though the game includes a tutorial at the beginning, brand new players might also have issues remembering the buttons they need to press to move and accidently press the wrong buttons. The result of these mistakes is that the player might not move at all, or they might move into harm's way, hurting their character or their teammates' characters. A redesign to reduce mistakes is to add the movement hotkeys onto the UI (such as the arrow keys) for new users when they begin playing. As the game senses that the user knows how to move fluidly, it can start to phase out these UI components. In this way, the interface would help new or newer players avoid mistakes while eventually producing a sleek UI for more experienced players.

Challenging: The game's shooting makes it extremely challenging: the player needs to put their crosshairs on an enemy (who could be very small at a distance), requiring a large amount of player control and dexterity. Missing a shot on an enemy is neither a slip nor a mistake: difficulty aiming does not result in a user committing a mistake or a slip.

4 QUESTION 4

Good Representation:

An interface that I use every single day and uses a good representation of its underlying content is the computer keyboard (namely, I will discuss the aspects of the computer keyboard shared amongst most types and model: the letter and number keys. The **connections** are extremely clear: the buttons have labels which, when pressed, send a signal to the computer that the computer interprets as input. With any type of word editor, IDE, or CLI, this computer interprets the signal as a letter or number that it displays on the screen. While a layer exists between the printed letters/numbers on the buttons and the display of those letters/numbers on the screen, the symbols printed on the buttons directly correspond to the symbols that appear when the button is pressed.

Criteria 1: Excluding Extraneous Details: The keyboard is a good representation partly because it excludes extraneous details. The printing on the keys do not provide extraneous details about the character mapping in the keyboard processor, nor does it provide details about the signals sent to the computer. Rather, the only information a user needs to know is what will appear on the screen when they press that button. Through this representation, the keyboard tells the user only what they need to know and nothing more.

Criteria 2: Bringing Objects and Relationships Together: Although obvious, the labels on the keyboard represent the letters that will appear when the button is pressed. Imagine instead a keyboard where each letter is represented by its positioning in the alphabet, for example. This type of interface would be confusing for the user as they would have to remember the mapping from number to letter. The keyboard's design, then, makes the representation of the object and the object itself one in the same by printing the letter directly on the key. As such, the keyboard successfully brings objects and relationships together.

Bad Representation:

Unlike the keyboard, I would argue that the mouse does not use a good representation of its underlying content. The mouse I am discussing here is a typical computer mouse with two buttons (without labels), and a scroll wheel. The **mismatch** is also clear here: the interface of a mouse has no connection to its representation of the pointer on the screen. A first-time computer user only learns that the two are related because the pointer on the screen moves when they move the mouse. Without this trial and error, a user would have no idea that the two are related to one another. Moreover, the default representation of the pointer on a screen is a little arrow. Mice, meanwhile, are little ovals. Even the shapes do not reflect one another. The mouse interface then is a poor representation of its underlying content.

Criteria 1: Make Relationships Explicit: The mouse's design tells the user nothing about how the different options relate to their function on the computer. Left clicking being the "select" click and right click being the "menu" click is entirely arbitrary, and nothing about the interface relates to its function with a computer. Simple labeling the buttons might make this clearer. The scroll wheel is also not made explicit: most webpages have a long scroll bar on the left side that slides up and down, but the mouse has a wheel. Once again, a user only begins to understand this representation through trial and error.

Criteria 2: Excludes Extraneous Details: Unlike the keyboard, which excludes just enough details so that the user understands the purpose but is not confused, the mouse excludes *too many* details. Most mice have no labels at all: simply two unlabeled buttons and an unlabeled scroll wheel. The result of this design is that the user has no clear idea what each button does when they first use a mouse. Moreover, they might miss that some functionality even exists: how long did it take you to learn that you could press down on the scroll wheel? While some other poor representations might violate this criterion because they include too many details, I argue that a mouse violates this criterion because it excludes too many details altogether, leaving a user unable to understand the representation.