Assignment P3

Angela Ruthven
aruthven3@gatech.edu

QUESTION 1

1.1 Design principles supporting the creation of invisible interfaces

Affordances support the creation of invisible interfaces by making the "identify actions" phase of the gulf of execution easy for the user. By definition, affordances and signifiers make the possible actions in an interface immediately apparent (e.g. a doorknob can be turned, and the rest of the door can be ignored), decreasing the time and cognitive effort the user has to spend on figuring out what actions are possible. This frees the user to focus on their task, making the interface invisible.

Mappings also support the "identify actions" phase of the gulf of execution, but in a slightly different way. Mappings help the user to translate their intentions determined in the initial phase of the gulf of execution into actions that are likely to achieve these goals in a particular interface. This makes the interface more invisible, since the more obvious the mapping between intention and action, the less time the user has to spend guessing how the interface works, or trying out different actions within the interface. It's more likely that the user's first attempt will be successful.

Perceptibility helps to bridge the gulf of evaluation. It helps with both the "interface output" stage and with the "interpretation" stage. One aspect of perceptibility is ensuring that the system produces output that can be detected by the user, i.e., that the user receives feedback through some sensory mechanism. Through this, the designer ensures that the output in the "interface output" stage exists and can be perceived by the user. The second aspect of perceptibility is ensuring that the sensory feedback received by the user is meaningful. This focus helps support the "interpretation" stage of the gulf of evaluation. By making the output of the interface easy to detect and interpret, perceptibility makes the interface more invisible to the user.

1.2 Design principles supporting the user’s context

The design principle of **flexibility** says that when designing interfaces, we should seek to cater not only to a user’s abilities, but also to their preferences. Since a user's preferences may include wanting to use the interface in a variety of contexts, application of this principle can result in interfaces that emphasize the participant view of the user. For a design to support a variety of contexts, it isn't sufficient to only consider what the user is physically able to do or the user's thought process as they use the application, but also what is going on in the world around them.

For example, some users may prefer to look up bus schedules from home. They may be distracted by trying to get ready for their upcoming excursion, and so have high cognitive load, but they have a lot of screen space on their home computer to look at the schedule. By contrast, others may prefer to look up bus schedules when they are already at the bus stop. They might have low cognitive load, since they're bored waiting for the bus, but only have a small amount of screen real estate available on their phone. All of this should be considered in the design of a flexible interface.

The principle of **ease** also requires considering the participant view of the user. An interface that is comfortable in one context can be uncomfortable and fatiguing in another. Therefore, focusing on ease requires the designer to consider the user’s context.

For example, suppose a user is checking bus schedules from her phone. If she is at home on the couch, with a floor lamp in the room, she may be quite comfortable. However, if she forgets to check the schedule until after turning off the lights to go to sleep, the same interface may now be too bright and cause eye strain, since she is in a dark room. If she waits until she arrives at the bus stop to check the schedule, and it’s a sunny day, she may again experience eye strain, but this time from squinting to read the characters with insufficient brightness or in the presence of glare from the sun. All of these considerations only come up when considering the participant view of the user.

QUESTION 2

At home, we have a small hand vacuum. It has a rechargeable battery that doesn't last long. To make it usable for the next person, we always aim to leave it charging when we're not using it. However, sometimes we forget.

To put the vacuum away properly, you need to complete two steps. First, you need to hang the vacuum up on the wall bracket. Second, you need to find the power cable and plug it into the vacuum. When you plug the cable into the vacuum, a red LED on the vacuum turns on (as far as I can tell, this is the only time this LED is used). This is shown in Figure 1.

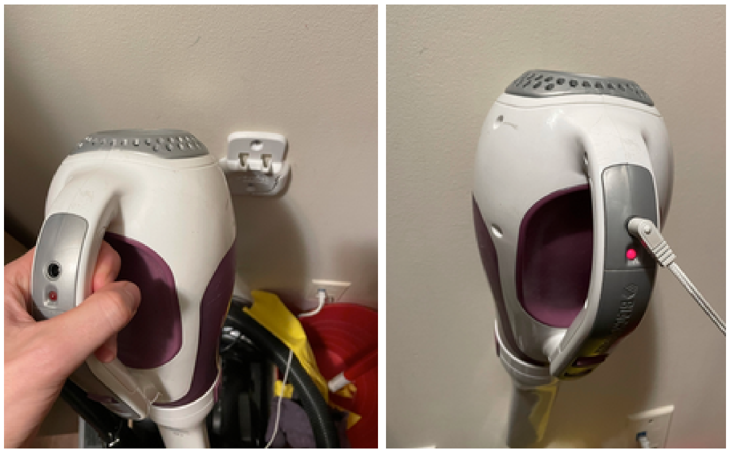

Figure 1—Putting away the vacuum. The photo on the left shows what the user sees when approaching the wall mount. Note that the power cord isn't visible (the white cable is an Ethernet cable, not the power cord). The photo on the right shows what it looks like when the vacuum has been put away properly.

It's easy to commit a memory lapse slip and forget to plug the power cable into the vacuum after placing it in the wall bracket. The penalty for this is that the next user can't use the vacuum for several hours, because there isn't enough suction.

The easiest way to solve this is with a **constraint**: embed the charging circuitry directly into the wall bracket, so that it is impossible to put the vacuum away without simultaneously charging it.

An alternative redesign would be to make the true affordance of plugging the power cord into the vacuum into a perceived **affordance**, making it very obvious that this action is possible. Currently, the wall bracket and the power cable are completely separate, so the power cable is often lying on the floor. If you aren't explicitly thinking about charging the vacuum, it's likely you won't notice the power cable at all. Technically, the empty power socket on the vacuum itself could remind you, but that doesn't look any different from how it looked when you were actually using the vacuum, so it is easily ignored. If instead, the power cable was clipped onto the side of the wall bracket, with its end dangling suggestively from it, you would see it when you put the vacuum into the wall bracket, and be reminded to plug it into the vacuum.

One final alternative to solving this problem is to improve the **mapping** between the internal state of the vacuum and the user's workflow. When I am actively using the vacuum, it is obvious from the sound the motor makes that the battery is really low. However, once I have turned off the vacuum, I have no way of knowing the battery level. The point at which I need to know this information about the state of the system is when I'm putting the vacuum away. The feedback given by the vacuum's interface (tell the user that the battery is low while the vacuum is in use) is not a good fit for my mental model of how I want to use the vacuum (charge the vacuum before the next time I use it). By adding a battery level indicator which is visible when the vacuum is off, the mapping between the system's state and my workflow would be more natural. For example, the red LED could blink whenever the vacuum has low battery and is not plugged in - the frequency could increase as the charge decreases and the need to recharge becomes more urgent. To make this mapping even more explicit, the red LED could have a standard battery symbol printed beside it. This way, when I am done using the vacuum, the blinking red light will remind me to plug in the power cord.

QUESTION 3

Hanabi is a co-operative card game in which players cannot see the cards in their own hand. However, they can see the cards held by every other player. Each player holds their hand so that the cards are facing away from them. Players learn information about the cards in their hand by being given clues.

A common slip in Hanabi is for players to look at their own cards at the start of the game, once the cards have been dealt, or after drawing a new card from the deck. The reason they do this is because Hanabi is inconsistent with most other card games, in which it is standard procedure to look at your own cards. In Hanabi, if a player looks at their cards at the start of a round, typically you need to start again and redeal all the cards. One way to decrease the frequency of this slip would be to print the words "Don't peek!" or "Don't flip me!" on the back of every card in the game. This might be enough of a reminder to stop players from reflexively looking at their cards.

To help a player learn about the cards in their hand, other players give them clues. There are rules about the form these clues can take. Cards in Hanabi have a numeric value and a colour. When giving someone a clue, you either point to all their cards of a particular value, or you point to all their cards of a particular colour. A common mistake in Hanabi, especially for new players, is to only point to a subset of the cards with a particular value or colour. For example, suppose the player has two blue cards, as shown in Figure 2. Pointing out only one of the two blue cards is not allowed; if you want to tell the player about their blue cards, you must point out every blue card. A player who misses this is likely making a mistake, and not a slip, because they don't understand the underlying game rule properly.

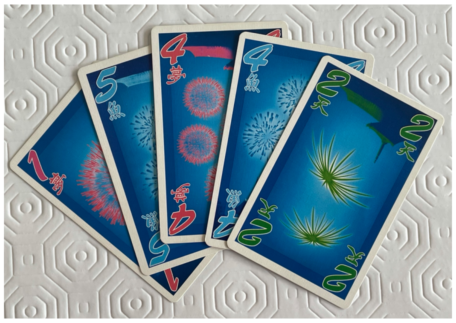

Figure 2—A sample hand of cards in Hanabi. Another player can give the clue "you have two blue cards," but it is a mistake to give the clue "you have one blue card."

I think the player might make this mistake because the instructions include a graphical example with the phrasing "You have 2 cards with a value of 1," as shown in Figure 3, even though that statement is technically true even if the player has 3 cards with value 1. The text beneath the picture clarifies this, but the user may skip this. To correct this, I think the instructions should instead recommend the phrasing "You have *exactly* 2 cards with a value of 1." In addition, there could be small reminder cards given to each player at the start of the game, with graphical examples and the revised phrasing printed on them for easy reference at any point of the game.

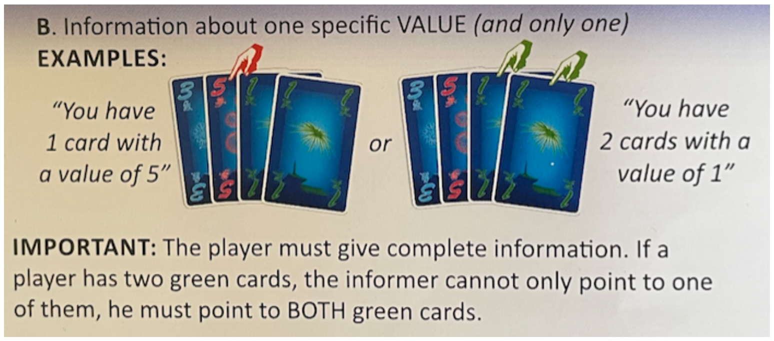
B. Information about one specific VALUE *(and only one)*
EXAMPLES:
"You have 1 card with a value of 5" *or* *"You have 2 cards with a value of 1"*
IMPORTANT: The player must give complete information. If a player has two green cards, the informer cannot only point to one of them, he must point to BOTH green cards.

Figure 3—A photo of the Hanabi rule pamphlet which explains which clues are permitted.

There are many aspects of Hanabi that make it challenging, but that don't qualify as slips or mistakes. For example, once another player gives you a clue, you need to remember that information until you play that card. If you forget, you may accidentally discard something important, or play a card at the wrong time. These actions make it harder for you to attain the highest possible score, but they aren't errors; they are not against the rules, and are an expected part of the game.

QUESTION 4

4.1 Good representation: battery charger

Last Thursday at work, I needed to swap some AAA batteries which were dead and needed recharging. I had never used the charger at work before, and was a bit apprehensive because in the past I have found battery chargers somewhat frustrating to deal with.

To my pleasant surprise, the charger that I found was very easy to use. This was because it presented me with a very good representation of its underlying content. Unfortunately, I forgot to take a photo of the charger before leaving work, and I haven't been able to find a photo of the exact model online, so I'll instead rely on a textual description.

First, it **made relationships explicit**. There were slots for 4 AAA batteries. Aligned with each battery slot and located directly above it was an LED indicator. This clearly indicated to me that the LED represented the status of the battery directly beneath it. At the time, there were batteries in all 4 slots - three LEDs were green and one was red. This made it very obvious to me that batteries could be charged separately (I've owned chargers which require that all 4 LEDs be charged at the same time, and from the same degree of discharge). Because green is the universally-accepted indicator of "go," I quickly determined that the batteries with green LEDs were fully charged, and replaced them with three of my dead LEDs. After this, all LEDs were red, the universal standard for "not good," which I assumed meant they were charging. Whenever an LED was missing from a slot, its LED was off.

Second, it **exposed natural constraints**. When I made any change to the system, the pattern of all four LEDs changed for a few seconds to display a single red indicator which appeared to move quickly from one battery indicator to the next, in a pattern. This reminded me of status spinners on graphical interfaces which indicate that the system is "thinking", and combined with the red colour, informed me that I should wait a few seconds. I quickly interpreted that the system needed to recalculate the status of all LEDs whenever a single LED was removed. I assumed this was a natural physical constraint of the system, and that I just needed to wait. I thought this spinning was an effective way of representing the fact that there was something going on with the system as a whole, and not with any single LED. If they had instead tried to hide this constraint, the behaviour may have been more confusing.

4.2 Poor representation: label maker

The reason I needed to recharge batteries in the first place was because the office label maker was displaying a low battery warning and I wanted it to go away. This indicator is presented as a text warning with the words "LOW BATTERY" that appears every time you press print. You need to acknowledge this warning before proceeding to print the label. I didn't have high confidence that the label maker actually needed new batteries because the label maker's battery indicator is not a good representation of its underlying state.

First, it **doesn't expose natural constraints**. At some point, the battery will run out and I will not be able to print labels anymore - this is a natural constraint of the system. However, when the battery indicator comes on, it does not tell me anything about how many more labels I can print. My coworker claims he once printed many tens of labels after the indicator came on, and the battery still hadn't died. The interface does not make it clear how soon I need to charge the batteries, or what the implications will be for my use of the label maker if I don't charge them now.

Second, the representation **does not adequately bring objects and relationships together**. If we consider one object to be the battery's charge, and the other object to be the ability to print a label, then the relationship between them is that the battery's charge will gradually decrease, and I will be able to print labels until the charge is below some minimum threshold. A good representation would show the battery charge gradually decreasing from a maximum to a minimum value. As a user I can infer that below the minimum, I won't be able to print a label, and the present charge lets me extrapolate how much time I have left before that point. However, the low battery indicator does not reflect this relationship at all. It is a binary signal that suddenly appears at some "low" point between the maximum and minimum value. As a user, I have no idea where the location of the minimum value is relative to my current state, so it doesn't help me know when I need to change the batteries.