CS-6750 - Assignment P3

Christopher Doane
cdoane3@gatech.edu

1 QUESTION 1 - DESIGN PRINCIPLES

1.1 Three Principles that Support Creation of Invisible Interfaces

Discoverability - is the design principle of making it possible to learn what is possible to do with an interface (Norman, 2013) without the need to consult documentation. In the context of bridging the gulf of execution, discoverability mainly concerns itself with the identify actions phase. Discoverability lets the user know what actions are possible in a system, so that a user can devise an action in the goal of moving forward towards achieving their intentions, thus leading to greater invisibility of the interface. Without a discoverable interface, the user may choose a suboptimal action, or may even be unable to identify an action to execute. Finally, the user will execute an action, which might even be novel to the user if the feature was discoverable.

Consistency - is about both reusing design elements in a consistent manner through the interface (Constantine, 1999), and being consistent with what others have done in the same problem domain so that knowledge can be transferred (Norman, 2013). In the context of bridging the gulf of execution, consistency is very important when identifying actions. If an interface works in a way that is consistent with other interfaces in the domain, then the user might be able to use the new interface without much thought, as interface elements are organized and work in a familiar fashion. When bridging the gulf of evaluation, consistency in displaying results helps the user interpret output faster. If an interface is consistent enough, then this evaluation can become automatic.

Flexibility - refers to letting users interact with interfaces in multiple ways, accommodating a wide range of user ability and preferences (Mace, 2006). In the gulf of execution, flexibility is important for supporting the identify action phase. If an interface supports a user's expectations and usage preferences, then there is no bottleneck to the user's action selection and execution. By supporting multiple means of interaction, the interface can cater to and different users use the method they are most comfortable with, helping to build invisibility and equity. While flexibility is mainly associated with quickly bridging execution, when ex-

amining it in context of the gulf of evaluation, it is important that the interface also outputs to the interface in a way that corresponds to the method used.

1.2 Principles that help create interfaces that emphasise the participant view

The participant view is concerned with looking at the broader mental and environmental context of a user and their personal abilities.

Perceptibility Systems should always let users know what the state of the system is. Perceptibility is particularly concerned with the participant view of the model, in that, a design should be able to communicate information regardless of user's abilities and environmental context (Mace, 2006).

In terms of catering to user abilities in the participant view of the user, perceptibility is all about making sure that the user can perceive the state of system, even if said users have different sensory abilities. If one user has hard time hearing, then a perceptible system will also cater in a way so that notifications are not only audio-based. Likewise, if a user is visually impaired, an interface following perceptibility can also communicate state in another means complementing the visual. Users may not have impaired abilities, but their environment that which they use the system might get in the way as well. Systems that follow this design principle, help take into account the participant views environment, if that environment is competing for attention with the user, or if the environment itself makes it hard to see a screen, hear a sound, or dedicate full attention.

Comfortable The comfortable design pattern is about ensuring appropriate sizing and shape among other things so that users of various physical needs and size are catered to. Too small buttons are not good for users with large fingers for instance. A classic example is given with a programmable car seat auto-adjusting its positioning to enable a uniform user experience when driving a car and reaching the pedals. The design here takes into account the environment and the needs the user has in order to use the interface. The same idea of comfortable design principle can be applied to say a motorized bed in an elderly home. Some users will have no issue getting into and out of bed, but some will need extra help to push themselves up into a sitting position so that they can move off the bed. Both types of users have the same goals and desires, but have different abilities, and by building a system that takes comfort into account, the system is ensuring that it can handle different participant views and user abilities. When designing for user comfort, system designers are building systems

that are also comfortable to use in various environmental contexts, which also helps make the system more pleasurable and appropriate to use at all times.

2 QUESTION 2 - INTERFACE INTOLERANT OF ERRORS

One interface that I've encountered that is fairly intolerant of errors, but in a round-about and expensive way, is the parking brake on my parents older Mercedes work van. When I visit them, I sometimes take the extra van out to go to the food store as they live on the country-side. The parking brake is really quite necessary, as the van otherwise has a tendency to roll if I can't find a curb to ledge it against and place the opposite gear direction on the shifter. The interface is composed of three parts. First, a lever is pushed down with the left foot to engage the brake. Second, a second handle is pulled in order to release the brake. Finally, a light and a siren is present on the dashboard that activates after driving a while with the brake on. The error comes in primarily in two ways. One is that I often forget to engage the parking brake, and thus the system gives no feedback other than the van starts rolling if not parked smartly with this in mind. The other error is when I start up the van and start driving, it isn't until one has driven way more than necessary that the alarm kicks on that tells one to release the parking brake. At this point I have most likely ruined the brake, and have been known to wear down their poor brake over the years and resulting in its needed replacement. I only get to know that I have done wrong after the fact is true, and by then it is often too late to recover from.

2.1 Improving with constraints

This interface for the parking brake could be improved by simply not letting me make the two types of errors in the first place. First of all, the system itself could automatically engage the parking brake when the car is turned off by default. In nearly all cases, I want the parking brake on when the van is parked anyways. Secondly, the van could prevent me from pushing on the gas pedal when the parking brake is engaged, thus never letting me get into a state of error where I ruin the brake.

2.2 Improving with better mappings

The brake interface could do a better job mapping how the brake works or in what state it currently it is in as well. Right now the brake is an unmarked lever that is fairly hidden, and is easy for me to just miss when I haven't driven their

van in a while. The release lever is similarly placed, and is in a totally different location in the car compared to the other primary controls when starting and shifting the car to get it going. The placement of the controls could be placed more closely to where they are meant to be used, possibly close to the start engine key hole instead or similarly so that one gets a reminder to do certain tasks together when starting or stopping the van. This is an example of grouping together certain functionality together that is used together in an interface. The warning signals and such are also not directly mapped to how the system actually works. It should instead warn immediately if such a warning is employed, to better map to how the state actually is. Otherwise I think everything is okay, until I learn that it is not, and then it is too late.

2.3 Improving with affordances

Finally, the brake interface could be improved with an adjustment to its affordances. Right now, the car does not communicate that it has its parking brake in unless one pulls on the release lever. The release lever could instead be designed to be pressed in visually when the brake is in, and extended when it is out. Right now it looks the same regardless of state. This gives a clue as to how and when to use the lever. Otherwise I find myself yanking the lever a few times before starting the car (when I remember!) and hope for the best. A form of hidden affordance could be to hide, but then make the parking brake controls more visible when stopping the car, whether that be lighting up the control or something to help remind the user that the feature exists. The lever to press down to engage the brake is also slightly hidden and fairly far into the bottom of the car, requiring me to stretch to reach it with my foot to engage the brake. This lever could instead be designed so that it is easier to access by those with shorter legs like me. By making it easier to reach, it also becomes more discoverable to new people driving the car, or to users like me who tends to forget about where it was placed between visits to my parents place.

3 QUESTION 3 - SLIPS, MISTAKES AND ERRORS IN A GAME

One game that I had the chance to play this last winter, was the game Cyberpunk 2077, that I played on release. It had a lot of issues, which I think makes it a great candidate to discuss here and examine how users could make slips and mistakes, as well as how some elements in the game were designed as they were to pose a challenge to the user. Cyberpunk is an action shooter game that lets a user

play the role of a character that can navigate an open world placed in the future to perform missions and side quests. If playing on the PC, users likely control the character's movements and interactions in the game using a keyboard and mouse.

3.1 A user slip in the game

One major action-based slip that I had a hard time avoiding was accidentally hurting a civilian when in a gun fight in the game with another faction or gangster group. This often was an accidental death by shooting the wrong individual other than the one I was supposed to in the game. I know not to hurt civilians, but it is easy to do accidentally. The result is that the game triggered a police response any time this happened, and since the game was a bit overly sensitive, it would render hundreds of police out of nowhere and I would die nearly instantly. It was so hard to avoid this slip sometimes as civilians were often packed in areas where I was supposed to take out some targets. Since the police response was so huge, and there was no way to recover from it, as the police would follow and spawn around me regardless if I ran away or even jumped in the water. The interface could be modified in a way to show an X on the cross-hair or some red highlight feedback when I pointed the weapon at the wrong person. This would be an example of increased mapping that an action going forward would result in a penalty. Or require a few hit-box shots before a person got hurt, so as to separate intent from slip. This would help me from first even committing the slip when I did not mean to as the system would be constrained to not overreact on the first mistake, but also would help me recover when I made an unintentional click on the wrong person or on a person who ran between the line of fire between my character and the target.

3.2 A user mistake in the game

One rule or knowledge-based mistake (depending on how one looks at it) that a user can make in the game is walking too close to an ongoing investigation area where police are questioning a suspect in a crime. I correctly interpret that distance to characters determines the audio level, however I incorrectly performed an action according to how the world works - not inline with my intentions. In the game, one can often listen in on conversations occurring between some police and suspects. However, if one stands near a police for too long, they suddenly get angry. Here I wanted to listen in on the conversation, and interpreted

the increased audio as a hint to stand closer - my model of the world was that I should stand closer in order to hear better. However that is a mistake, as the game would then label me as hostile and take me out. I was not meaning to be hostile, I was only trying to listen, using cues from the game to steer me towards the conversation. The interface could be modified instead to give the user a longer warning period. I believe this was patched later on in the game, but a good way to mitigate this mistake would have been to have the police tell me to back off audibly and give me a period of time to do so, which would promote discoverability as to what will happen if I continue, instead of immediately trying to blow my brains out from listening in too closely. The screen could provide a better representation with a blinking warning light or facial character expression changes that I am about to make the police angry visually as well.

3.3 A challenge that is neither a slip nor mistake

One thing that makes Cyberpunk challenging, but that is neither a slip nor mistake, is the system for deciding which side quests to take on and in what order. A user can decide to take on just about an side quest that is present on the game map, however, taking on a quest that one is not prepared for will likely lead to failure. This is however a design choice by the game makers, to make the experience of balancing risk and reward as a part of the experience in playing the game. One will lose a life a few times before figuring out what one can handle and not handle in terms of quest difficulties at various stages of the game. This is not a slip, as the user does not accidentally engage in a side quest, it is very much a purposeful engagement. It is not a mistake either, because the user does know how to engage in a side quest, it is just sometimes the character is way in over their heads.

4 QUESTION 4 - REPRESENTATION IN INTERFACES

Designers get to choose how interfaces are presented to users. Some interfaces are good, following criteria for good representation, while others violate those criteria. To illustrate this, I have selected two clock interfaces that differ greatly in their adherence to good representation.

4.1 A good interface

An interface that I consider to have a good representation to its underlying content is the interface to select between the different clock functionality of my

android phone. To build context, I have attached a portion of the interface below.

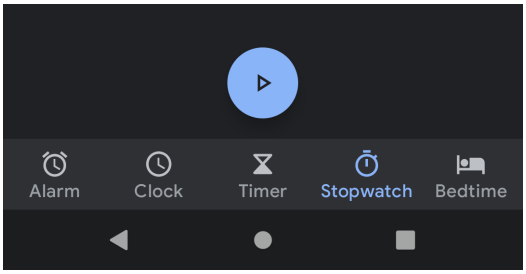

Figure 1—My Android Clock Interface

Here we can see that designers chose a representation of three components. Icons, text and color to show what each button represents in terms of functionality, as well as communicating which function is selected. This is an example of bringing objects and relationships together, by representing the features as figures of what they manipulate, the designers have managed to close the gap on the distance between the objects being manipulated and the way they are present in the interface. The representation is also simple, eliminating extraneous details that the user does not need. If they want to see more information on each function, they can instead click on each button to configure each function. The phone technically has support for many more nuances of the above main five categories, however it only shows them when relevant to the user - that is when a user has explored deeper within one of the clock function views. The representation also exposes quite clearly that one can only select one button at a time visually. As we will see in the next interface example, sometimes this is not as obvious. Additionally, the relationships here are explicit. Clicking on the icon representing a stopwatch will open up the stopwatch on the phone. Clicking on the alarm-clock icon, will open up a configuration page to select what alarms to turn on or off.

4.2 A bad interface

One interface that I feel does a bad job matching its representation to the underlying content is the interface of changing the time or settings on my physical Casio A158WE watch. The watch is a digital one, and supports showing the time, setting an alarm, and running a stopwatch. One can access each of these features by pressing the three buttons on the edge of the watch. These buttons are however of exactly the same size and shape and are unlabelled. There is absolutely no way to know what button does what until one presses on them.

The mismatch here is that one does not know what button corresponds to what feature. One might even be surprised to learn that two of the buttons are ones you press and release, while one is instead designed to be held in, and turns on a lamp while held in. Now, Casio has added a mapping for the buttons at the top of the watch, however this might not be obvious to the user at first. The picture below will help build context.

Figure 2—My Casio Watch Interface

The representation here violates two things, first that the relationship does not bring objects and relationships together well. The individual buttons are of the same size and shape, and do not have a direct label or even symbol attached to them to depict what each does (unlike the symbols in the android version). Only if one manages to squint hard, can one see that the designers instead put a cheat sheet up above the time display, however each of these labels is both small and far away from the actual buttons that it is easy to miss them. A better design would have placed the labels near each button. Secondly, the representation does not expose its natural constraints in a straight-forward way. It is not clear for instance that pressing the light button disables the rest of the buttons. Nor is it clear that a stopwatch can be running in the background on the watch by looking at the main front display unless on navigates to it. Another fun fact is that the watch does not show how much charge is left in the battery until it dies. This might be disastrous if one had planned on using the watch as an alarm to wake up before a trip, but the watch died before the timer went off. The rules of the system are not very clear, and require experimentation to understand.

5 REFERENCES

[1] Constantine, Larry L (1999). *Software for use : a practical guide to the models and methods of usage-centered design*. eng. Reading, Massachusetts: Addison Wesley. ISBN: 0-201-92478-1.

[2] Mace, Ronald (2006). *Principles | Institute for Human Centered Design*. URL: https://humancentereddesign.org/inclusive-design/principles (visited on 06/19/2021).

[3] Norman, Don (2013). *The design of everyday things: Revised and expanded edition*. Basic books.