Assignment P3:CS6750

Sachin Jose
sachinjose@gatech.edu

QUESTION 1

1.1 Discoverability

The principle of discoverability deals with making the functionality visible or discoverable. To make an interface invisible, the user must spend minimal time thinking about the interface and most of the time interacting with the object itself. The principle of Discoverability does this by helping bridge the gulf of execution during the **planning and performance stage** by conveying the available functionality of the interfaces effectively to the user. *eg: the menus of the classic iPod. All functionalities were three clicks away*

1.2 Affordances

Affordance is the principle by which the very design of the interface tells the user how to use it. Affordances supports the creation of an invisible interface by bridging the gulf of execution by alerting the user to the capabilities of the interface by utilising signifiers. The use of Affordances help the user to **plan and specify** their action thereby making the interface invisible. *eg: Arrows pointing to the next image in a profile on Instagram*

1.3 Feedback

Good feedback helps make the interface invisible by bridging the gulf of evaluation. Feedback is linked to the **'Perceive'** stage of the gulf of evaluation. When interacting with an object using an interface, the effect of the action on the interface is evaluated using effective Feedback. A good feedback can be interpreted and compared to the desired result and if it is not what is required, it can be corrected to match the required state. *eg: the bluescreen on windows.*

1.4 Ease

The principle of Ease deals with the interface being used efficiently and comfortably. The principle accomplishes this by reducing the cognitive load and fatigue caused to the user by the interface. *Ease emphasizes the participant model of the user by reducing the time taken to think about the interface and thereby making the* *interface invisible. eg: when opening a chat application the most frequently used contacts are at the top*

1.5 Constraints

Constraints are another principle that could be used to emphasize the participant model of the user. It involves constraining the user to be able to perform only the correct course of action. *Constraints emphasize the participant model of the user by preventing and reducing slips and mistakes that can be performed by the user when working on an interface. Eg: a constraint the prevents the user from driving when drunk*

QUESTION 2

My current job is that of an Application Admin and most of my time is spent using PuTTy to access Linux Servers. *The interface that is intolerant of errors in this example is that of a Linux command line.*

The interface consists of the multiple lines, to the left of each prompt the <username>:<group_name> is displayed providing information about the current user/group. The $ sign prompts for a command input. The previously entered commands can be found on top of the current commands. The command line interface does not deal well or at all with user errors.

For example to remove a tmp directory inside a folder that I navigated into, If I use the command *'rm -rf /tmp'* and provided I have access to the /tmp folder on the server the command actions on the server present at the root level rather than the tmp folder in the current directory and once the command is executed there is no way to revert the steps taken than by using a backup if taken, if not you're on your own. I once accidentally messed up a server by deleting the /etc/ folder as root when I wanted to perform some configuration changes inside a folder, thankfully it was a practise sandbox server but all it takes is one misplaced character to make an unrecoverable error.

Linux does put in constraints by providing user groups and permission groups and making sure that people only have restricted access to specific directories but as it is said in the 'Design of Everyday Things'. Professionals are more likely to commit an error more than novices. . A **constraint** that could prevent the above mentioned error of deleting from a wrong directory would be that in case of a name resolution situation, the name closest to the current directory would be chosen first, similar to scoping for variable names in C++. So if a tmp directory exists in the directory that the user is trying to perform the deletion of the closest tmp would be deleted. For the root tmp folder I would use a more complex scope resolution operator rather than ‘/’ to access the root directory.

A **mapping** that can be used to help the linux interface would be to print out all the all files encompassing the folder before asking for a confirmation. This would enable the user to understand which directory the action is actually being performed on with a really graphic reminder of the directory where the action is being implemented.

An **affordance** that can be used to avoid errors would be to add color coding to the command line text when typing, similar to code editors. A ‘/’ could be coded displayed in red to alert the user that this points to a file location outside of the directory that they are in. Linux commands can be color coded based on their ‘revertability’ ie, a command like cp and mv doesn't break the system and can be easily reverted albeit manually. For commands that can’t be reverted like rm this can be color coded to a more cautionary fashion.

QUESTION 3

The game that I’ve chosen for this question is that of CS:GO.

3.1 Slip

A slip that can be made while playing this game occurs when using snipers or guns that require aiming. For guns like the sniper rifles the right mouse button provides the user with a scope view, pressing the right mouse button once provides a zoomed in view and pressing it twice provides a really zoomed in view and the left mouse button is used for shooting. These are the default controller settings. One slip that is caused by the game is when we play a multiplayer game, in between aiming and shooting. I accidently made the slip of right clicking instead of performing a left click. Usually when I make this slip, this occurs in the later rounds of the game where I might be the only player left and have objectives to perform additionally there are multiple hostiles on the enemy team who are looking to take me out so I am also constantly on edge. By the time I exit out the scope and re-aim and shoot I would be dead. The cognitive load placed by this situation and the proximity of both functionality is why this slip is made. This slip can be prevented by changing the mapping of this functionality to the keyboard. So that the user can control the scope using the keyboard thereby reducing the proximity of both the buttons.

3.2 Mistake

A Mistake that can be made while playing CS:GO occurs when defusing a bomb. The game revolves around a T side which plants a bomb and a CT side which aims to prevent the plant or to defuse the bomb if planted. Oftentimes when the CT side, when defusing a bomb there are players on the T side still active to prevent one from defusing the bomb. So the player is always on the lookout for the T side players while defusing to eliminate them before continuing to defuse. But, in the latest update of CS:GO if we look around while defusing the bomb above an unknown angle, the timer to reset the bomb is reset. Inexperienced players often do not know the fine thread and often go over this 'viewing angle threshold' when defusing the bomb and reset their timer in the process. This mistake is primarily caused by the peer-pressure of the T side and how the game is structured as the CT is most vulnerable when defusing the player and tries to get an edge looking around when defusing.

3.3 Challenges

One challenge that is faced in the game is ammo management. Sadly, unlike movies where users never run out of bullets in CS:GO we have to reload our guns. If you run out of bullets when exchanging fire with the opposite team and you are not covered, this means sure elimination. This is a challenge because in almost all parts of the game you need to seek cover when engaging your opponent to reload and have to strategically use your other resources to prevent running out of ammo mid-fire.

QUESTION 4

An interface that you use a good representation of its underlying content would be Apple Books. In Apple Books various books have a progress bar and a % read option to see how much of the book is remaining to read in real life where if you use a bookmark, we can physically see how much of the book is left reading. Another good representation is that the text is broken down into pages rather than a continuous scroll of text when we swipe the next page a virtual page flip is simulated to make the constraints of a page more obvious and to make the relationships between text and pages, chapters and the overall book more obvious. The connection exemplifies two characteristics of a good representation: *making the relationships explicit* and *good representation exposes natural constraints*. The individual Books in Apple Books make the relationships between chapters explicit by providing hyperlinks between referenced sections of the book like in the index. It exposes natural constraints by limiting the contents to a page rather than providing a continuous scroll and the pages remaining in a chapter are displayed below similar to the representation of a real book.

An interface that does not use a good representation of its underlying content for me is the windscreen wipers in my car. My car is a 2016 Tata Nano and its windshield wiper controls are a really bad representation of the underlying content. The knob can be shifted both up and down from a neutral position on moving the knob one level up, it sets the wiper to an intermittent setting where the wiper activates every 5 - 10 seconds. Moving the knob one level higher moves it to the maximum speed setting. Moving the knobs one level down sets it to an intermediate speed and a level below sets it to the slowest speed there is. So the wiper knob works as shown below.

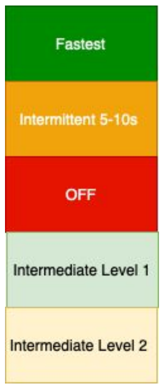

Figure 1— Controls level of my wiper knob

I live in one of the wettest regions in the world so for a large part of my commute I constantly find myself fiddling with my car's wiper controls. It violates the characteristic of *making the relationships explicit* as the fastest and slowest controls are placed right next to each other followed by 0 and then the two intermediary settings. The relationships between the controls are not explicit (there are no markings to indicate the setting). The second characteristic that is violated is *exposing natural constraint*s the knob wiper controls on my car are not restricted on their movements. The car that I use currently when I let them go or do a tug in one direction it immediately turns off. But the wiper controls in my nano has no such boundary that would turn off the wiper.