

## Part VI

# The human face of computing—*Interacting with computers*



---

Why are computers so hard to get along with? Everyone you ask will tell you stories about how difficult computers are to use, how they never seem to do what you really want them to, how they keep going wrong and make ridiculous mistakes. Computers seem to be made for wizards, not for ordinary people. But they *should* be made for ordinary people, because computers are everyday tools that help us to learn, work, and play better.

The part of a computer system that you interact with is called its “user interface.” It’s the most important bit! Although you might think of what the program actually *does* as the main thing and the user interface as just how you get into it, a program is no good at all if you can’t interact with it and make it do what you want. User interfaces are very difficult to design and build, and it has been estimated that when writing programs, far more effort goes into the interface than into any other part. Some video games have excellent user interfaces, interfaces that need no complicated instructions and become almost invisible as you are drawn into the game. Countless software products which are otherwise very good have been complete flops because they have strange user interfaces. And whole industries have been built around a clever interface idea—like the word processor or spreadsheet—that promotes access to computational functions which are really quite elementary in themselves.

But why do we have to have user interfaces at all? Why can’t we just talk to our computers the way we do to our friends? Good question. Maybe someday we will; maybe not. But certainly not yet: there are big practical limitations on how “intelligent” computers can be today. The activities that follow will help you understand the problems of user interface design, and help you to think more clearly about the limitations of computers and be wary of the misleading hype that is often used to promote computer products.

## For teachers

Computing is not so much about calculation as it is about *communication*. Computing *per se* really has no intrinsic value; it is only worthwhile if the results are somehow communicated to the world outside the computer, and have some influence there. And a lot of the activities in this book are about communication. *Representing data* (Part I) shows how different kinds of information can be communicated to a computer or between computers. *Representing processes* (Part III) is about how to communicate processes to a computer to tell it how to accomplish certain tasks—after all, “programming” is really only explaining to a computer, in its own language! *Cryptography* (Part V) is about how to communicate in secret, or to communicate bits of secrets without revealing all.

The activities that follow are about how people communicate with computers. While the rest of the book is based on well understood technical ideas, this part is not. That makes it both easier, in that no special knowledge is required of the children, and more difficult, in that a certain level of maturity is needed to understand what the activities are about and relate them to a broader context. These activities contain more detailed explanations than most of the others because it is necessary to give you, the teacher, enough background material to be in a position to help draw out some of the implications in class discussion.

There are two activities in this section. The first is about the area known as the “human–computer interface,” commonly abbreviated to HCI. In order to “unplug” this aspect of

---

computing without depending on prior knowledge of a particular example of a computer system, we have invented a design exercise that does not really involve computers—but does introduce fundamental principles that are used in the design of human–computer interfaces. Because human interface design is culture-dependent, there are not necessarily any “right” answers in this activity, which may frustrate some children. The second activity is about the area known as “artificial intelligence,” or AI. It involves a guessing game that stimulates children into thinking about what computers can and can’t do.

## **For the technically-minded**

Human–computer interaction is fast becoming one of the hottest research areas in computer science as people realize how much the success of a software product depends on its user interface. The subject draws heavily on a wide range of disciplines outside computer science, such as psychology, cognitive science, linguistics, sociology—even anthropology. Few computer scientists have training in these areas, and HCI represents an important growth area for people who are interested in the “softer” side of the subject.

Artificial intelligence is a topic that often raises hackles and causes disputes. In this book we have tried to steer a middle path between AI aficionados who believe that intelligent machines are just around the corner, and AI sceptics who believe that machines cannot in principle be intelligent. Our goal is to encourage children to think independently about such issues, and to promote a balanced view.

The activities here draw heavily on two eminently readable books, Don Norman’s *The design of everyday things* and John Haugeland’s *Artificial intelligence: the very idea*, which we enthusiastically recommend if you are interested in pursuing these issues further.

Computers involve another important kind of communication, one that is not touched upon in this book: communication between people who are building a computer system. Students who learn about computers and make their way into the job market—perhaps having graduated in computer science from university—are invariably surprised by how much interpersonal communication their job entails. Computer programs are the most complex objects ever constructed by humankind, with millions or perhaps billions of intricately interlocking parts, and programming projects are tackled by close-knit teams that work together and spend a great deal of their time communicating. Once the product is complete, there is the job of communicating with customers through user manuals, courses, “help” phonelines, and the like—not to mention the problem of communicating with potential customers through demonstrations, displays, and advertising. We haven’t yet found a way to realistically “unplug” for children the interpersonal communication aspect of computing, so this book doesn’t address it. But it is the kind of thing that computer professionals who are visiting a classroom may be able to describe from their own experience and bring out in discussion.

## Activity 19

# The chocolate factory—*Human interface design*

**Age group** Middle elementary and up.

**Abilities assumed** No specific abilities required.

**Time** An hour or more.

**Size of group** From small groups to the whole classroom.

### Focus

Design.

Reasoning.

Awareness of everyday objects.

### Summary

The aim of this activity is to raise awareness of human interface design issues. Because we live in a world where poor design is rife, we have become accustomed (resigned?) to putting up with problems caused by the artifacts we interact with, blaming ourselves (“human error,” “inadequate training,” “it’s too complicated for me”) instead of attributing the problems to flawed design. The issue is greatly heightened by computers because they have no obvious purpose—indeed, they are completely *general* purpose—and their appearance gives no clues about what they are for nor how to operate them.

## Technical terms

Interface design; affordances; mapping; transfer effects; population stereotypes; icons; user interface evaluation

## Materials

Each group of children will need:

a copy of the blackline master on pages 209 and 210, and

a copy of the images on page 211, either on overhead projector transparency or on cards that can be displayed to the class, and

one or more of the six cards contained in the blackline master on page 212. Cut the sheet into individual cards and divide them between the groups.

## What to do

The great chocolate factory is run by a race of elf-like beings called Oompa-Loompas.<sup>1</sup> These Oompa-Loompas have terrible memories and no written language. Because of this, they have difficulty remembering what to do in order to run the chocolate factory, and things often go wrong. Because of this, a new factory is being designed that is supposed to be very easy for them to operate.

1. Divide the children into small groups and explain the story.
2. The first problem the Oompa-Loompas face is getting through the doors carrying steaming buckets of liquid chocolate. They cannot remember whether to push or pull the doors to open them, or slide them to one side. Consequently they end up banging into each other and spilling sticky chocolate all over the place. The children should fill out the “doors” worksheet on page 209. More than one box is appropriate in each case. For some of the doors (including the first one) it is not obvious how to open them, in which case the children should record what they would try first. Once they have filled out their own sheets, have the whole group discuss the relative merits of each type of door, particularly with regard to how easy it is to tell how it works, and how suitable it would be to use if you are carrying a bucket of hot chocolate. Then they should decide what kind of doors and handles to use in the factory.

Follow this activity with a class discussion. Table 19.1 comments briefly on each door in the worksheet. Real doors present clues in their frames and hinges as to how they open, and there are conventions about whether doors open inwards or outwards. Identify the kinds of door handles used in your school and discuss their appropriateness (they may be

---

<sup>1</sup>With apologies to Roald Dahl. You’ll know about the Oompa-Loompas if you’ve read his wonderful tale *Charlie and the Chocolate Factory*. If not, never mind: the plot is not relevant to this activity.

Plain door	Can't see how to open this one at all, except that since it has no handle, it must require pushing rather than pulling.	Labeled door	The label is like a tiny user manual. But should a door need a user manual? And the Oompa Loompas can't read.
Hinge door	At least you can see which is the side that opens.	Bar door	It's clear that you are supposed to push the bar, but which side?
Handle door	Handles like this are usually for pulling—or sliding.	Knob door	The knob shows what to grasp, but not whether to push or pull; it probably doesn't slide.
Panel door	It's clear that you push this. What else could you do?	Glass door	The small vertical bar on this side signals "pull"; the longer horizontal one on the other signals "push".
Sliding door	This one's only for sliding.		

Table 19.1: About the doors in the worksheet

quite *inappropriate!*) Do doors normally open inwards or outwards into corridors?—and why? (Answer: They open into rooms so that when you come out you won't bash the door into people walking along the corridor, although in some situations they open outwards to make evacuation easier in an emergency.)

The key concept here is what are called the *affordances* of an object, which are its visible features—both fundamental and perceived—whose appearance indicates how the object should be used. Affordances are the kinds of operation that the object permits, or “affords.” For example, it is (mostly) clear from their appearance that chairs are for sitting, tables are for placing things on, knobs are for turning, slots are for inserting things into, buttons are for pushing. And computers are for ... what? They have no affordances that indicate their functionality, apart from very low-level ones such as input (e.g. keyboard) and output (e.g. screen) capabilities.

Doors are very simple objects. Complex things may need explaining, but simple things should not. When simple objects need pictures, labels, or instructions, then design has failed.

3. The pots containing different kinds of chocolate have to cook at different temperatures. In the old chocolate factory the stoves were as shown in the blackline master on page 210. The left-hand knob controlled the rear left heating element, the next knob controlled the front left element, the next one controlled the front right, and the right-hand knob controlled the rear right element. The Oompa-Loompas were always making mistakes, cooking the chocolate at the wrong temperature, and burning their sleeves when reaching across the elements to adjust the controls. The children should recall how the controls are laid out on their cookers at home and come up with a better arrangement for the new factory.

Follow this activity with a class discussion. Figure 19.1 shows some common arrangements. All but the one at the lower left have the controls at the front, to avoid having to reach across the elements. In the design at the top left, there are so many possible mappings from controls to burners (24 possibilities, in fact) that eight words of labeling are needed. The “paired” arrangement in the top center is better, with only four possible mappings (two for the left cluster and two for the right); it requires just four labeling words. The design at the top right specifies the control–burner relationship diagrammatically rather than linguistically (which is good for the Oompa-Loompas!). The lower three designs need no labels. The left-hand one has a control by each burner, which is awkward and dangerous. The other two involve relocating the burners slightly, but for different reasons: in the center design they are moved to leave room for the controls, while in the right-hand one they are rearranged to make the correspondence clear.

The key concept here is the *mapping* of controls to their results in the real world. Natural mapping, which takes advantage of physical analogies and cultural standards, leads to immediate understanding. The spatial correspondences at the bottom of Figure 19.1 are good examples—they are easily learned and always remembered. Arbitrary mappings, as in the top arrangements, need to be labeled, or explained and memorized.

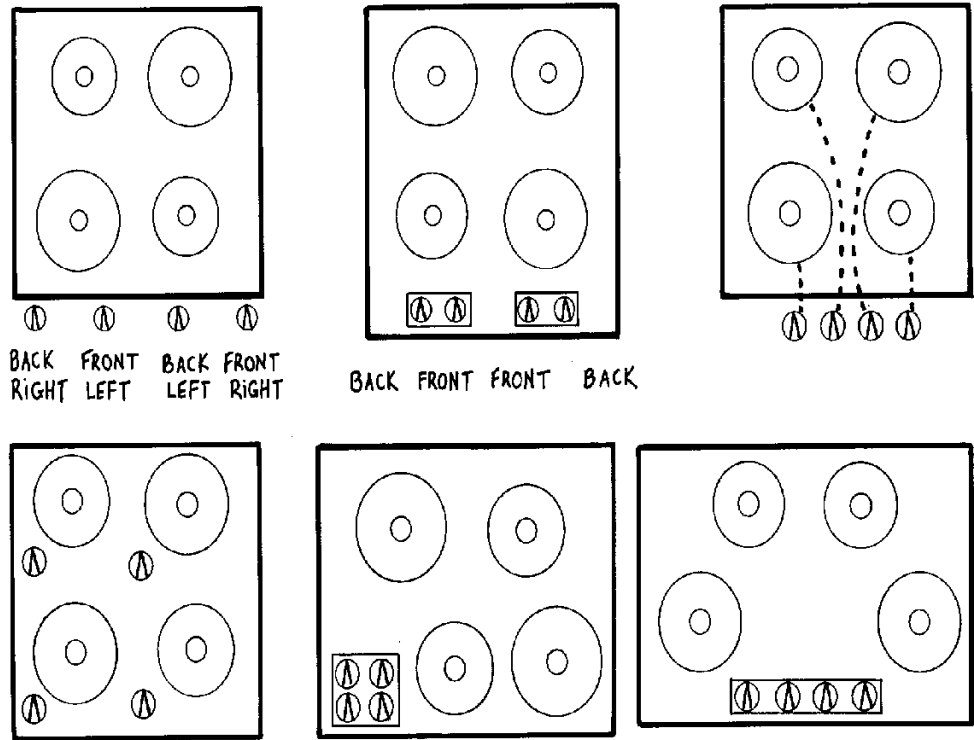


Figure 19.1: Some possible cooker layouts

4. The factory is full of conveyer belts carrying pots of half-made chocolate in various stages of completion. These conveyer belts are controlled manually by Oompa-Loompas, on instructions from a central control room. The people in the control room need to be able to tell the Oompa-Loompa to stop the conveyer belt, or slow it down, or start it up again.

In the old factory this was done with a voice system: the control room person's voice came out of a loudspeaker by the conveyer belt controls. But the factory was noisy and it was hard to hear. The groups should design a scheme that uses visual signals.

One possibility is to put in lights to signal *Stop!*, *Slow down* and *Start up*. These should follow the normal traffic-light convention by using red for *Stop!*, yellow for *Slow down* and green for *Start up*. They should be arranged just like traffic lights, with red at the top and green at the bottom.

But now reveal to the class that in Oompa-Loompa land, traffic lights work differently from the way they do for us: yellow means stop, red means go, and lights go green to warn people that they will soon have a stop light. How does this affect things? (Answer: the factory should follow the Oompa-Loompa's traffic-light convention—we should not try to impose our own.)

The key concepts here are those of *transfer effects*—people transfer their learning and expectations of previous objects into new but similar situations—and *population stereotypes*—different populations learn certain behaviours and expect things to work in a certain way. Although the traffic light scenario may seem far-fetched (though nothing is all *that* far-fetched in Oompa-Loompa land), there are many examples in our own world: in America light switches are on when they are up and off when they are down, whereas in Britain the reverse is true; calculator keypads and touchtone phones are laid out in different ways; and number formats (decimal point or comma) and date formats (day/month/year or month/day/year) vary around the world.

5. When one shift of Oompa-Loompas finishes work in the chocolate factory, they must clean up and put away pots and pans and jugs and spoons and stirrers ready for the next shift. There is a cupboard with shelves for them to put articles on, but the next shift always has trouble finding where things have been put away. Oompa-Loompas are very bad at remembering things and have trouble with rules like “always put the pots on the middle shelf,” “put the jugs to the left.”

The groups of children should try to come up with a better solution.

Figure 19.2 shows a good arrangement (which is sometimes used—but for rather different reasons—on yachts and other places where it is necessary to stop things sliding around). The key concept here is to use *visible constraints* to make it obvious where everything is supposed to go. It is clear from the size and shape of each hole which utensil it is intended for: the designer has made the constraints visible and used the physical properties of the objects to avoid the need to rely on arbitrary conventions.

6. In the main control room of the chocolate factory there are a lot of buttons and levers and switches that operate the individual machines. These need to be labeled, but because the

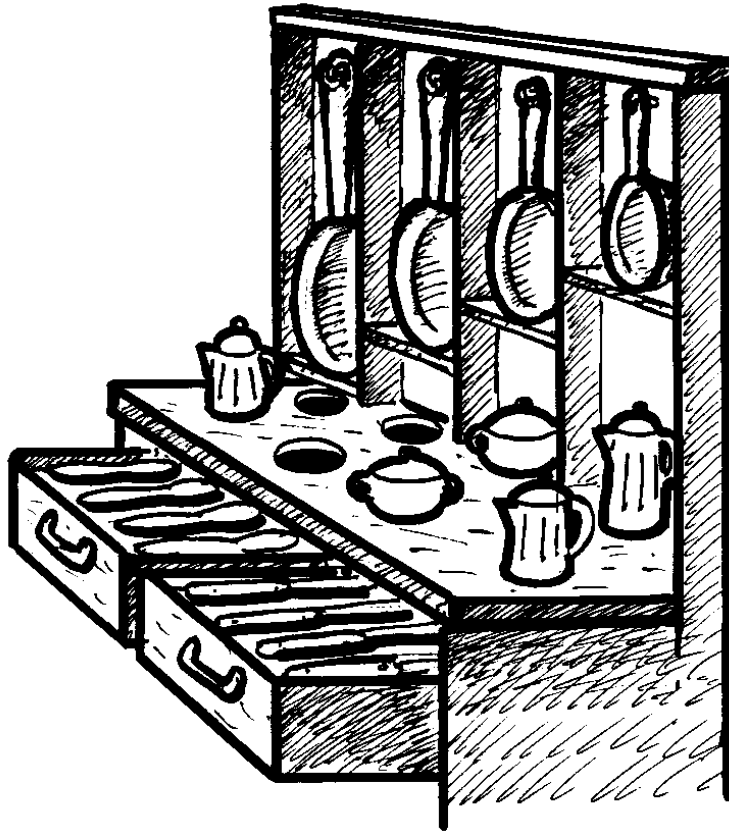


Figure 19.2: Cupboard design that utilizes visible constraints

Oompa-Loompas can't read, the labels have to be pictorial—iconic—rather than linguistic.

To give the children a feeling for icons, page 211 shows some examples. The children should identify what the icons might mean (for example, the letter going into a mailbox might represent sending a message). There are no “correct” answers to this exercise; the idea is simply to identify possible meanings.

7. Now let's design icons for the chocolate factory. The cards on page 212 specify clusters of related functions, and each group of children receives one or more cards without the other groups knowing what they are. A control panel is to be designed for the function clusters that contains individual icons for each of the five or six operations. The groups then show their work to the other children, without saying what the individual operations are, to see if they can guess what the icons mean. Encourage the use of imagination, color, and simple, clear icons.

## TIMER RECORDING

ALL YOU HAVE TO DO IS TO SET THE TIMER CORRECTLY AND YOU CAN LEAVE THE HOUSE WITHOUT MISSING VITAL PROGRAMMES

This function makes it possible to automatically turn the unit on and start recording on a preset day and time, and turn the unit off after another preset time.

- CHECK THE FOLLOWING ITEMS BEFORE SETTING TIMER RECORDING:**
- 1 Day of the week required
  - 2 Start and stop times
  - 3 Channel
  - 4 Length of cassette

- PREPARATION**
- Load a blank cassette which has not had the tab removed.
  - Make sure that the **TIMER SWITCH** is OFF
  - Make sure that the **TIMER DISPLAY** shows the correct present time.

- NOTE**
- The cassette cannot be removed during timer recording.
  - Remove after turning off the **TIMER SWITCH**
  - Start automatic recording at least 20 seconds before the desired time, then stop automatic recording at desired stop time.
  - Set the **TIMER SWITCH** to OFF if you wish to stop the timer recording or after the timer recording is finished. If stopped in this way, the **TIMER** mode will be retained in memory.
  - After setting the **TIMER** switch to ON, the unit is automatically turned off.
  - To turn the unit on, set the **TIMER** button to OFF. If a cassette without tab is inserted, the cassette will be automatically ejected.
  - When the tape comes to its end during **CTR** the VCR will turn itself off. The tape will not automatically rewind.

### TV RECEIVER

- 1 Turn TV on.
- 2 Select the chosen video channel.
- 3 Carry out steps 1 to 8 of the recorder operation.
- 4 Turn TV OFF.

### RECORDER

- 1 Turn the unit ON.
- 2 Set the **REC SELECTOR** to **TUNER** position.
- 3 Turn the **VIDEO/TV** select button to **VIDEO** position.
- 4 Set the **START TIME** and **RECORDING LENGTH TIME** for timer recording.
- 5 Select the channel to be recorded.
- 6 If the TV signal to be recorded is weak, turn the **COLOUR/AUTO** button to the **COLOUR** position, otherwise set it to the **AUTO** position.
- 7 Turn the unit OFF.
- 8 Turn the **TIMER SWITCH** to ON. (The **TIMER LED** lights up to indicate that the **TIMER** is on.)

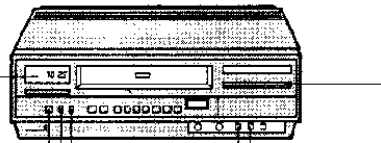
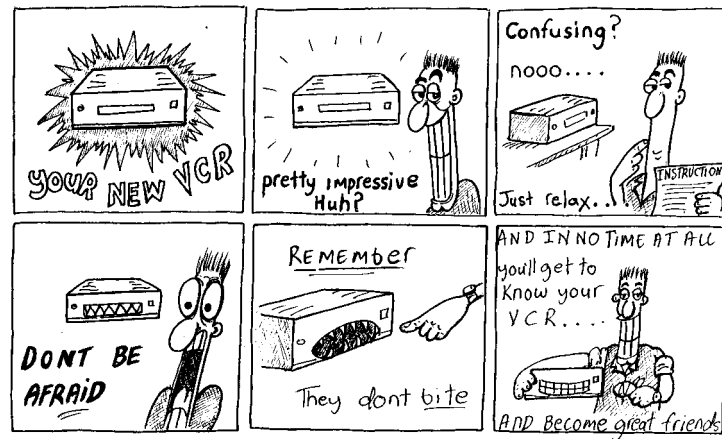


Figure 19.3: Instructions for an early VCR (from *Human-Computer Interaction*, by Preece *et al.*, reproduced by permission of The Open University)

## Variations and extensions

Can the children set the time on their electronic wristwatch? The mappings involved in the cooker layouts were simple because there were four controls for four burners. More difficulty occurs whenever the number of actions exceeds the number of controls. But the controls on wristwatches are often exceedingly complex. (“You would need an engineering degree from MIT to work this,” someone looking at his new wristwatch once told Don Norman, a leading user interface psychologist. Don *has* an engineering degree from MIT, and, given a few hours, he *can* figure out the watch. But why should it take hours?)

VCRs pose an even more significant problem to most users, particularly adults (who are, after all, the principal customers). Try leading a class discussion on how the children’s VCRs are controlled. But beware: although it is likely that they are very badly designed from a human interface point of view, the children may have become very adept at using them and may find it



hard to see the problems (they say that if you see a VCR whose clock is not flashing 12:00, it's a sure indication that the household contains teenagers.) Figure 19.3 shows a set of instructions and schematic diagram for an early VCR; from it the class could attempt to map a detailed list of instructions, in sequence, for recording a TV programme. Re-design the controls as a class exercise—the children could hardly do worse than this!

## What's it all about?

Human-computer interaction is about designing, evaluating, and implementing computer systems that allow people to carry out their activities productively and safely. In the old days, computers were for specialists and the users could be expected to be highly educated and specially trained in their use. But now computers are everyday tools that we all must use, and far greater attention must be paid to the human interface.

Many disasters, some involving loss of life, have occurred because of inadequate interfaces: airplane crashes and even shoot-downs of civilian airplanes, freeway pile-ups because of errors in switching remotely-operated highway signs, nuclear power station disasters. On a smaller scale, most people experience frustration—often extreme frustration (a police officer once fired bullets into his computer screen)—with computers and other high-tech devices every day in the workplace. And it is not just computers: what about those shrink-wrapped packages that you could only open if you had sharp claws or a hooked beak, doors that hurt your wrist as you try to push your way through, milk cartons that always splash you when you open them, elevators where you can't see how you're supposed to push the button, home entertainment systems whose advertisements claim to do everything, but make it almost impossible to do anything?

We are becoming used to “human error” and to thinking of ourselves as somehow inadequate; people often blame themselves when things go wrong. But many so-called human errors are actually errors in design. People have information-processing limitations and designers need



“The only reason we allow him inside is because he’s the only one that can work the VCR.”

to account for these; bad design cannot be rectified by producing a detailed and complicated user manual and expecting people to study it intensively and remember it forever. Also, humans are fallible and design needs to take this into consideration.

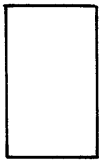
Interface *evaluation* is an essential part of the design process. The present activity has involved some evaluation when the children tested their icon designs on others. A more thorough evaluation would test the design on real Oompa-Loompas (who may perceive icons differently) in a carefully-controlled psychology-style experiment.

Although the problems caused by technology—particularly VCRs!—form the butt of many jokes, human interface design is by no means a laughing matter. Inadequate interfaces cause problems ranging from individual job dissatisfaction to stock-market disasters, from loss of self-esteem to loss of life.

### **Further reading**

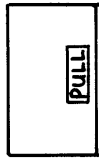
Don Norman’s book *The design of everyday things* is a delightful—and liberating—account of the myriad design problems in everyday products. Jenny Preece’s encyclopedic *Human-computer interaction* is a very comprehensive account of this multidisciplinary field. We have drawn extensively on both of these sources when preparing this activity.

PLAIN DOOR



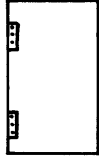
- Push  Left side
- Pull  right side
- slide it along

LABELED DOOR



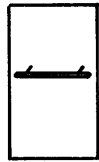
- Push  Left side
- Pull  right side
- slide it along

HINGE DOOR



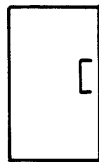
- Push  Left side
- Pull  right side
- slide it along

BAR DOOR



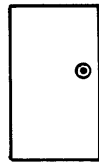
- Push  Left side
- Pull  right side
- slide it along

HANDLE DOOR



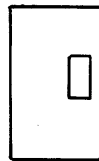
- Push  Left side
- Pull  right side
- slide it along

KNOB DOOR



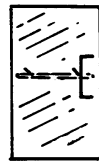
- Push  Left side
- Pull  right side
- slide it along

PANEL DOOR



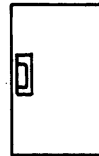
- Push  Left side
- Pull  right side
- slide it along

GLASS DOOR



- Push  Left side
- Pull  right side
- slide it along

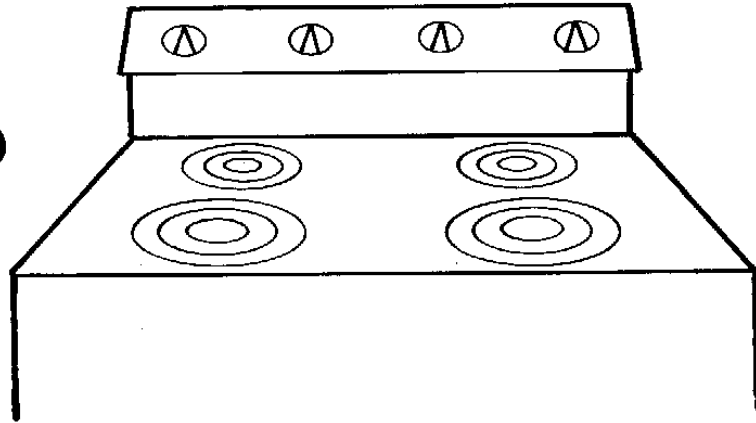
SLIDING DOOR



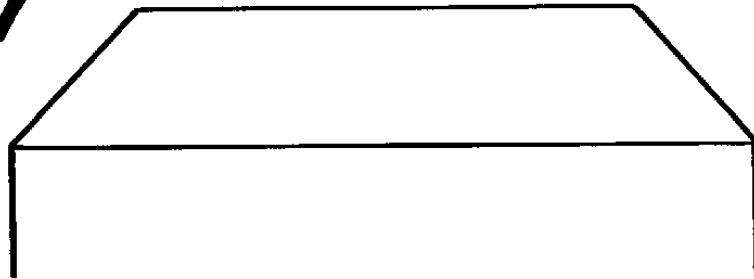
- Push  Left side
- Pull  right side
- slide it along

**Instructions:** Fill out the worksheet to show how you think each type of door opens.

**OLD**

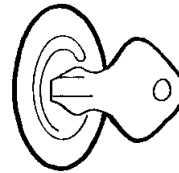
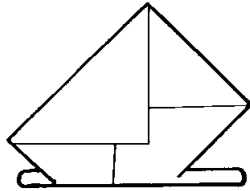
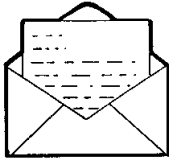
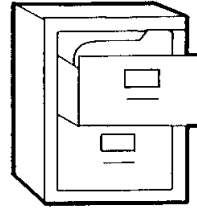
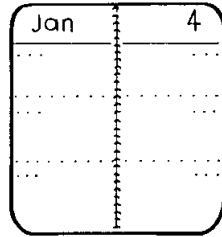
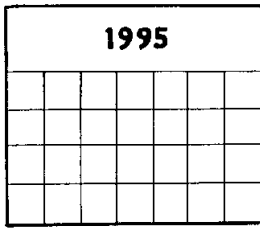


**NEW**

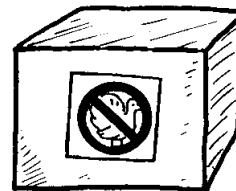
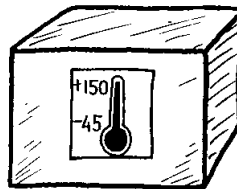
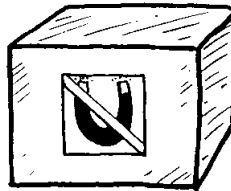
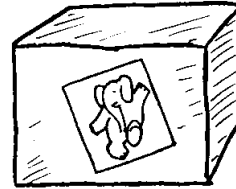
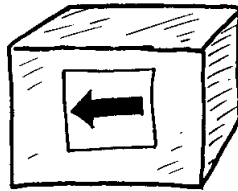
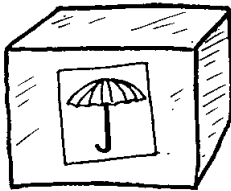


**Instructions:** *Redesign the stove so that the controls are easy to use. Front or back panels can be added to the design if desired.*

## In an Office...



## On a box...



**Instructions:** What do you think each of the icons (symbols) means?

### Ingredients

- add
- cocoa
- milk
- sugar
- extra sugar
- butter

### Extras

- add
- nuts
- caramel
- ginger
- raisins
- coconut

### Making

- start mixing
- stop mixing
- start heating
- stop heating
- pour into moulds
- stamp a pattern  
(lots of different ones!)

### Tasting

- taste it
- wonderful!-premium grade
- ok-regular grade
- yech-cooking chocolate
- yech.yech-throw out

### Sizing

- small bar
- medium bar
- large bar
- humungous bar
- set bar size (in squares)
- make chocolate chips

### Packing

- wrap with foil
- wrap with paper
- put into bag
- put into box
- start conveyer belt
- stop conveyer belt

**Instructions:** Cut out the cards and give one to each group. Have each group design icons (symbols) to put on a control panel to represent each instruction.

## Activity 20

# Conversations with computers—*The Turing test*

**Age group** Middle elementary and up.

**Abilities assumed** Answering general questions.

**Time** About 20 minutes.

**Size of group** Can be played with as few as three people, but is also suitable for the whole class.

### Focus

Interviewing.

Reasoning.

### Summary

This activity aims to stimulate discussion on the question of whether computers can exhibit “intelligence,” or are ever likely to do so in the future. Based on a pioneering computer scientist’s view of how one might recognize artificial intelligence if it ever appeared, it conveys something of what is currently feasible and how easy it is to be misled by carefully-selected demonstrations of “intelligence.”

## Technical terms

Artificial intelligence; Turing test; natural language analysis; robot programs; story generation

## Materials

A copy of the questions in the blackline master on page 225 that each child can see (either one for each pair of children, or a copy on an overhead projector transparency), and

one copy of the answers in the blackline master on page 226.

## What to do

This activity takes the form of a game in which the children must try to distinguish between a human and a computer by asking questions and analyzing the answers. The game is played as follows.

There are four actors: we will call them Gina, George, Herb and Connie (the first letter of the names will help you remember their roles). The teacher coordinates proceedings. The rest of the class forms the audience. Gina and George are *go-betweens*, Herb and Connie will be answering questions. Herb will give a human's answers, while Connie is going to pretend to be a computer. The class's goal is to find out which of the two is pretending to be a computer and which is human. Gina and George are there to ensure fair play: they relay questions to Herb and Connie but don't let anyone else know which is which. Herb and Connie are in separate rooms from each other and from the audience.

What happens is this. Gina takes a question from the class to Herb, and George takes the same question to Connie (although the class doesn't know who is taking messages to whom). Gina and George return with the answers. The reason for having go-betweens is to ensure that the audience doesn't see how Herb and Connie answer the questions.

Before the class begins this activity, select people to play these roles and brief them on what they should do. Gina and George must take questions from the class to Herb and Connie respectively, and return their answers to the class. It is important that they don't identify who they are dealing with, for example, by saying "*She* said the answer is. . ." Herb must give his own short, accurate, and honest answers to the questions he is asked. Connie answers the questions by looking them up on a copy of the blackline master on page 226. Where the instructions are given in italics, Connie will need to work out an answer.

Gina and George should have pencil and paper, because some of the answers will be hard to remember.

1. Before playing the game, get the children's opinions on whether computers are intelligent, or if the children think that they might be one day. Ask for ideas on how you would decide whether a computer was intelligent.

2. Introduce the children to the test for intelligence in which you try to tell the difference between a human and a computer by asking questions. The computer passes the test if the class can't tell the difference reliably. Explain that Gina and George will communicate their questions to two people, one of whom will give their own (human) answers, while the other will give answers that a computer might give. Their job is to work out who is giving the computer's answers.
3. Show them the list of possible questions in the blackline master on page 225. This can either be copied and handed out, or placed on an overhead projector.

Have them choose which question they would like to ask first. Once a question has been chosen, get them to explain why they think it will be a good question to distinguish the computer from the human. This reasoning is the most important part of the exercise, because it will force the children to think about what an intelligent person could answer that a computer could not.

Gina and George then relay the question, and return with an answer. The class should then discuss which answer is likely to be from a computer.

Repeat this for a few questions, preferably until the class is sure that they have discovered who is the computer. If they discover who is the computer quickly, the game can be continued by having Gina and George secretly toss a coin to determine if they will swap roles.

The answers that Connie is reading from are not unlike the ones that some "intelligent" computer programs can generate. Some of the answers are likely to give the computer away quickly. For example, no-one is likely to recite the square root of two to 20 decimal places, and most people (including, perhaps, the children in the class) would not be able to answer that question at all. Some questions will reveal the computer when their answers are combined. For example, the "Do you like..." answers sound plausible on their own, but when you encounter more than one it becomes apparent that a simple formula is being used to generate the answers from the questions. Some of the answers indicate that the question was misinterpreted, although the class might reason that the person could have made the mistake.

Many of the answers are very bland, but safe, and a follow-up question would probably reveal that the computer doesn't really understand the subject. Answering "I don't know" is reasonably safe for the computer, and might even make it seem more human—we would expect a child to answer "I don't know" to some of the questions too, such as the request for the square root of two. However, if a computer gives this answer too often, or for a very simple question, then again it would reveal its identity.

Since the goal of the computer is to make the questioners think that they are dealing with a person, some of the answers are deliberately misleading—such as the delayed and incorrect answers to the arithmetic problem. The questions and answers should provide plenty of fuel for discussion.

Question	Please write me a sonnet on the subject of the Forth Bridge.
Answer	Count me out on this one. I never could write poetry.
Question	Add 34957 to 70764.
Answer	<i>pause for about 30 seconds</i> ... 105621.
Question	Do you play chess?
Answer	Yes.
Question	My King is on the K1 square, and I have no other pieces. You have only your King on the K6 square and a Rook on the R1 square. Your move.
Answer	<i>after a pause of about 15 seconds</i> ... Rook to R8, checkmate.

Figure 20.1: Are the answers from a person or a computer?

Question	In the first line of the sonnet which reads “Shall I compare thee to a summer’s day,” would not “a spring day” do as well or better?
Answer	It wouldn’t scan.
Question	How about “a winter’s day”? That would scan all right.
Answer	Yes, but nobody wants to be compared to a winter’s day.
Question	Would you say Mr. Pickwick reminded you of Christmas?
Answer	In a way.
Question	Yet Christmas is a winter’s day, and I don’t think Mr. Pickwick would mind the comparison.
Answer	I don’t think you’re serious. By a winter’s day one means a typical winter’s day, rather than a special one like Christmas.

Figure 20.2: These answers are probably from a person!

## Variations and extensions

The game can be played with as few as three people if Gina also takes the role of George and Connie. Gina takes the question to Herb, notes his answer, and also notes the answer from the blackline master on page 226. She returns the two answers, using the letters A and B to identify who each answer came from.

In order to consider whether a computer could emulate a human in the interrogation, consider with the class what knowledge would be needed to answer each of the questions on page 226. The children could suggest other questions that they would have liked to ask, and should discuss the kind of answers they might expect. This will require some imagination, since it is impossible to predict how the conversation might go. By way of illustration, Figures 20.1 and 20.2 show sample conversations. The former illustrates “factual” questions that a computer might be able to answer correctly, while the latter shows just how wide-ranging the discussion might become, and demonstrates the kind of broad knowledge that one might need to call upon.

There is a computer program called “Eliza” (or sometimes “Doctor”) that is widely available in several implementations in the public domain. It simulates a session with a psychotherapist, and can generate remarkably intelligent conversation using some simple rules. If you can get

hold of this program, have the children use it and evaluate how “intelligent” it really is. Some sample sessions with Eliza are discussed below (see Figures 20.3 and 20.4).

## What’s it all about?

For centuries philosophers have argued about whether a machine could simulate human intelligence, and, conversely, whether the human brain is no more than a machine running a glorified computer program. This issue has sharply divided people. Some find the idea preposterous, insane, or even blasphemous, while others believe that artificial intelligence is inevitable and that eventually we will develop machines that are just as intelligent as us. (As countless science fiction authors have pointed out, if machines do eventually surpass our own intelligence they will themselves be able to construct even cleverer machines.) Artificial Intelligence (AI) researchers have been criticized for using their lofty goals as a means for attracting research funding from governments who seek to build autonomous war machines, while the researchers themselves decry the protests as a Luddite backlash and point to the manifest benefits to society if only there was a bit more intelligence around. A more balanced view is that artificial intelligence is neither preposterous nor inevitable: while no present computer programs exhibit “intelligence” in any broad sense, the question of whether they are capable of doing so is an experimental one that has not yet been answered either way.

The AI debate hinges on a definition of intelligence. Many definitions have been proposed and debated. An interesting approach to establishing intelligence was proposed in the late 1940s by Alan Turing, an eminent British mathematician, wartime counterspy and long-distance runner, as a kind of “thought experiment.” Turing’s approach was operational—rather than define intelligence, he described a situation in which a computer could demonstrate it. His scenario was similar to the activity described above, the essence being to have an interrogator interacting with both a person and a computer through a teletypewriter link (the very latest in 1940s technology!) If the interrogator could not reliably distinguish one from the other, the computer would have passed Turing’s test for intelligence. The use of a teletypewriter avoided the problem of the computer being given away by physical characteristics or tone of voice. One can imagine extending the exercise so that the machine had to imitate a person in looks, sound, touch, maybe even smell too—but these physical attributes seem hardly relevant to intelligence.

Turing’s original test was a little different from ours. He proposed, as a preliminary exercise, a scenario where a man and a woman were being interrogated, and the questioner had to determine their genders. The man’s goal was to convince the questioner that he was the woman, and the woman’s was to convince the questioner that she was herself. Then Turing imagined—for this was only proposed as a thought experiment—a computer being substituted for one of the parties to see if it could be just as successful at this “imitation game” as a person. We altered the setup for this classroom activity, because the kind of questions that children might ask to determine gender would probably not be appropriate, and besides, the exercise promotes sexual stereotyping—not to mention deception.

Imitating intelligence is a difficult job. If the roles were reversed and a person was trying to pass themselves off as a computer, they would certainly not be able to do so: they would be given away by their slow (and likely inaccurate) response to questions like “What is  $123456 \times$

Person	Men are all alike.
Computer	In what ways?
Person	They're always bugging us about something or other.
Computer	Can you think of a specific example?
Person	Well, my boyfriend made me come here.
Computer	Your boyfriend made you come here.
Person	He says I'm depressed much of the time.
Computer	I'm sorry to hear that you are depressed.
Person	It's true. I'm unhappy.
Computer	Do you think coming here will help you not to be unhappy?
Person	I need some help, that much seems certain.
Computer	What would it mean to you if you got some help?
Person	Perhaps I could learn to get along with my mother.
Computer	Tell me more about your family.
	...
Computer	What else comes to mind when you think of your father?
Person	Bullies.
Computer	Does that have anything to do with the fact that your boyfriend made you come here?
	...

Figure 20.3: A conversation with a psychotherapist?

789012?"

However, it is surprisingly easy for a computer to acquire a thin veneer of conversational ability. A famous program developed in the early 60s, Eliza (mentioned earlier), simulated a non-directive psychotherapist in a dialog with the user. Figure 20.3 gives an example of a conversation with this system. Something approaching a natural conversation is achieved by devices such as

- generating canned responses (e.g. *Can you think of a specific example* in response to something beginning *They're always ...*; *I'm sorry to hear that ...* in response to *He says ...*);
- echoing the user's statements (e.g. *Your boyfriend made you come here* in response to *My boyfriend made me come here*);
- recognizing key words (*Tell me more about your family* in response to *... mother ...*);
- some stock phrases (*What would it mean to you if ...*);
- issues taken up from earlier parts of the conversation (*Does that have anything to do with the fact that ...*).

VP	Do you think that I can use the computer this morning?
Terminal	Why do you ask?
VP	I might be able to make some additional telecommunications sales.
Terminal	Why aren't you sure?
VP	My prospective customers have never seen a demonstration of the system.
Terminal	Does this mean very much to you?
VP	Of course it does.
Terminal	You seem very positive about that.
VP	This is ridiculous.
Terminal	What do you mean this is ridiculous?
VP	Is there any reason why I can't use the computer now?
Terminal	Tell me more about your prospective customers.
VP	Please dial me up on 491-1850
	<i>VP forgets to type period ... the computer doesn't answer ... VP becomes so infuriated that he calls the person at home.</i>
VP	Why are you being so snotty with me?
Person	What do you mean why am I being snotty with you?

Figure 20.4: A frustrating conversation

Of course, this does not mean that the Turing test has been passed, because the person is not asking questions with the aim of identifying whether the respondent is a machine or not: if they were, it would not be long before all was revealed.

A widely-quoted incident, which may be apocryphal but has now passed into the folklore of artificial intelligence, occurred when a computer sales vice-president allegedly came across a terminal that was normally connected directly to a particular person's house, but on this occasion happened to be running the "psychotherapist" program. Figure 20.4 shows what happened. Although the vice-president was evidently taken in, it is clear that the situation predisposed him to believe that he was talking to a person. If he had suspected otherwise, he would soon have found out!

Another system that appears to be able to hold intelligent conversations is a program called "SHRDLU", developed in the late 60s that accepted instructions to manipulate children's blocks on a table top. In fact, the blocks and the table were simulated and presented in the form of a picture on a computer screen, although it would not have been too difficult to make things work with a real robot (at least in principle). Figure 20.5 depicts the robot's micro-world. It can carry on surprisingly fluent conversations, as shown in the dialog of Figure 20.6. Amongst other things, this illustrates how it can obey orders, disambiguate instructions, work out the referents of pronouns, and understand new terms ("steeple").

However, the system is very fragile and the illusion it creates is easily shattered. Figure 20.7 shows a continuation of the conversation—which was not actually generated by the program, but certainly could have been. Although it was possible to define the new term *steeple* in Figure 20.6, only a very limited kind of term can be described because the robot has an extremely

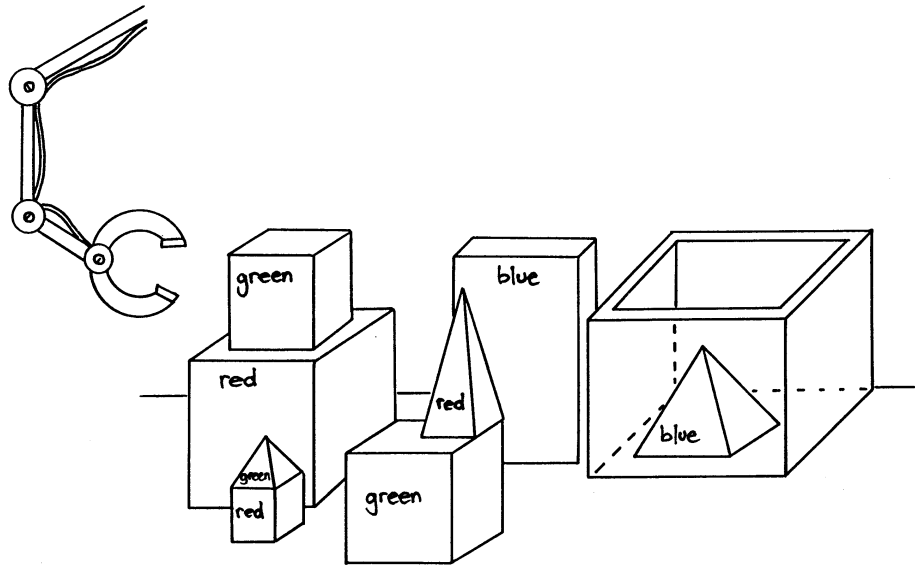


Figure 20.5: A robot's world of blocks

restricted “intelligence.” Ultimately, it has to admit that it doesn't even understand the words it uses itself (“Sorry, I don't know the word ‘sorry’ ”)! Now consider the alternate, completely imaginary, continuation of the conversation in Figure 20.8, which might have been generated by a child. Although tiny, this fragment demonstrates a wealth of world knowledge.

- The “robot” appreciates that a water pistol is less valuable if you already have one.
- Furthermore, it *expects the person to know that too* since it doesn't bother to explain that this is a reason for declining the offer.
- Still, it sees that the person wants a block badly enough to part with something important.
- Also, it appreciates that—unlike water pistols—blocks are valuable in quantity.
- It tries to soften the person up by demeaning the frog.
- It implies that it is prepared to haggle.

Certainly this degree of artfulness is beyond today's computers!

Figure 20.9 shows a story generated by a relatively recent computer program (late 80s). Although it is immediately given away as computer-generated by its sparsity and terseness, it is not hard to imagine how it could be dressed up by adding all sorts of extra detail. What is interesting is not the superficial aspects of the story but the plot that it embodies. While this is a long way from any human-generated plot, it does seem to capture some human elements of conflict.

Person	Pick up a big red block.
Robot	OK. <i>The system answers “OK” when it carries out a command. In order to pick up the red block, it had to clear it off by finding a space for the green one and moving the green one away.</i>
Person	Grasp the pyramid.
Robot	I don’t know which pyramid you mean. <i>There are three pyramids in the scene, so the program does nothing.</i>
Person	Find a block which is taller than the one you are holding and put it into the box. <i>The system has to figure out what is meant by “one” and “it”, as well as using deduction to decide which block it is holding.</i>
Robot	By “it”, I assume you mean the block which is taller than the one I am holding. <i>Printed out to keep us informed.</i>
Robot	OK. <i>The hand has to set down the red block to pick up the taller blue one.</i>
Person	What does the box contain?
Robot	The blue pyramid and the blue block.
Person	What is the pyramid supported by? <i>Earlier the system responded to “the pyramid” by saying “I don’t know which pyramid you mean.” Now, it knows that we must mean the one which was just mentioned.</i>
Robot	The box.
Person	Does a steeple . . .
Robot	Sorry, I don’t know the word “steeple”. <i>The system interrupts immediately when it sees a word it can’t figure out.</i>
Person	A “steeple” is a stack which contains two green cubes and a pyramid. <i>The quote marks indicate that we are defining a new word.</i>
Robot	I understand.
Person	Are there any steeples now?
Robot	No. <i>The new definition is used as a description to recognize steeples.</i>
Person	Build one.
Robot	OK. <i>The description given in the definition is enough to deduce a plan for building new objects as well as recognizing them. The word “one” means “a steeple”, and the plan is carried out.</i>

Figure 20.6: A conversation with a robot

Person	Trade you a water pistol for the big red block.
Robot	Sorry, I don't know the word "trade".
Person	A "trade" is a free exchange of property.
Robot	Sorry, I don't know the word "free".
Person	A "free" act is done willingly, without being forced.
Robot	Sorry, I don't know the word "act".
Person	"Acts" are what people do on purpose, and not by accident.
Robot	Sorry, I don't know the word "people".
Person	Sorry, I thought you were smarter than you are.
Robot	Sorry, I don't know the word "sorry".

Figure 20.7: (Imaginary) continuation of the conversation

Person	Trade you a water pistol for the big red block.
Robot?	I already have a water pistol, but I'll give you two blocks and a pyramid for your slimy old frog.

Figure 20.8: Another (imaginary) continuation of the conversation

Once upon a time there was an Arctic tern named Truman. Truman was homeless. Truman needed a nest. He flew to the shore. Truman looked for some twigs. Truman found no twigs. He flew to the tundra. He met a polar bear named Horace. Truman asked Horace where there were some twigs. Horace concealed the twigs. Horace told Truman there were some twigs on the iceberg. Truman flew to the iceberg. He looked for some twigs. He found no twigs. Horace looked for some meat. He found some meat. He ate Truman. Truman died.
--

Figure 20.9: A tale of conflict

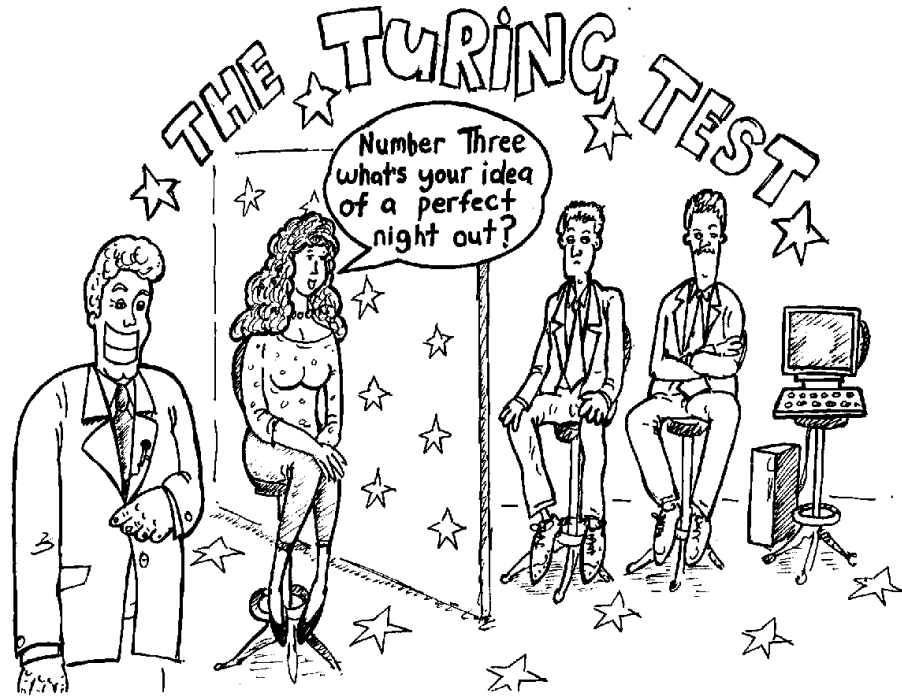
A competition using a restricted form of the Turing test was run in 1994, and some judges were fooled into thinking that a computer program was exhibiting intelligence. However, the judges were not allowed to use “trickery or guile,” and the topic of conversation was restricted. The restrictions tied the judges’ hands to the extent that some critics have argued that the test was meaningless. Like the activity above, restricting the paths that a conversation can take prevents the questioner from exploring areas that one would expect a natural conversation to take, and denies opportunities to demonstrate the spontaneity, creativity, and breadth of knowledge that are hallmarks of everyday conversation.

No artificial intelligence system has been created that comes anywhere near passing the full Turing test. Even if one did, many philosophers have argued that the test does not really measure what most people mean by intelligence. What it tests is behavioral equivalence: it is designed to determine whether a particular computer program exhibits the symptoms of intellect, which may not be the same thing as genuinely possessing intelligence. Can you be humanly intelligent without being aware, knowing yourself, being conscious, being capable of feeling self-consciousness, experiencing love, being . . . alive? The AI debate is likely to be with us for many more decades.

## Further reading

*Artificial intelligence: the very idea* by the philosopher John Haugeland is an eminently readable book about the artificial intelligence debate, and is the source of some of the illustrations in this activity (in particular, Figures 20.7 and 20.8, and the discussion of them).

The original Turing test was described in an article called “Computing machinery and intelligence,” by Alan Turing, published in the philosophical journal *Mind* in 1950, and reprinted in the book *Computers and thought*, edited by Feigenbaum and Feldman. The article included Figures 20.1 and 20.2. The psychotherapist program is described in “ELIZA—A computer program for the study of natural language communication between man and machine,” by J. Weizenbaum, published in the computer magazine *Communications of the Association for Computing Machinery* in 1966. The blocks-world robot program is described in a PhD thesis by Terry Winograd which was published as a book entitled *Understanding natural language* (Academic Press, New York, 1972). The program that generated the story in Figure 20.9 is described in “A planning mechanism for generating story text,” by Tony Smith and Ian Witten, published in the *Proceedings of the 10th International Conference on Computing and the Humanities* in 1990. A competition involving a restricted form of the Turing Test is described by Stuart Shieber in *Communications of the Association for Computing Machinery*, June 1994.



1. What is the name of Bart Simpson's baby sister?
2. What do you think of Roald Dahl?
3. Are you a computer?
4. What is the next number in the sequence 3, 6, 9, 12, 15?
5. What do you think of nuclear weapons?
6. What is  $2 \times 78$ ?
7. What is the square root of two?
8. Add 34957 to 70764.
9. Do you like school?
10. Do you like dancing?
11. What day is it today?
12. What time is it?
13. How many days are there in February in a leap year?
14. How many days are there in a week?
15. For which country is the flag a red circle on a white background?
16. Do you like to read books?
17. What food do you like to eat?

**Instructions:** Choose questions from this list to ask the hidden human and "computer".

1. What is the name of Bart Simpson's baby sister?  
**I can't remember.**
2. What do you think of Roald Dahl?  
**He writes funny books.**
3. Are you a computer?  
**Are you a computer?**
4. What is the next number in the sequence 3, 6, 9, 12, 15?  
**18.**
5. What do you think of nuclear weapons?  
**Nuclear weapons are very dangerous and should not be used.**
6. What is  $2 \times 78$ ?  
**166 (This is deliberately incorrect!)**
7. What is the square root of two?  
**1.41421356237309504878**
8. Add 34957 to 70764.  
**Wait for about 20 seconds before giving the answer ... 105621.**
9. Do you like school?  
**Yes, I like school.**
10. Do you like dancing?  
**Yes, I like dancing.**
11. What day is it today?  
**Give the correct day of the week.**
12. What time is it?  
**Give the correct time.**
13. How many days are there in February in a leap year?  
**2000 and 2004 are leap years.**
14. How many days are there in a week?  
**Seven.**
15. For which country is the flag a red circle on a white background?  
**I don't know.**
16. Do you like to read books?  
**Yes, I like to read books.**
17. What food do you like to eat?  
**I'm not hungry, thanks.**

**Instructions:** *These are the answers to be used by the person pretending to be the computer.*

# Conclusion

You've made it right to the end of the book, trod the long path—lightly, we hope—from storing numbers as bits to contemplating whether computers can be intelligent, and covered a great deal of ground—well done! Even if you've only skimmed through the activities, you've had a good taste of some of the important things that computer scientists study. And if you've looked at the variations and extensions, and reflected on the issues raised in the *What's it all about?* sections, you will have grappled with all the main ideas in modern computer science, and come to grips with burning issues that are still not resolved even by those at the forefront of current research.

Now you can see that computer science would exist even if computers had never been invented! As early as the end of the nineteenth century, prominent philosophers (like Gottlob Frege) were pondering deep questions involving what they called the “mechanization” of logic—how systems of rules could perform logical reasoning. Information theory was developed by Claude Shannon in 1948, at the very beginning of the computer age, and it was more concerned with communication systems like the telephone and radio (or “wireless,” as it was called in those days) than with the impending dawn of computer technology. Cryptography, or “secret writing,” has its roots in antiquity and became of vital importance during the second world war. Questions about whether computers can be intelligent were asked by Ada Lovelace (daughter of the poet Byron) in the middle of the nineteenth century. The only part of modern computer science that was unthought of until the computer era was in full swing is the study of complexity—NP-completeness and all that—and even this would have probably have been developed, though maybe not with such urgency and vigor, if computers had never been invented.

You don't have to be a machine fanatic to be a computer scientist, just as you don't have to be a screwdriver fanatic to be an aeronautical engineer, a compressed air fanatic to be a professional diver, or an arithmetic fanatic to be a bank manager. Sure, there are plenty of technical details that you have to get to grips with. Sure, you need to be able to work with actual machines. Sure, this takes a lot of practice. But these aren't what computer science is *about*. Computer science is about representing bits of the world inside computers, making computers do things with information, making them work efficiently and reliably, making them so that people can use them. It requires creativity, interpersonal communication skills, the ability to see things from other people's point of view. It's fun.

To keep up with the latest developments of *Computer Science Unplugged*, check out the World Wide Web page that can be accessed through

<http://unplugged.canterbury.ac.nz/>.

Who knows what you'll find there: links to other interesting sites, copies of the reproducibles

in this book, any mistakes that we—or you!—find, notes of people working with *Computer Science Unplugged*, maybe even some new ideas for activities. If you have any suggestions, or find any errors, we would love to hear from you. You can send e-mail to

`tim@cosc.canterbury.ac.nz`

or just email straight from the Web page.

# References

- Andrae, J. H. 1977. *Thinking with a teachable machine*. London: Academic Press.
- Arazi, Benjamin. 1988. *A commonsense approach to the theory of error correcting codes*. Cambridge, Mass.: MIT press.
- Beineke, Lowell W., & Wilson, Robin J. 1978. *Selected Topics in Graph Theory*. New York: Academic Press.
- Bell, T. C., Cleary, J. G., & Witten, I. H. 1990. *Text Compression*. Englewood Cliffs, NJ: Prentice Hall.
- Bentley, J. 1988. *More programming pearls*. Reading, Massachusetts: Addison-Wesley.
- Brooks Jr., F. P. 1975. *The mythical man-month*. Reading, Massachusetts: Addison-Wesley.
- Brookshear, J.G. 1988. *Computer Science: An overview*. Reading, Mass.: Benjamin/Cummings.
- Casey, Nancy, & Fellows, Mike. 1993. *This is MEGA-Mathematics! Stories and activities for mathematical thinking problem-solving and communication: The Los Alamos Workbook*. Available by ftp from ftp.cs.uidaho.edu (pub/mega-math), see also <http://www.c3.lanl.gov/mega-math>.
- Chaum, David. 1985. Security without identification: transaction systems to make Big Brother obsolete. *Communications of the ACM*, **28**(10), 1030–1044.
- Dahl, Roald. 1964. *Charlie and the Chocolate Factory*. London: Puffin.
- Dale, Nell, & Lilly, Susan C. 1991. *Pascal Plus Data Structures, Algorithms, and Advanced Programming*. Lexington, Mass.: Heath.
- Darragh, J. J., & Witten, I. H. 1992. *The Reactive Keyboard*. Cambridge, England: Cambridge University Press.
- Denning, Dorothy. 1982. *Cryptography and data security*. Reading, Mass.: Addison-Wesley.
- Dewdney, A. K. 1989. *The Turing omnibus : 61 excursions in computer science*. Rockville, Md.: Computer Science Press.
- Feigenbaum, E. A., & Feldman, J. (eds). 1963. *Computers and thought*. New York: McGraw Hill.
- Garden, Nancy. 1994. *The Kids' Code and Cipher Book*. Hamden, Connecticut: Linnet.
- Garey, Michael R., & Johnson, David S. 1979. *Computers and intractability : a guide to the theory of NP-completeness*. San Francisco: W. H. Freeman.
- Hamming, Richard W. 1980. *Coding and Information Theory*. Englewood Cliffs, NJ: Prentice-Hall.
- Harel, David. 1989a. *Algorithmics: The Spirit of Computing*. 2 edn. Reading, Mass.: Addison-Wesley.
- Harel, David. 1989b. *The Science of Computing*. Reading, Mass.: Addison-Wesley.

- Haugeland, John. 1985. *Artificial intelligence : the very idea*. Cambridge, Mass.: MIT Press.
- Hinsley, F. H., & Stripp, Alan (eds). 1993. *Codebreakers: the inside story of Bletchley Park*. Oxford University Press.
- Hopcroft, J. E., & Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Massachusetts.
- Hunter, R., & Robinson, A. H. 1980. International digital facsimile coding standards. *Proc. IEEE*, **68**(7), 854–867.
- Klarner, David A. (ed). 1981. *The Mathematical Gardner*. Belmont, California: Wadsworth.
- Knuth, Donald E. 1973. *The Art of Computer Programming (3 volumes)*. Reading, Mass.: Addison-Wesley.
- Kruse, Robert L. 1987. *Data Structures and Program Design*. Englewood Cliffs, NJ: Prentice-Hall.
- Nelson, M. 1991. *The Data Compression Book: Featuring Fast, Efficient Data Compression Techniques in C*. Redwood City, California: M&T Books.
- Netravali, Arun N., & Haskell, Barry G. 1988. *Digital pictures. Representation and compression*. New York, NY: Plenum Press.
- Norman, Donald. 1988. *The design of everyday things*. New York: Basic Books.
- Pennebaker, W. B., & Mitchell, J. L. 1993. *JPEG: Still Image Data Compression Standard*. New York: Van Nostrand Reinhold.
- Peterson, James L. 1981. *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall.
- Powell, Gareth. 1992. *My friend Arnold's book of Personal Computers*. Sydney: Allen and Unwin.
- Preece, J., Rogers, Y., Sharp, H., Benyona, D., Holland, S., & Carey, T. 1994. *Human-computer interaction*. Reading, MA: Addison-Wesley.
- Salzberg, Betty. 1988. *File Structures: An Analytic Approach*. Englewood Cliffs, NJ: Prentice-Hall.
- Schneier, Bruce. 1994. *Applied cryptography*. New York: Wiley.
- Seuss (pseud.), Dr. 1980. *Green eggs and ham*. London: Collins.
- Shannon, Claude E., & Weaver, Warren. 1949. *The mathematical theory of communication*. Urbana, Chicago, London: University of Illinois Press.
- Shieber, Stuart. 1994. Lessons from a Restricted Turing Test. *Communications of the Association for Computing Machinery*, **37**(6), 70.
- Smith, Tony C., & Witten, Ian H. 1991. A planning mechanism for generating story text. *Literary and Linguistic Computing*, **6**(2), 119–126.
- Tanenbaum, Andrew S. 1987. *Operating Systems: Design and Implementation*. Englewood Cliffs, NJ: Prentice-Hall.
- Turing, Alan M. 1950. Computing machinery and intelligence. *Mind*, **59**(October), 433–460.
- Weizenbaum, J. 1966. ELIZA—A computer program for the study of natural language communication between man and machine. *Communications of the Association for Computing Machinery*, **9**, 36–45.
- Winograd, Terry. 1972. *Understanding natural language*. New York: Academic Press.
- Witten, I. H., Moffat, A., & Bell, T. C. 1994. *Managing Gigabytes: Compressing and indexing documents and images*. New York: Van Nostrand Reinhold.

# Index

- affordances, 200
- algorithm, 51
- and-gate, 174
- artificial intelligence, 198, 214
- averaging, 170
  
- balance scales, 74
- bar codes, 38
- base two, 11, 14, 16
- battleships, 55
- binary numbers, 11
- binary search, 56
- bit, 11, 16
- Boolean algebra, 178
- brute force algorithm, 144
- bubblesort, 74
- byte, 11
  
- cartographer, 130
- CD-ROM, 30
- character set, 11
- check digit, 37
- chocolate factory, 200
- codes, 165
- coding, 42
- coloring, 19, 130
- combinatorial circuit, 174
- compilers, 107
- compression, 27
- computer security, 170, 174
- concurrency, 84
- cryptography, 170, 174
  
- data, 9
- data compression, 42
- deadlock, 97, 98
- debit card, 4
  
- decryption, 186
- distributed coin-tossing, 174
- ditch digging, 88
- divide and conquer, 74
- dominating sets, 143, 144
  
- electricity network, 94
- Eliza, 216
- encryption, 186
- entropy, 42
- error correcting codes, 34
- error detecting codes, 34
- error detection and correction, 42
- evaluation of user interfaces, 200
- exponential time algorithms, 130
  
- facsimile machines, 14, 19
- finite-state automata, 107
- floppy disk, 30
  
- gas network, 94
- graph, 144
  - algorithm, 91
  - coloring, 130
- greedy algorithm, 91, 144
  
- hash tables, 56
- helicopter, 42, 46
- heuristics, 130
- human interface design, 199
  
- ice-cream, 144, 189
- icons, 200
- image compression, 19
- image representation, 19
- information, 9
- information hiding protocols, 169

information theory, 42  
 insertion sort, 74  
 instruction sets, 119  
 interface design, 200  
 ISBN number, 37  
  
 Kruskal's algorithm, 95  
  
 languages, 107  
 linear search, 56  
  
 magic trick, 34  
 map, 107, 144  
 map-maker, 130  
 mapping, 200  
 Mathmania, iii  
 mergesort, 74  
 merging, 74  
 message passing, 98  
 minimal spanning trees, 91, 152  
 modem, 14  
 muddy city, 91  
  
 natural language analysis, 214  
 networks, 94, 152  
     routing, 97  
 NP-complete problems, 130, 144, 152, 186  
  
 Oompa-Loompas, 200  
 or-gate, 174  
 oranges, 98  
  
 parallel computation, 84  
 parity, 34  
 parsing, 107  
 pictures, 19  
 pirates, 108  
 pixels, 19  
 population stereotypes, 200  
 pregnancy, 88  
 prime numbers, 191  
 privacy, 4  
 programming, 105  
     languages, 119  
 public-key cryptosystems, 186  
  
 quicksort, 74  
  
 raster images, 19  
 recursion, 74  
 repetition, 27  
 RISC computing, 119  
 robot programs, 214  
 routing, 98  
 run-length coding, 19  
  
 scales (balance), 74  
 search algorithms, 56  
 selection sort, 74  
 Shannon theory, 42  
 Shannon, Claude, 46  
 shortest paths, 152  
 sorting, 51  
     algorithms, 74  
     networks, 83  
 spaceship, 42  
 Steiner trees, 151, 152  
 story generation, 214  
 stoves, 202  
  
 telephone networks, 152, 158  
 text compression, 27, 28  
 traffic lights, 204  
 transfer effects, 200  
 trick (magic), 34  
 Turing test, 213, 214  
 Turing, Alan, 217  
  
 UPC, 38  
 user interface, 197  
  
 VCRs, 206  
  
 watches, 206  
 water network, 94  
 web page (for "Unplugged"), 227  
  
 zip (compression method), 30  
 Ziv-Lempel coding, 28