# Table of Contents

[Table of Contents](#_j5nwfjhver8g)

[Meta-Data](#_wmw6vju0tvzt)

[Lesson Goals](#_y0f332pzpsd3)

[Lesson Outcomes](#_yuun61fn00lj)

[Assessments](#_qy4klj35a395)

[Lesson Plan](#_jhf5z8xjau2m)

[Script](#_x1hgyjwhsibz)

[2.3.1 Introduction to Direct Manipulation and Invisible Interfaces](#_xgus6bu1y1vv)

[2.3.1.1 Headshot Studio](#_vodcqk72ajb9)

[2.3.2 Direct Manipulation: The Desktop Metaphor 1](#_f4wizmtjvdpm)

[2.3.2.1 Tablet Studio (On Camera)](#_4yuhdk1xwhky)

[2.3.3 Direct Manipulation: The Desktop Metaphor 2](#_grbnylof0ikq)

[2.3.3.1 Tablet Studio](#_ffwoqu4gmdod)

[2.3.3.2 Tablet Studio (On Camera)](#_glq1d8wzhf0o)

[2.3.4 Paper Spotlight: “Direct Manipulation Interfaces”](#_qv1vb7rj84wj)

[2.3.4.1 Tablet Studio](#_636yq0l61v15)

[2.3.5 Direct Manipulation and Virtual Reality](#_nubze7rjcvms)

[2.3.5.1 Headshot Studio](#_1cojuh1mrfqp)

[2.3.6 Reflections: Direct Manipulation](#_t0eldeytwzho)

[2.3.6.1 Headshot Studio](#_6914n74q36nf)

[2.3.6.2 Exercise](#_6m4el4sqssmk)

[2.3.6.3 Headshot Studio](#_kd8dnvftm5lv)

[2.3.7 Babies and Direct Manipulation](#_4uc52g8benas)

[2.3.7.1 David’s House (Playroom)](#_51z18gpfuzz3)

[2.3.8 Exercise: Direct Manipulation](#_sa0m37fok5d0)

[2.3.8.1 Tablet Studio (On Camera)](#_bdbwi08nywjq)

[2.3.8.2 Exercise](#_pau8mjrpdt9d)

[2.3.8.3 Tablet Studio (On Camera)](#_hl3srllwplfs)

[2.3.9 Making Indirect Manipulation Direct](#_r0nj75q55bse)

[2.3.9.1 Tablet Studio (On Camera)](#_5d3jto90p78n)

[2.3.10 Exploring HCI: Direct Manipulation](#_n9daxe36ow0h)

[2.3.10.1 Tablet Studio (On Camera)](#_o9s06sp2roju)

[2.3.11 Invisible Interfaces](#_texxb3hgjnl7)

[2.3.11.1 Tablet Studio](#_uyw82dk5hu0j)

[2.3.12 Good Design, Bad Design](#_42zc2ywif487)

[2.3.12.1 Headshot Studio (Morgan)](#_787sb9zdor2s)

[2.3.13 Invisibility by Learning](#_gpydifls9ova)

[2.3.13.1 Headshot Studio](#_5bvwalvdbn64)

[2.3.14 Invisibility by Design](#_vsujrnljmgxe)

[2.3.14.1 Headshot Studio](#_3h97wrgvh5rm)

[2.3.15 5 Tips: Invisible Interfaces](#_i473bmbymigi)

[2.3.15.1 Headshot Studio](#_l4nzud77u700)

[2.3.16 Reflections: Invisibility by Design](#_8p3f2tymurbu)

[2.3.16.1 David’s House (Living Room)](#_wd9h86hrus8i)

[2.3.16.2 Exercise](#_siqquexi9d0m)

[2.3.16.3 David’s House](#_bjhpd5fxip8d)

[2.3.17 Design Challenge: The Universal Remote](#_s2r7zfx38zte)

[2.3.17.1 David’s House (Living Room)](#_kbiykniwua5d)

[2.3.17.2 Exercise](#_fdwpjeng8yr0)

[2.3.17.3 David’s House](#_v5p09twybxyr)

[2.3.18 Conclusion](#_bl6nc95h87ru)

[2.3.18.1 Headshot Studio](#_vwaijv7cpyeg)

# Meta-Data

## Lesson Goals

* Students will understand the power of direct manipulation in designing interfaces.
* Students will understand the idea of an invisible interface between a user and a task.
* Students will understand that invisibility may come via learning or via design, but via design is the goal.

## Lesson Outcomes

* Students will be able to identify instances of directness of manipulation in different interfaces.
* Students will be able to leverage direct manipulation in designing interfaces of their own.
* Students will be able to observe the degree to which users attend to the interface over the task in certain designs.

## Assessments

* Students will reflect on the application of the lesson’s concepts to their chosen area of HCI.
* Students will engage in a short design task based on the lesson’s concepts.
* Students will complete a short answer assignment in which they critique a provided interface from the perspective of the lesson’s concepts.
* Students will complete a short answer assignment in which they select an interface to critique from the perspective of the lesson’s concepts.
* Students will complete a short answer assignment in which they design a revision of one of the critiqued interfaces from the perspective of the lesson’s concepts.

## Lesson Plan

* Students will be introduced to the idea of direct manipulation and invisible interfaces as interaction design at its best.
* Students will then be given the desktop metaphor for operating system design, from the perspective of direct manipulation of files.
* Students will then be presented some of the challenges and opportunities of direct manipulation in emerging designs.
* Direct manipulation will be used to introduce the idea of invisible interfaces.
* Students will be introduced to methods for achieving invisible interfaces, via learning and via good design.

# Script

## 2.3.1 Introduction to Direct Manipulation and Invisible Interfaces

### 2.3.1.1 Headshot Studio

* [C] David talking
* [A] Video clips of the lesson playing to the right
* [B] Lesson; Direct Manipulation and Invisible Interfaces
* Today, we’ll talk about two applications of good feedback cycles, direct manipulation and invisible interfaces.
* [B] Definition; Direct manipulation: the user should interact directly with their task
* Direct manipulation is the principle that the user should feel as much as possible like they’re **directly** controlling the object of their task.
* For example, if you’re trying to enlarge an image on your phone, it might be better to be able to use your drag your fingers around than to tap a button marked ‘zoom in’ because you’re more directly interacting with the photo.
* [B] For example: touch screens
* **New technologies** are making it more and more possible to feel like we’re directly manipulating something even when there’s an interface in the way.
* [B] Definition; Invisible Interface: an interface that disappears between the user and their task
* At their best, the interface actually disappears, which is what we mean by an **invisible interface**.
* With an invisible interface, the user spends no time having to think about the interface they’re using: all their time is dedicated to thinking about the task that they’re performing.

## 2.3.2 Direct Manipulation: The Desktop Metaphor 1

### 2.3.2.1 Tablet Studio (On Camera)

* [C] David sitting at the desk
* Our goal is to narrow the gulf of execution and gulf of evaluation as much as possible.
* Arguably the ultimate form of this is something called direct manipulation.
* Today, direct manipulation is a very common interaction style, but in the history of HCI, it was a revolutionary new approach.
* To understand direct manipulation, let’s talk about the desktop metaphor.
* The way files and folders are stored is meant to mimic a real desktop.
* So, here on my physical desktop, I have some files.
* What do I do if I want to move them?
* <David moves them>
* I pick them up, and I move them
* What do I do if I want to put them in a folder?
* <David puts them in a folder>
* I pick them up and put them in a folder.
* I’m directly manipulating the files to make them go where I want.
* If files and folders on a computer are meant to mimic files and folders in the real world, then shouldn’t the action of moving them also mimic the real-world action of moving them?
* Wouldn’t it narrow the gulf of execution to leverage that real-world expectation and experience?

## 2.3.3 Direct Manipulation: The Desktop Metaphor 2

### 2.3.3.1 Tablet Studio

* [T] Visual of the Mac terminal -- possibly even just interacting with it live
* Files and folders on my computer are meant to mimic files and folders on my desk.
* So, we ideally want the action of moving them around on my computer to mimic the action of moving them around on my physical desk.
* It wasn’t always like this, though.
* Before graphical user interfaces were common, we moved files using command line interfaces like this.
* The folder structure is still the same on the operating system, but instead of visualizing it as folders and icons, I’m interacting with the text-based command line.
* To move files, I need to type a command like this.
* <David types the command>
* There’s nothing natural about this. I need to memorize it. If I don’t know it, there’s no immediate queue what to do, and it’s not a natural thing to do.
* It does not mimic how I directly manipulate things in person at all.
* <David minimizes the terminal>
* Then, the computer mouse came along.
* With it came the ability to move a cursor around the screen.
* Equipped with this, my action of moving files and folders becomes much more direct.
* Now, instead of typing in some cryptic command, I click on the file I want to move, and physically drag it into the folder in which I want it.

### 2.3.3.2 Tablet Studio (On Camera)

* [C] David sitting at the desk
* Moving the mouse around is a lot better than having to type in those commands, but the gulf of execution and evaluation are still present, especially for novice users.
* There’s still some interpretation that has to happen between the hand moving to the left and the cursor moving to the left.
* Clicking feels kind of like grabbing, but there’s still some interpretation there, too.
* It’s more direct than the command line, but there’s still a gap.
* Modern touch screens, though, make direct manipulation more direct than ever.
* [A] Cut to David using his phone screen if possible
* Let’s say I want to put this icon in this folder. How do I do it?
* I hold it down, then drag and move it.
* The fact that dragging it entails holding it down means that this action is nearly entirely direct manipulation.
* I don’t need any prior knowledge to attempt to do what feels natural for moving that icon into that folder.
* [A] Cut back to David sitting at the desk
* This gives us a nice general heuristic to keep in mind: how do we help the user interact most closely with the target of their task? How do we make it so they’re manipulating it as directly as possible?

## 2.3.4 Paper Spotlight: “Direct Manipulation Interfaces”

### 2.3.4.1 Tablet Studio

* [T] [Paper](http://www-ihm.lri.fr/~mbl/ENS/FONDIHM/2013/papers/Hutchins-HCI-85.pdf)
* The seminal paper on direct manipulation interfaces came out in 1985, authored by Edwin Hutchins, James Hollan, and Donald Norman.
* You’ll see two of these names again later in this course: Hutchins authored the foundational paper for distributed cognition, and Norman created one of the most accepted sets of design principles.
* In 1985, direct manipulation was starting to become a more common design strategy.
* Hutchins, Hollan, and Norman identify the two aspects of directness.
* <scroll to Distance>
* The first, distance, is the distance between the user’s goals and the system itself.
* This is the idea of gulfs of execution and evaluation that we talk about in the context of feedback cycles.
* [A] Highlight “the feeling of directness is inversely proportional to the amount of cognitive effort it takes to manipulate and evaluate a system”.
* The authors note that <read quote>.
* In other words, the more cognitive load required to use the system, the less direct the interaction feels.
* The authors break distance into two components: semantic distance and articulatory distance.
* <scroll to Figure 4>
* Semantic distance refers to the difference between the user’s goals and their expression in the system. In other words, it’s how hard it is to know what to do.
* Articulatory distance is the distance between that expression and its execution. In other words, it’s how hard it is to actually do what you know to do.
* <scroll to Figure 6>
* This is brought together here.
* <incorrectly rotate the image the wrong way>
* The user starts with some goals, translates them into their form of expression in the interface, and executes that expression. The system then returns some output in some form of expression, which is translated by the user into their understanding of the new state of the system.
* For example, when I brought up this figure, I needed to rotate the paper to display it correctly. That was my goal.
* I translated that goal into the form required in the system, a series of clicks on the right menus.
* Then, I executed those clicks. We could even break that down into smaller cycles of menus popping up.
* Then, the computer did some reasoning, and returned the output in the form of a rotated PDF.
* I then evaluated whether this was what I wanted. I discovered that it wasn’t, so I rotated it a couple more times until the output matched by goal.
* This cycle is happening constantly when you’re interacting with any interface.
* But distance is only one component. It’s possible to have interfaces with little distance that are nonetheless not examples of direct manipulation.
* Everything we’ve talked about so far is true for feedback cycles in general, not just direct manipulation.
* <scroll to direct engagement>
* What sets direct manipulation apart is the second component, direct engagement.
* [A] Highlight “the systems that best exemplify direct manipulation all give the qualitative feeling that one is directly engaged with control of the objects--not with the programs, not with the computer, but with the semantic objects of our goals and intentions.”
* The authors write that <read quote>.
* If we’re moving files, we should be physically moving the representation of the files.
* If we’re playing a game, we should be directly controlling our characters.
* If we’re navigating channels, we should be specifically selecting clear representations of the channels we want.
* That’s what takes a general feedback cycle and makes it an instance of direct manipulation.
* We can shorten the gulfs of execution and evaluation in a number of ways without direct manipulation, but direct manipulation is a powerful method for shortening that distance.

## 2.3.5 Direct Manipulation and Virtual Reality

### 2.3.5.1 Headshot Studio

* [C] David talking
* [A] Ideally, some clips of VR in action if we can grab some
* Virtual reality right now is making some incredible strides in facilitating direct manipulation in places where it hasn’t been possible before.
* Traditionally, when designers are designing stuff in 3D, they’re forced to use 2D interfaces. That translation from 2D to 3D really gets in the way of directly manipulating whatever is being designed.
* Through virtual reality, though, designers are able to view what they’re designing in 3D, as if it’s in the room there with them.
* They can rotate it with the same hand motions you would use to rotate something in person.
* They can physically move around it to get different angles on it.
* So, virtual reality is allowing us to bring the principle of direct manipulation to tasks it hasn’t touched before.
* But there’s still a lot of work to do.
* Gesture interfaces like those used in virtual reality struggle with some feedback issues.
* We aim to make the user feel like they’re physically manipulating the artifact, but when you’re working with something physically, it pushes back against your hands.
* How do we recreate that in virtual reality?

## 2.3.6 Reflections: Direct Manipulation

### 2.3.6.1 Headshot Studio

* [C] David talking
* Take a moment real quick and reflect on some of the tasks you perform with computers day to day.
* What are some places where you do not interact through direct manipulation?
* If you’re having trouble thinking of one, think especially about places where technology is replacing things you used to do manually.
* Chances are, the physical interface was a bit closer to the task than the new technical one.
* How can the technical interface better leverage direct manipulation?

### 2.3.6.2 Exercise

* [E] What is an example of a task you perform regularly that does not leverage the principle of direct manipulation?
* [E] (box for student’s answer)

### 2.3.6.3 Headshot Studio

* [C] David talking
* So, when I was writing the script for this exact morsel, I was interrupted by a text message from a friend of mine.
* In the reply I was writing, I wanted to include a smiley face emoji -- we know that using emojis and emoticons tends to humanize textual communication.
* On my phone, the interface for entering an emoji is to tap an icon to bring up the list of all emojis, then select the one you want.
* But when I’m reacting to someone in conversation, I’m not mentally scrolling through a list of emotions. I’m just reacting naturally.
* Why can’t my phone capture that?
* Instead of having to select ‘smiling’ from a list of emojis, maybe my phone could just have a button to insert the emoji corresponding to my current facial expression, so to wink I could just wink or to frown I could just frown.
* It wouldn’t capture every possible face, but for the most commonly used ones, it might be more efficient.

## 2.3.7 Babies and Direct Manipulation

### 2.3.7.1 David’s House (Playroom)

* [C] David with Lucy
* There may be no better example of the power of direct manipulation than watching a baby using an interface.
* [C] Lucy on her tablet
* My daughter Lucy is 18 months old. Yet when I give her an interface that uses direct manipulation, she’s able to use it.
* She wouldn’t be able to use a keyboard or a mouse yet, but because she’s directly interacting with things on the screen, she can use it.
* ...wait, actually. There’s an even better example.
* [C] Cats on a tablet
* There are games made for tablet computers for cats.
* Yes, cats can use a tablet computer when it uses direct manipulation.

## 2.3.8 Exercise: Direct Manipulation

### 2.3.8.1 Tablet Studio (On Camera)

* [C] David at the touchpad
* Let’s try a quick exercise on direct manipulation.
* The Mac touchpad is famous for having lots of different kinds of interactions.
* <run through interaction styles>
* Which of these are examples of direct manipulation?

### 2.3.8.2 Exercise

* [E] “Which of these Mac touchpad gestures are examples of direct manipulation?”
* [E] A. Pressing down to click
* [E] B. Pressing two fingers down to right-click
* [E] C. Dragging two fingers up and down to scroll
* [E] D. Double-tap to zoom in and out
* [E] E. Pinching to zoom in and out

### 2.3.8.3 Tablet Studio (On Camera)

* [C] David at the touchpad
* There’s some room for disagreement here, but I think these five are pretty cut and dry.
* We can think of whether these are direct manipulation based on whether the same actions would make sense if we were able to do it directly on the screen.
* For clicking, yes: I would consider that direct manipulation because usually I would just tap on the screen.
* For right-clicking, though: no, there’s nothing direct about two fingers operating in this way.
* For scrolling, that makes sense, the way I drag my fingers mirrors the way the document moves.
* For zooming, though, there’s similarly little direct about double-tapping to zoom in and out.
* Pinching, however, makes sense because the content would stay under the fingers as they move.

## 2.3.9 Making Indirect Manipulation Direct

### 2.3.9.1 Tablet Studio (On Camera)

* [C] David talking at the touchpad
* The Mac touchpad has some interesting examples of how you can make indirect manipulation direct.
* For example, if I swipe left from the right edge, it brings up this notification center.
* This feels direct because the notification center pops up on the right, right where I swiped it.
* The touchpad is like a miniature version of the screen.
* But they could have placed the notification center anywhere.
* This isn’t like scrolling where there’s something more fundamental about the content that demands a certain kind of interaction.
* But by placing it there *and* using this interaction to bring it up, this feels more direct.
* Animation can accomplish this, too.
* I can clear the windows by making this motion, starting with my fingers together and spreading them apart.
* While that’s kind of like clearing off your desk, I would argue it’s not close enough to feel that direct, *except* the animation on screen mimics the action as well.
* The windows could have faded away or slid to the bottom and still accomplished the same function.
* The animation they chose, however, makes the interaction feel more direct.
* The same thing applies to the launchpad, which we bring up with the opposite interaction.
* The animation makes it look like we’re zooming out a bit to see the launchpad come into view, just as the gesture is similar to the zooming out gesture.
* So, direct manipulation isn’t just about designing interactions that feel like you’re directly manipulating the interface. It’s also about designing interfaces that lend themselves to interactions that feel direct.

## 2.3.10 Exploring HCI: Direct Manipulation

### 2.3.10.1 Tablet Studio (On Camera)

* [C] David talking
* Depending on the area of HCI you chose to explore, direct manipulation might be a big open question.
* For gesture-based interfaces, for example, you’re generally not actually touching anything.
* Direct manipulation is contingent on immediate feedback that maps directly to the interaction, though, so how do you create that in a gesture-based interface?
* This is a big challenge for virtual reality as well. Virtual reality thrives on making you feel like you’re somewhere else visually and auditorily, but it has a long way to go kinesthetically.
* How do you create the feeling of direct manipulation based on physical action when you can only give feedback visually and auditorily?
* So, take a second and reflect on how these principles of direct manipulation apply to your chosen area of HCI.

## 2.3.11 Invisible Interfaces

### 2.3.11.1 Tablet Studio

* [V] Feedback cycle diagram
* Whether through using direct manipulation, through innovative approaches to shrinking these gulfs, or through the user’s patience and learning, our ultimate goal is for the interface between the user and the task to become invisible.
* [V] Interface shrinks and disappear
* What this means is that even though there is an interface in the middle, the user spends no time thinking about it.
* Instead, they feel as if they are interacting directly with the task.
* [T] Switch to Sketchbook
* For example, if I take a stylus and write on this tablet, I’m interacting with an interface translating my drawing into data inside the system.
* But for me, it feels just as if I’m drawing on a page.
* The interface has become largely invisible between the user, me, and the task, drawing.
* Contrast that with trying to draw with the mouse.
* I’m very well aware of the mouse as the interface between myself and the task.

## 2.3.12 Good Design, Bad Design

### 2.3.12.1 Headshot Studio (Morgan)

* [C] Morgan writing on a tablet computer
* Good design: interfaces that are metaphorically invisible.
* [C] Morgan writing on an invisible tablet computer
* Bad design: interfaces that are literally invisible
* Well, kind of.
* Gesture-based interfaces are, in one sense, literally invisible.
* That’s actually why it’s so important for them to give great feedback, because otherwise it’s tough to gauge the success of a gesture interaction.

## 2.3.13 Invisibility by Learning

### 2.3.13.1 Headshot Studio

* [C] David talking with a prop steering wheel available
* We shouldn’t fall into the trap of assuming that just because an interface has become invisible, the design is great.
* Interfaces become invisible not just through great design, but also through users learning to use them.
* With enough practice and experience, many users will become sufficiently comfortable with many interfaces to feel invisibly integrated into the task.
* Take driving, for example.
* Let’s say I’m driving a car and I’m heading right for something.
* In fact, let’s try this -- Amanda, come here!
* <Amanda walks over>
* So, I’m in a car, and I suddenly realize I’m barreling toward her.
* What’s my reaction?
* <David turns the wheel and pumps the imaginary brake>
* I turn the wheel and slam on the brakes.
* It’s instinctive, I do it immediately.
* But think about that action.
* <David puts down the steering wheel>
* If I was running down the street and suddenly saw Amanda in front of me, would it be natural for me to go like this?
* <David pretends to turn the steering wheel>
* Of course not.
* The steering wheel was an interface I use to turn to the left, but it’s become invisible during the task of driving based on all my practice with it.
* But just because the interface has become invisible doesn’t mean it’s a great interface.
* People spend months learning to drive, pay hundreds of dollars, and pass a complicated test.
* Driving is important enough that it can have that kind of learning curve.
* But for the interfaces we design, we generally cannot expect users to give us an entire year to learn to use them.
* We’ll be lucky if users give us an entire minute.
* So, our goal is to make our interfaces invisible by design.

## 2.3.14 Invisibility by Design

### 2.3.14.1 Headshot Studio

* [C] David talking
* Our goal is to create interfaces that are invisible from the moment the user starts using them.
* They should feel immediately as if they’re interacting with the task underlying the interface.
* Now, this is an extremely tall order, and one we probably won’t meet very often, but it’s the goal.
* In fact in my opinion, this is why people tend to underestimate the complexity of HCI: when you do things right, people won’t realize you’ve done anything at all.
* So, how do we create interfaces that are invisible from the first moment the user starts using them?
* That’s precisely what we’ll discuss in a lot of our conversations about HCI.
* We’ll talk about some principles for creating interfaces that disappear, like leveraging prior expectations and providing quick feedback.
* We’ll also talk a lot about how to get inside the user’s head and understand what they’re seeing when they look at an interface so that we can help their internal model match the system.
* In fact, if we consider ‘invisiblity’ to be a hallmark of usable design, then this entire course could be retitled ‘Creating Invisible Interfaces’.

## 2.3.15 5 Tips: Invisible Interfaces

### 2.3.15.1 Headshot Studio

* [C] David talking
* Here are 5 tips for designing invisible interfaces.
* [B] 1. Use affordances.
* **1. Use affordances.** We talk more about affordances when we discuss design principles. Affordances are places where the visual design of the interface suggests how it is to be used. Buttons are for pressing, dials are for turning, switches are for flicking. Use these expectations to make your interface more usable.
* [B] 2. Know your user.
* **2. Know your user.** Invisibility means different things to different people. Invisibility to a novice means that all the interactions is natural, but invisibility to an expert means maximizing efficiency. Know for whom you’re trying to design.
* [B] 3. Differentiate your user.
* **3. Differentiate your user.** Maybe you’re designing for both novices and experts. If that’s the case, provide multiple ways of accomplishing tasks. For example, having copy and paste under the ‘Edit’ menu keeps those options discoverable, but providing Ctrl+C and Ctrl+V as shortcuts keeps those actions efficient.
* [B] 4. Let your interface teach.
* **4. Let your interface teach.** When we think of teaching users how to use our software, we usually think of tutorials or manuals, but ideally the interface itself will do the teaching. For example, when users select copy and paste from the edit menu, they see the hotkey that corresponds to that function. The goal is to teach them the more efficient way of performing those actions without requiring them to already know it in order to work.
* [B] 5. Talk to your user.
* **5. Talk to your user.** We’ll say this over and over again, but the best thing you can do is talk to the user. Ask them what they’re thinking while they use an interface. Note especially whether they talk about the task or the interface: if they’re talking about the interface, then it’s pretty visible.

## 2.3.16 Reflections: Invisibility by Design

### 2.3.16.1 David’s House (Living Room)

* [C] David sitting on the couch
* Reflecting on where we’ve encountered invisible interfaces is difficult because… well, they were invisible. What makes them so good is the fact that we didn’t notice them.
* But give it a try anyway. Try to think of a time when you picked up a new interface for the first time and immediately knew exactly how to use it to accomplish the task you had in mind.

### 2.3.16.2 Exercise

* [E] What is a time when you picked up a new interface and knew immediately how to use it to accomplish your goal?
* [E] (box for student answer)

### 2.3.16.3 David’s House (Living Room)

* [C] David sitting on the couch
* One of my favorite examples of an interface that is invisible by design comes from a video game, Portal 2.
* In lots of video games, you use a control stick to control the camera.
* But different people have different preferences about how the camera should behave.
* Some feel that when you press Up, you should look Up.
* Some feel when you press Up, you should look down, as if you’re moving the camera up.
* In most games, you have to set this manually by going to Options, selecting Camera Controls, and enabling or disabling Invert Y Axis.
* In Portal 2, though, watch what happens.
* <Clip from Portal 2, where you’re asked to look up.>
* A character in the game asked me to look up. The game assumed whichever direction I pressed was the way I would press when I want to look up, and set my preference accordingly.
* No options screen, no changing settings, the game automatically and invisibly had me complete my goal of correctly setting up my camera preference.

## 2.3.17 Design Challenge: The Universal Remote

### 2.3.17.1 David’s House (Living Room)

* [C] David on the couch
* For this design challenge, let’s tackle one of the most common problems addressed in undergraduate HCI classes: designing a better remote control.
* This is probably not a very good interface.
* That’s not to say it’s poorly designed, but the constraints on how many things this remote must do and how limiting this physical structure is make it difficult to use.
* You might have seen humorous images of people online putting tape over certain buttons to make remotes easier to use for their parents or kids.
* How would we design an invisible interface for a universal remote control, one that doesn’t require the learning curve that this thing has?

### 2.3.17.2 Exercise

* “Click to continue” exercise

### 2.3.17.3 David’s House (Living Room)

* [C] David on the couch
* Personally, I think that this is a great candidate for a voice interface.
* In fact, Comcast, Amazon, and others have already started experimenting with remote interfaces for remote controls.
* One of the challenges with voice interfaces is that the commands aren’t very discoverable. Generally, if you don’t know what you can say, you have no easy way of finding out.
* But watching TV and movies is such a normal part of our conversations that we already have a vocabulary of how to say what we want to do.
* The challenge is for us designers to make a system that can understand that vocabulary.
* When I saw, “watch Community”, it understands that Community is a TV show, and it tries to figure out: do I grab it from the DVR? Do I grab it from OnDemand? Do I see if it’s on live?
* The vocabulary for the user was very natural.
* For example, “watch Conan”
* <play first ~15 seconds of David’s Conan video>
* I had to put that in somewhere

## 2.3.18 Conclusion

### 2.3.18.1 Headshot Studio

* [C] David on left
* [A] Clips of earlier videos playing on right
* [B] Topic; Direct manipulation
* [B] Topic; Invisible interfaces
* Today we’ve talked about two applications of effective feedback cycles, **direct manipulation** and **invisible interfaces**.
* We talked about how interfaces are most effective when the user has a sense that they’re directly manipulating the object in their task.
* We talked about how modern technology, like touchscreens and virtual reality, are making it possible for manipulation to feel more and more direct.
* We talked about how the most effective interfaces become invisible between the user and the task, such that the user spends no time thinking about the interface itself at all.
* And, we talked about how interfaces can become invisible via either learning or design, and we’re most interested in designing them to become invisible.
* To a large extent, that’s the definition of usable design: designing interfaces that disappear between the user and their task.