THE LIVING WORLD
PLANTS MODULE
The Living World – Plants Module contains the following:

- Teacher's Resource
- The Living World – Plants (student projects)
- The Living World – Plants Activities
Teacher’s Resource
Table of Contents

Introduction .................................................. 2
Exploring the World of Plants .............................. 3
The Living World-Plants .................................. 4
10 Activity Sheets .......................................... 4
Teacher’s Resource ......................................... 5
Online Help / User’s Guide ................................. 6
Activity Sheet Information ................................. 7

Plant Projects ............................................... 8
Chapter 1 Flower Facts .................................... 8
Chapter 2 Flower Helpers ................................. 16
Chapter 3 Plant Wiz Quiz ................................. 26
Chapter 4 Green Games - Lost on Earth ............. 33

Where to Go From Here .................................. 43

References .................................................... 44
**INTRODUCTION**

This is the second module designed to help students learn about MicroWorlds within a science context. The first module introduced MicroWorlds and its multimedia features. By the end of the first module, students were able to create a multimedia presentation that included text, sound, video and graphics, plus simple animation. In this module, students go one step farther. They begin to create more complex projects in which the action and pacing are under partial program control. Animations are somewhat more complex with multiple actions occurring simultaneously. Students are introduced to the power of parallel processing, although they may not realize they are making use of this powerful programming capability. What they will be aware of is that they are developing and running animations and simple simulations in a natural, logical way.

With these projects, students see that knowing how to use a tool such as MicroWorlds makes it possible to create more than just multimedia reports. They are able to develop different types of presentations and activities, including games. Games are a very powerful motivator for both children and teenagers. When developing their own games, students not only need to know the scientific information relevant to the game, but they need to go through all the critical thinking and creative analysis that are key steps in developing an interesting game. And, as anyone who has ever created something knows, it is as much fun (if not more!) creating your own game as it is playing someone else’s.
EXPLORING THE WORLDS OF PLANTS

One of children’s very earliest school experiences is planting seeds and observing how plants grow. Botany, one of the two main divisions of biology, remains an important part of the curriculum throughout the school years. It is a rich topic, full of connections to many other subject areas, from literature to mathematics to art.

The objectives of the Plant Module are:

- To introduce students to the various parts of a flower and learn about their roles.
- To help students learn about some of the birds and insects that are instrumental in cross-pollination.
- To give students an opportunity to do independent research, gathering information about the types of plants that grow in different biomes in order to create a game.

In terms of MicroWorlds, students will do the following:

- Learn how to write and use procedures.
- Become familiar with the difference between commands and reporters.
- Learn how to start, stop, and synchronize parallel processes (multiple processes running simultaneously).
- Learn how to program colors to present textual information.
- Work with conditional statements that make use of information typed in as a procedure is running.

The Plant Module consists of a student’s project book, activity sheets, and a teacher’s guide. It is highly recommended that you review the project book and activity sheets before introducing the concepts to your students. The book is designed so that students can work independently.

This material has been written to work with both Windows and Macintosh computers. When computer
specific key functions are referred to in the project book or activity sheets, they are provided in the following order: keys for computers using Windows; keys for Macintosh computers.

THE LIVING WORLD - PLANTS

The student project book contains four projects: Flower Facts, Flower Helpers, the Plant Wiz Quiz, and Green Games - Lost on Earth. Prior to beginning each project, students will need to do some research and, in the case of Green Games - Lost on Earth, some preplanning of their games.

The first two projects, Flower Facts and Flower Helpers, elaborate on the idea of creating a multimedia presentation about flowers, using somewhat more complex animations and techniques than were used in the Weather Module. In the first chapter, students complete an interactive diagram showing the parts of a flower. In the second project, they create an animation illustrating how a bird and insects assist in cross-pollination. In the third and fourth projects, students learn a set of techniques and ideas that let them design both a simple quiz and a more complex game.

New ideas in each project are carefully explained. Students are referred to various topics in the Online Help (Windows 95) or User’s Guide (Macintosh) in case they need to review information about techniques previously introduced in the Weather Module.

10 ACTIVITY SHEETS

Some of the activity sheets are designed to be worked on once students have completed the first two projects and others only after they’ve completed the third and fourth projects. By dividing the sheets in this way, students have an opportunity to practice the techniques learned at each stage and to learn new techniques that are at about the same level of programming difficulty before moving on to the next level. There is a range of subjects in the activities,
including some that explore ideas relating to the cycle of nature (food chains, bees) and others that link to different subject areas to integrate science with other areas of the curriculum, such as language arts and mathematics.

**TEACHER’S RESOURCE**

The following section provides detailed information about each chapter in the Plants projects. The information about each chapter is divided into the following areas:

- **Content Objectives** relates to the aspect of plants pertinent to this chapter.

- **MicroWorlds Learning** is a summary of the techniques/concepts introduced in the chapter.

- **Classroom Discussion** includes topics that can benefit from full-group discussion. These topics relate to aspects of plant and animal life. The discussion’s objective is to integrate the more theoretical learning with what the students will be doing with MicroWorlds.

- **Off-Computer Activity** may be an experiment or investigation integrating the MicroWorlds project with a real-world activity.

- **Working Through the Chapter** is a section with extracts from each chapter, pinpointing problem areas and solutions. More technical explanations are included to help you solve problems the students may encounter. It may also indicate any research and preplanning necessary prior to beginning a project.
ONLINE HELP / USER’S GUIDE

Just a reminder that the MicroWorlds program comes with Online Help (Windows) and User's Guide (Macintosh) that include comprehensive information on MicroWorlds techniques and programming.

Windows
Choose the Help menu item. You will see MicroWorlds Help Topics Vocabulary and Last Message.

- **MicroWorlds Help Topics** range from introductory techniques to Logo programming concepts, everything you need to get the most out of MicroWorlds.

- **Vocabulary** comprises all the primitives in the MicroWorlds Logo language with definitions and examples.

- **Last Message** provides information about error messages that are printed as you are working with the MicroWorlds program.

The online system is designed so that you can be working on a MicroWorlds project and refer to a section in the Help menu at the same time. The projects and activity sheets have pointers to the topics in the Online Help Topics to find information. If you wish, you may print out pages of information to read away from the computer.

Macintosh
Choose the Help menu item. You will see Vocabulary, Last Message, and a ? The ? changes the mouse pointer to a question mark. Click the ? on tool icons, objects on the page (turtles, buttons, sliders, text boxes), shapes, and colors to get Help information on those icons or objects. Choose ? again to exit Help.

The projects and activity sheets have pointers to the topics in the User's Guide to find information.
**ACTIVITY SHEET INFORMATION**

The following chart indicates the correspondence of cards to projects. All the activities may take more than one session to complete.

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>CROSS-CURRICULAR LINKS</th>
<th>MICROWORLDS LEARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FLOWER FACTS AND FLOWER HELPERS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bee Dance</td>
<td>Make an animation of the dances bees do to communicate.</td>
<td>language arts mathematics</td>
<td>right left</td>
</tr>
<tr>
<td>Seed Scatterer</td>
<td>Show how the wind randomly scatters seeds.</td>
<td>mathematics</td>
<td>random</td>
</tr>
<tr>
<td>A Tree Tale</td>
<td>Describe the events in a tree’s life based on its trunk rings.</td>
<td>language arts</td>
<td>procedures multi-media story</td>
</tr>
<tr>
<td>Metamorphosis</td>
<td>Use a language arts approach to show that some animals change form.</td>
<td>language arts</td>
<td>setx sety</td>
</tr>
<tr>
<td>Food Factory</td>
<td>Create an animation to show the processes that aid in photosynthesis.</td>
<td>parallel processing</td>
<td></td>
</tr>
<tr>
<td>Hot Words</td>
<td>Create shapes out of words to animate and annotate text.</td>
<td>language arts</td>
<td>creating new shapes</td>
</tr>
<tr>
<td><strong>THE PLANT WIZ QUIZ AND GREEN GAMES</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>What’s in a name?</td>
<td>Show the source of plant names through animation.</td>
<td>language arts art</td>
<td>procedures using shapes</td>
</tr>
<tr>
<td>What’s for Dinner?</td>
<td>Create an animation of a plant that digests insects.</td>
<td>language arts</td>
<td>waituntil touching? conditionals</td>
</tr>
<tr>
<td>Intergalactic Food Chain</td>
<td>Animate an imaginary food chain based on the idea of an earth food chain.</td>
<td>language arts</td>
<td>waituntil touching? conditionals</td>
</tr>
<tr>
<td>Quiz Show</td>
<td>Make a more complex quiz that can use words (not just numbers) as answers</td>
<td>mathematics</td>
<td>ifelse</td>
</tr>
</tbody>
</table>

Bold text indicates commands and reporters not previously introduced in the projects.
Plant Projects

CHAPTER 1

FLOWER FACTS

Content Objectives:
- Become familiar with the parts of a flower and the role each part plays in plant reproduction.

MicroWorlds Learning:
- Programming colors
- Writing procedures
- Controlling text presentation with commands
- Addressing or talking to text boxes

CLASSROOM DISCUSSION

Discuss why plants have flowers. Do they have a role to play in the plant’s life? Many students may not realize how many different parts there are to a flower. Ask them to identify what the different parts of a flower are. Explain that flowers contain a plant’s reproductive parts. The stamen is the male reproductive organ and the pistil is the female reproductive organ. Egg cells are produced in the ovary, which is at the base of the pistil. Pollen grains, containing sperm cells, are produced at the tip of the stamen which is called the anther.

You may want to include some discussion about the various other parts of a flowering plant, including the stem, leaves, and roots. Each part serves an important function in the growth and maintenance of the plant. For example, the leaves play the principal role in photosynthesis, the process in which the plant uses energy from the sun to convert carbon dioxide into food for the plant. In so doing, green plants play a vital role in converting energy from the sun into energy that can be used by both plants and animals.
Not all plants have flowers. Ask students if they know other ways plants reproduce. Have they ever grown a plant from a leaf or stem cutting? If someone has, ask him or her to explain what they did and what changes the plant underwent.

**OFF-COMPUTER ACTIVITY**

If possible, and depending on the season, bring in some of the following flowers: chrysanthemums, daisies, dandelions, poinsettias, and flowers from willow or poplar trees. Have students work in small groups. Have each group examine one of these flowers carefully and write a description of the flower. Their descriptions should include the color, scent, and any unusual markings on the flower. If possible, try looking at these flowers under an ultraviolet light to see if other markings appear.

Flowers such as chrysanthemums, daisies, and dandelions actually grow in clusters. They are called inflorescence. The individual flowers in an inflorescence are often mistaken for petals. For example, a dandelion may have 100 or more tiny flowers.

Flowers from poplar and willow trees are called catkins. These inflorescence have neither petals nor sepals. Flowers such as these are called imperfect or incomplete flowers. A flower with all four parts is called a perfect flower.

Some flowers have bracts just beneath the flower. In some plants these can be so large and bright that they also look like part of the flower. The large red bracts of a poinsettia are an example of these. The flowers of a poinsettia are actually very small.

Visit a local greenhouse or botanical garden so students can see different types of flowers. Work with the class to prepare a list of questions to ask on the trip. Students should look for complete and incomplete flowers and any inflorescence.
WORKING THROUGH THE CHAPTER

When students open this project they see an unlabelled diagram of a flower. Although students could complete it by just adding labels, in this project they are shown a more interesting (and elegant) way to add information. They learn how to make whole boxes of text appear and disappear with the click of a mouse button. As they complete the project, students write their first project procedure. Although the procedure is simple, the idea behind it is very powerful. Once students learn how to write and use procedures, they can create a much wider range of projects.

Before Starting on the Computer

Students should do some research about the following parts of a flower: the calyx, the corolla, the stamen, and the pistil. In addition, let them find out about the receptacle, which is the part of the stem where the flower begins to grow. They can write a short paragraph describing the role each part plays in reproduction. During their research, the students may discover that these parts are each made up of other smaller structures. These then can be described.

Make sure all the text is showing. You may want to change the style, size or color of all or part of your text.

If the text box is not large enough, all the text won’t show when the text appears. Since the text box will be transparent, all that will appear is an incomplete description. The text box cannot be enlarged once it is transparent. Remind students of how to enlarge a text box:

- Select it by dragging around it with the regular pointer.
- When the handles appear, drag a handle to set the size.
Text boxes as well as turtles are objects in MicroWorlds. Since there can be many text boxes on a page, it’s necessary to have a way to direct instructions to the appropriate text box. In MicroWorlds this is done by naming the text box and then “talking to” that box, just as in speaking you could address the person to whom you’re talking by name. In this case, the name of the text box is Corolla. The comma after the name is a shortcut for a MicroWorlds command that works when attached to the end of an object name. It stands for “talk to.” (There is also a talkto command.)

If there is only one text box on the page, the “talk to” command does not need to be used. Any commands that affect text boxes will apply to that single box. So, if students don’t type corolla, the first commands introduced, hidetext and showtext, will address the box named Corolla anyway. But, once there are more text boxes, it is easy to unwittingly change, hide, or even erase the contents of a text box by accident. Since there will be many text boxes used in this project, students are shown this technique at the very start.

All appropriate commands (those that affect text boxes) will continue to affect the text box that was addressed until a new “talk to” command is given.

It’s always a good idea for objects to have meaningful names—ones that are easy to remember and that in some way make it clear which object is being addressed. Examples of names that aren’t meaningful include flowerpart1, flowerpart2, etc.
The input to `wait` is in tenths of a second. Remind students that if information disappears too quickly, they can make use a bigger number as input to `wait`. Encourage them to experiment until they are satisfied with the timing.

There are two different greens in the drawing tools. If students are not sure which green to program, they can do the following:

For Macintosh computers only:
- Choose the ? from the Help menu.
- Click on the color on the diagram to see the color number.
- Click on each green to see if it’s the green used for the stamen (for example, if the color under the turtle is 103, then the green used for the stamen is the one containing colors numbered from 100 -109).

In Windows:
- Create a turtle and place this turtle on the stamen in the diagram. Make sure the center of the turtle is over the green.
- In the Command Center, type:
  ```plaintext
  show colorunder
  ```
  A number appears in the Command Center.
Open the Drawing Center. Move the mouse onto each green to see if it's the green used for the stamen (for example, if the color under the turtle is 103, then the green used for the stamen is the one containing colors numbered from 100 to 109).

A dot on a color indicates that it is programmed. When programming colors it's important to remember that a color group is programmed and not a shade of a color. So if a color is programmed, clicking on anything drawn in any of the shades of that color will cause the same instructions to run.

Students may be surprised to see that all hidden text boxes become visible temporarily when they click on the eye tool. Clicking on the eye tool is one way to find a hidden text box and make it visible.

Remind students that the eye tool can be used to show how many objects have been created or if a turtle is programmed. It can also be used to open the dialog box of any object. If students need to change an object, they can select the eye tool and click on that object. If the object is invisible, simply select the eye tool and then click on the object as soon as it's visible. They must then select Visible.
In this project, students are introduced to the power of procedures. It may benefit students to discuss what a procedure is. MicroWorlds has a built-in vocabulary, but you can also add your own words. These words that you add can do exactly what needs to be done in a specific situation. Writing a procedure is like defining a new word. Once it is defined, MicroWorlds “knows” it. Procedures always consist of a group of built-in commands or other procedures in a specific order.

In some ways, procedures act just like built-in commands, but there are key differences:

- A procedure can be edited—if it doesn’t do exactly what you want, you are able to change it.
- Procedures are available only in the project in which they were defined. This is sometimes confusing to students when they begin writing procedures. To use the same procedure in another project, you must either define it again in the Procedures page of the new project or copy it from the first project and paste it into the Procedures page of the new project.

Remind students to take note of MicroWorlds messages. This one says that MicroWorlds does not know the command it’s been given. In order for MicroWorlds to learn it, the students need to define a procedure with that name (see below). Students can get information about messages and some solutions for preventing the situation that caused the message by selecting Last Message in the Help menu.

What happens? A message appears in the Command Center:

I don’t know how to information

You can add new words to MicroWorlds vocabulary by writing procedures. Procedures act just like MicroWorlds built-in commands except you can edit or erase them and they work only in the project in which you made them.
You may want to practice creating real-life procedures. For example, get students to write a procedure called “getreadyforschool.” It could look like this:

```
to getreadyforschool
    wakeup
    getdressed
    eatbreakfast
    catchbus
end
```

How important is the order of the commands? In some cases, commands may be interchanged. In others, they cannot be (you cannot catchbus before you wakeup).
**FLOWER HELPERS**

Content Objectives:
- Get a better understanding of the interdependence of animals and plants by understanding how birds and insects assist plants in cross-pollination.

MicroWorlds Learning:
- Using commands to change the turtle’s direction
- Finding and setting the turtle’s coordinates
- Using subprocedures
- Addressing or “talking to” turtles
- Using commands to run instructions for programmed turtles
- Launching parallel procedures

**CLASSROOM DISCUSSION**

This project is a continuation of the previous project and should be linked in discussion. In this project, students create an animation showing a hummingbird and a butterfly flying around flowers in a garden. The following discussion may help students understand why these animals exhibit this behavior. It will also give them ideas as to what type of flowers to put in their garden background.

Start by asking students to name some of the benefits that animals get from flowers. What benefits do people get from flowers? Flowers are a source of food for many animals. Many animals depend on the nectar and/or pollen they find in flowers. In addition, people use flowers for medicine, dyes, and perfumes, as well as food.

While doing their research for the first project, students read about pollination. Pollination is the transfer of pollen from a stamen to a pistil. Ask students to describe what pollination is and why it is important. It is only after
pollination occurs that a seed is produced through fertilization (a sperm from the pollen joins with the egg cell in the base of the pistil). Without pollination and fertilization, there would be no seeds.

Discuss how animals such as insects and birds help plants in the process of pollination. Explain the difference between self-pollination and cross-pollination. In self-pollination, pollen from the stamen of a flower is transferred to the pistil of the same flower or another flower on the same plant. In cross-pollination, an “agent” carries pollen from the stamen of one flower to the pistil of a flower on another plant.

Ask students, “What type of animals help in cross-pollination?” Try to have them be specific. Instead of answering “insects,” they should suggest what types of insects (butterflies, moths, bees, flies, beetles). More flowers are pollinated by bees than by any other insect. Other animals that help include some birds (in North America, mainly the hummingbird) and bats.

Ask what they think attracts insects and birds to flowers.

**OFF-COMPUTER ACTIVITY**

It’s possible to figure out what type of animal helps a plant during pollination just by observing what the flower looks and smells like. Have students either walk around the neighborhood (weather permitting) or visit a local florist or gardening center. Ask each student to select five of the flowering plants that they see and develop hypotheses about what type of animal they think helps pollinate each of the plants. (To add some variety, look at different seed packets that may show additional flowers.) The students should present the reasons why they’ve come to this conclusion, using information from the chart below.
<table>
<thead>
<tr>
<th>AGENT</th>
<th>UNIQUE CHARACTERISTICS</th>
<th>ATTRACTED TO: COLOR</th>
<th>SCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bee</td>
<td>Cannot see red. Can see ultraviolet light and markings not visible to humans.</td>
<td>yellow, blue</td>
<td>sweet</td>
</tr>
<tr>
<td>Butterfly</td>
<td>Like lots of nectar. Like flowers with nectaries and long tube-shaped corollas because butterfly mouthparts are long and narrow.</td>
<td>yellow, blue</td>
<td>sweet</td>
</tr>
<tr>
<td>Moth</td>
<td>Like lots of nectar. Like flowers with nectaries, and may have long tube-shaped corollas because moth mouthparts are long and narrow. Nocturnal so like flowers that open only at night.</td>
<td>pale or white</td>
<td>strong released at night only</td>
</tr>
<tr>
<td>Birds (mainly, hummingbirds)</td>
<td>Can see red as well as other colors.</td>
<td>red, orange, yellow</td>
<td>no—weak sense of smell. Flowers usually are odorless.</td>
</tr>
<tr>
<td>Beetles and Flies</td>
<td>Like lots of nectar and pollen. Have short mouthparts, so like flat flowers.</td>
<td>white or dull-colored</td>
<td>Spicy or bad smells (e.g. skunk cabbage)</td>
</tr>
<tr>
<td>Bats</td>
<td>In tropics and desert areas. Pollinate bananas, avocados, vanilla. Nectar eating bats help pollinate cacti such as the organ-pipe cactus and the saguaro.</td>
<td></td>
<td>Strong scents</td>
</tr>
</tbody>
</table>

Have students do some research to see if they are right. Suggest that students either look up the information in books, contact a local botanical garden or greenhouse, or search for information online (see References).

Students should use the above information to help design their garden background for the project.

**WORKING THROUGH THE CHAPTER**

What better setting than a garden to see many things happening at once—butterflies fluttering, birds flying, bees buzzing around. In creating this project, students will
expect their on-screen animal helpers to behave in a natural way - with each animal moving at the same time as the others. In order to have this happen, students use a programming technique called parallel processing. Each flower helper’s animation is developed independently, so that students can focus on how each animal moves. Then students are shown a simple technique to get all of the animations to run at the same time.

Try different numbers with right.

\[ \text{rt 90 fd 20} \]
- rt is short for right.
- rt 90 turns the turtle one quarter of the way around.

Although most students easily understand that forward moves the turtle ahead a certain amount, they often expect right to cause the turtle to move forward in a horizontal line. Instead right (and left) make the turtle pivot a certain number of degrees without changing its position. Position and direction or heading are two independent components of the turtle’s state. Forward and back change the turtle’s position. Right and left change its heading. Keeping these components independent makes them easier to control.

The turtle moves in a circle. It goes forward a little and right a little, 36 times. To move in a larger circle, type:

\[ \text{repeat 36 [fd 2 rt 10]} \]

By using forward and right or left, it’s possible to have the turtle move (or, if the turtle’s pen is down, draw) various geometric shapes. For example, to have the turtle draw a square, it must move forward a certain amount and turn right 90 degrees (or a right angle), repeating these instructions four times. In Logo terms:

\[ \text{repeat 4 [fd 40 rt 90]} \]
The turtle ends in the same position and heading in the same direction as when it started. It has turned a total of 360 degrees (4 * 90). In Logo, this is usually referred to as the Total Turtle Trip.

Using the same concept, it’s easy to draw different geometric shapes, such as:

triangle
\[
\text{repeat 3 [fd 50 rt 120]} \quad \text{---} \quad 3 \times 120 = 360
\]

pentagon
\[
\text{repeat 5 [fd 50 rt 72]} \quad \text{---} \quad 5 \times 72 = 360
\]

hexagon
\[
\text{repeat 6 [fd 50 rt 60]} \quad \text{---} \quad 6 \times 60 = 360
\]

Taking this idea one step further, students can draw a circle by increasing the number of turns, making the angle smaller.

\[
\text{repeat 36 [fd 2 rt 10]} \quad \text{---} \quad 36 \times 10 = 360
\]

or

\[
\text{repeat 360 [fd 1 rt 1]} \quad \text{---} \quad \text{This makes a very large circle.}
\]

Once students have completed this project, you may want to spend some time letting them explore different shapes. Before beginning, remind them to type in the Command Center:

\text{pd} \quad \text{---} \quad \text{Puts the turtle’s pen down for drawing.}

\text{pu} \quad \text{---} \quad \text{Picks up the turtle’s pen for animation and moving.}
To see all the instructions work together, go to the Procedures page and write a fly procedure. For example:

```plaintext
procedure fly
    repeat 18 [fd 2 setsh "humbird1" rt 10 setsh "humbird2"]
end
```

If students have trouble understanding why these instructions are used, remind them that to get the turtle to move in a half circle, they type:

```plaintext
repeat 18 [fd 2 rt 10]
```

(If they are not sure why this is so, remind them that $18 \times 10 = 180$, which is half of 360—the number of degrees in a complete circle or the Total Turtle Trip.)

To get the “hummingbird” to flap its wings, they type:

```plaintext
setsh "humbird1"
setsh "humbird2"
```

The fly procedure puts these instructions together.

You may also want to discuss why it’s a good idea to put these instructions into a procedure and not just type them into the turtle’s dialog box. It would be difficult to see the complete line of instructions if it was in the dialog box. Not seeing the whole line makes it more difficult to edit the instructions or correct any typographical mistake. Anytime a set of instructions begins to grow, and especially if it includes multiple sets of brackets, it is a sign that these instructions should all be put into a procedure.

As students go through the projects and activity sheets and gain more practice in procedure writing, they will begin to use procedures more readily in their own projects.
There are two ways to run a procedure:

- Directly, by typing its name in the Command Center or putting it in a button or turtle.
- Indirectly, by running the procedure inside the definition of another procedure.

When you run the fly procedure, you are running the procedure hover indirectly. Fly runs it for you.

Hover is called a subprocedure. Fly is called a superprocedure.

Students may be confused by the use of seth instead of right or left. Seth always sets the turtle in the direction indicated by its input. For example, if students type seth 90, the turtle will always be heading towards the right side of the screen, no matter in what direction it was heading when the fly procedure was stopped. Right 90, on the other hand, turns the turtle right 90 degrees relative to where it was heading when it was stopped. If students use the stop keys (Ctrl+Break/ evade) to stop the fly procedure, they may not know in which direction the turtle is heading when it is stopped. This heading will be different each time it’s stopped. For example:

The hummingbird-turtle stops turtle heading 135
right 90 turtle heading 225
The hummingbird-turtle stops turtle heading 45
right 90 turtle heading 135

Now, edit your fly procedure so that the hummingbird moves a little and then hovers. Use hover in the fly procedure just like any of MicroWorld’s built-in words.

Use seth to change the turtle’s starting direction. Go to the Procedures page and edit your procedure so that the turtle is heading in the same direction each time it starts to fly.
Drag the hummingbird-turtle to where you want it to start. Type:

\texttt{show pos}

MicroWorlds has two types of built-in words—commands and reporters. Commands like \texttt{forward} and \texttt{hidetext}, tell MicroWorlds to do something.

\texttt{Pos} is a reporter. Reporters “report” information about something, providing information for commands. In this case, \texttt{pos} reports the position of the turtle to show. Reporters always report their information to a command.

If the turtle is in the center of the screen and you type \texttt{pos} in the Command Center, you get the following message:

\texttt{I don’t know what to do with [0 0]}

The first number is the turtle’s x coordinate. The second number is its y coordinate.

A turtle’s position on the page is determined by its x coordinate (its horizontal position) and its y coordinate (its vertical position). The center of the screen is position [0 0]. You may want to illustrate this on the board.

Show students that x and y coordinates may be negative, positive, or 0.

Move the hummingbird-turtle to the side. Type:

\texttt{setpos [-50 10]}

Remind students to use the brackets around the input to \texttt{setpos}. The brackets show that the input is a list, since a set of two numbers is required. (See Logo Programming in the Online Help / User’s Guide for more information about lists.)
Now, drag the butterfly-turtle to where you want it to start and find its position. First type:

```
butterfly,
```

Turtles are objects like text boxes. Remind students that they need to name and “talk to” turtles in the same way as they named and “talked to” text boxes in the first project. When students are using just one turtle, they do not need to use a name in order to get the turtle to follow the instructions. But, as students begin to use more and more turtles, it becomes necessary to have a way to address each one so that the right turtle is affected by the instructions. Although you can control the turtle affected by the commands by clicking on it before typing commands, this will not work if a project is under procedure control. Using a “comma command” helps avoid any confusion.

It’s also a good idea to remind students that a turtle’s name should reflect their function (in other words, hummingbird is better than t1).

```
Edit your setup procedure on the Procedures page adding this position.

to setup
  Add hummingbird, since there are two turtles.
  butterfly, setpos [-50 10]  Use your turtle’s position.
  hummingbird, setpos [-50 10]
end
```

If, when students try their setup procedure, the hummingbird doesn’t move, they may have forgotten to add the instruction `hummingbird`, to the first line of the procedure. Remind them why they need to add this instruction. If the student has done this, check that the name isn’t misspelled in either the procedure or the turtle’s dialog box.
When you click on a turtle that’s been programmed to run an instruction line, you are starting an independent process (set of instructions). By using everyone [clickon], students start several processes that run concurrently. One turtle begins running the instructions in its dialog box, and immediately after it begins, the second turtle begins to run the instructions in its own dialog box. The second turtle does not wait for the first turtle to finish all its instructions before starting. These independent processes run at the same time. In programming, this is called parallel processing.

Students do not need to know that they are doing parallel processing. What they see happen is what they expect to have happen. In real life, many processes occur at the same time. Parallel processing makes it possible for students to create projects that make things happen more like they do in real life. (For more information, refer to Processes and Synchronization in the Online Help / User’s Guide.)

Use the Melody Editor or Import Music to play music on your page. For example, try a few lines from “Flight of the Bumble Bee” or think of another tune. Browse the musical pieces in the Media folder. Leave the Melody or Music icon on the page and click on it to start your tune.

If students add their melody name (which becomes the instruction to play the melody) to their fly or flutter procedures, no other instructions for that turtle would run while the melody is playing. In order to have the melody play while the hummingbird is flapping its wings and moving, for example, students need to launch an independent process. By using the Melody icon, they can do this. There are also commands that let students do this. (See Online Help / User’s Guide—Processes and Synchronization.)
THE PLANT Wiz QUIZ

Content Objectives:

- Research and learn various facts about plants in order to create an interactive quiz to give to classmates.

MicroWorlds Learning:

- Opening a dialog box to ask and get answers to questions
- Writing conditional statements that start with if
- Using text box names as reporters

CLASSROOM DISCUSSION

Since the students are each researching their own topics use discussion time to stimulate interest in plants and to show how essential plants are to life on earth. Start by discussing the following statement with your class: “One could easily argue that photosynthesis is the most important biological process on earth.”

Ask students why photosynthesis is so important. In what ways do people benefit from photosynthesis?

Most plants’ energy needs are filled by sunlight, which is pure energy. But sunlight is not in a very useful form—it must be converted to other forms. Photosynthesis is the process by which plants store the energy from sunlight as chemical energy, the type of energy used by living things. For example, plants use the energy of sunlight to change carbon dioxide from the air into sugars, starches, and other carbohydrates that can be stored. In so doing, oxygen is released. Fortunately, plants often store—in roots, stems, fruits, or seeds—more food than they can use. We get energy from eating the plant itself or its products. By studying photosynthesis, scientists hope to learn how to efficiently convert solar energy to other forms of energy.
How else is photosynthesis related to energy? How is it related to other materials we use, such as cloth and building materials? Ask students if they know the source of coal, oil, and natural gas. These are all derived from ancient plants and animals and the chemical energy in them originally came from sunlight converted through photosynthesis.

Once students are aware of the role of photosynthesis, you can discuss a number of different ideas. For example, one theory that scientists have about the disappearance of dinosaurs says that a huge meteor struck the earth, sending up enormous dust clouds. These dust clouds were large enough to greatly reduce the amount of sunlight that reached the earth. What may have happened then?

**OFF-COMPUTER ACTIVITY**

Try to conduct a variety of small experiments to provide information about different aspects of plant life. The following are some recommended activities:

1) In clear plastic cups, lined with toweling, germinate 10-20 lima beans and watch in which direction the roots grow. About a week after the roots have begun to grow, turn the seeds upside down, so that the root is pointing upward. Have students observe what happens to the roots after a week.

2) Germinate 20-30 pea seeds in soft toweling under indirect light. As soon as the root hook appears, plant the seeds in potting soil in two different pots. Put half the seeds in an unlit area (for example, a closet or a closed box). After two weeks, compare the two groups of seedlings. Have students describe the differences. What happens if both pots are now left in the light for a week? Have students compare the roots of seedlings from each pot.

3) What is transpiration? Place a glass bottle or jar over a small plant or seedling and place it in the sun. After a few days, water vapor appears. Ask the class where the
water comes from and what they think is the reason for transpiration.

4) Water is necessary for plants to carry out the many processes that occur within them, such as photosynthesis. How does water travel through a plant? Take a celery stalk or a white carnation. Cut the stalk or stem in half from the base to half way up to the top. Place one side of the stalk or stem in water. Place the other half in ink or water in which blue or red food coloring has been added. Put the experiment in a sunny place. After a few days, what do you see?

WORKING THROUGH THE PROJECT
Several studies have shown that when children act as teachers, they learn as well as teach. For example, in one study in which children were asked to use Logo to create software that would teach other children about fractions, the children-designers acquired as much or more knowledge about fractions than children in a control group. As your students design a quiz about plants, they actively participate in their learning. They will also be using new types of MicroWorlds Logo instructions. One type allows information to be added to a program as the program is running. The other, conditional statements, make it possible for students to design projects that can test conditions and then respond in different ways depending on the outcome of the test. Understanding conditional statements will make the design of more complex simulations and games possible.

Before Starting on the Computer
Planning is the key to this project and the next. Students choose a topic relating to plants and do enough research to create a five question “yes or no” quiz. They should know the correct answers to each question and have some background information on why this is the answer.
Students may have answered the question by typing 1, so they won’t hear a cheer. If the first input to if is false, nothing happens. In this case, they’ll hear a sound when they try the next instruction.

If waits to see if its first input (answer = 2) is true or false. It runs the list of instructions only if the first input is true.

For example, if students were to type the number 1 as the answer and then type:

```
show answer = 1
true
```

This is what would be printed.

True, in this case, has nothing to do with whether or not the correct answer to the question is 1. It only tells you if this was the answer that was given after the question command was used.

Students may be surprised by the use of the equal sign. This instruction is the same as saying:

If the answer you typed was 2, MicroWorlds will cheer.

If a student gets the message:

```
I don’t know how to answer=1
```
or

```
I don’t know how to =1
```

This indicates that there is no space between the word answer and the equal sign (=) and/or between the equal sign and the 1. Remind them that spaces are needed on either side of any infix math reporters (see Online Help / User's Guide for more information).

Sounds such as cheer must be imported into a project. To do this, students must use Import in the File menu and find the sound they need in the Media folder (Windows) or the Sounds and Movies folder (Macintosh). They must follow these steps for each sound they need in the project.
Start by writing a question1 procedure that asks the first question and lets the player know through sound effects if the selected answer is right or wrong.

Each question in the quiz will have its own procedure. Remind students to add a number after the word question for the procedure name. There is no space between question and 1.

First, create a text box, Text1. Go to the Procedures page and use the select and copy keys to copy your first question. Go to your page and paste the question into Text1:

Putting each question into its own text box is a very concrete way for students to think about how question works. Text1 is the name of the first text box created on a page. The name of the text box (in this case, Text1) is a reporter, reporting its contents to the command question. By using this technique:

- The question procedures become easier to read, write, and understand.
- Students are less likely to forget to use a second bracket as often happens when they have to put a long sentence in brackets.
- It is easier to edit the question because it’s easier to edit a text box than a sentence written in the middle of a procedure.

To see a hidden text box, use the eye tool. This makes all hidden text boxes visible.

Hidden text boxes can be placed anywhere on the page. Suggest to students that the boxes not overlap, so that it will be easier to find each one and make it visible if editing is required.
CHAPTER 3: THE PLANT WIZ QUIZ

If you need to edit your question or if it is too long and not all the words are showing, just edit the text in the text box. First, click the Eye tool and the text box will be visible. Then click on visible in the dialog box. Edit your text. Remember to hide the text box when you are finished editing.

More text can be included in a text box than in the dialog box that appears when using the question command. Students may end up losing the last lines of their questions. Remind them to keep the questions short. If they can't shorten their questions, suggest that they include a text box on their screen that contains general instructions, such as:

“For each question, type 1 for yes and 2 for no.”

For example, for question1, include this line:

```plaintext
if answer = 1 [oops announce [Better luck next time.]]
```

The use of double brackets may be confusing. Each of the closing brackets corresponds to one of the opening brackets (one set for the list of instructions following if, the other for the list of words that is part of the announce instruction). As students begin to write more complex procedures, they often write long lists of instructions. What ultimately happens is a loss of control over what is happening within the procedure, plus a lot of confusion. If there is a bug in the procedure it is much harder to find.

Although it isn't necessary to write a subprocedure at this time, doing so would make the question1 procedure much easier to understand and easier to debug if there is a problem.
Write a **correct** procedure so that the star flashes and you hear the cheer when the answer is correct.

Students will probably first think to call this procedure “right.” But **right** is a built-in word in MicroWorlds vocabulary and cannot be used as a procedure name. If they forget and use the word right as a procedure name, they get this message in the Command Center when they go to a project page:

**right already used**
CHAPTER 4

GREEN GAMES: LOST ON EARTH

The form of this game is much the same as the quiz in the previous project. It is strongly recommended that students do not start this project if they have not done the previous project—the Plant Wiz Quiz.

Content Objectives:

- Research and learn various facts about biomes in order to create a game in which clues are hidden in different places around the world.

MicroWorlds Learning:

- Creating a game completely under program control
- Using a startup procedure
- Reinforcing learning about techniques for interactive programming
- Reinforce learning about conditional statements using if

CLASSROOM DISCUSSION

Discuss what a biome is. Students are probably familiar with the term ecosystem, the interactions of plants, animals, geographic conditions, and weather. How does it relate to a biome? Biomes are collections of ecosystems in a region with the same climate. In this case, climate refers to weather changes over a long time, where weather is determined by day-to-day conditions.

There are at least nine major land biomes: tundra, desert, chaparral (scrublands), treeless grassland, savannah, temperate coniferous forest, boreal forest, deciduous forest, and tropical rain forest. Ask students if they can name any biomes. To help them get started, name one, for example, the tundra or the desert.
Ask students what biome they live in. Have them describe their biome—the climate, seasons, animal and plant life. What other places in the world are in the same biome?

Bring in pictures of the different biomes and show where they are located on a world map. For example:

<table>
<thead>
<tr>
<th>BIOME</th>
<th>LOCATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUNDRA</td>
<td>Greenland, extreme northern part of Canada</td>
</tr>
<tr>
<td>DESERT</td>
<td>southwestern United States, Middle East,</td>
</tr>
<tr>
<td>CHAPARRAL</td>
<td>southern Europe (for example, large parts of Italy), Mediterranean coast of Africa</td>
</tr>
<tr>
<td>TREELESS GRASSLAND</td>
<td>plains area of the United States</td>
</tr>
<tr>
<td>SAVANNAH</td>
<td>central Africa, Australia</td>
</tr>
<tr>
<td>TEMPERATE</td>
<td>northwestern United States, south coast of Chile</td>
</tr>
<tr>
<td>CONIFEROUS FOREST</td>
<td>northwest coast of North America, south coast of Chile</td>
</tr>
<tr>
<td>BOREAL FOREST</td>
<td>northern Canada, northern Europe</td>
</tr>
<tr>
<td>DECIDUOUS FOREST</td>
<td>western Europe, eastern United States</td>
</tr>
<tr>
<td>TROPICAL RAINFOREST</td>
<td>southeast Asia (for example, Malaysia), parts of South and Central America</td>
</tr>
</tbody>
</table>

A biome may encompass many ecosystems. Examples of the desert biome include the Negev Desert of Israel, the Mojave desert of southwestern United States, the Gobi Desert of Mongolia, and the Sahara Desert of Africa.

You may also want to discuss with students what links the elements (both plant and animal) within an ecosystem. If they are not sure, emphasize the idea of links. When you think of links, you often think of a chain, in this case, a food chain. In general, energy flow links the elements of a ecosystems. The different components of an ecosystem interact in both the flow of energy through the system and the cycling of elements in it. Discuss what energy flow has to do with food chains. From where does the energy originate in a food chain?
OFF COMPUTER ACTIVITIES

Divide the class into several groups and assign a different biome to each group. Have each group build a model (or chart) of its biome. The model should include both the plants and animals found in these areas, and indicate the type of terrain generally associated with the biome. Have students describe the climatic conditions that define the biome and list where the biome is located both locally and worldwide.

As a class project, you may want to create a miniature version of a biome, for example, the desert biome, using real plants and plastic models of some of the animals that inhabit this biome. Since plants need water, sunlight, and, in some cases, food, this will give you an opportunity to discuss the desert climate and the amount of precipitation and light these plants and animals require.

WORKING THROUGH THE CHAPTER

In this project, students get a chance to use what they’ve learned about interactive programming, conditional statements, text boxes, and turtles to develop their own computer game. Some students may notice similarities between the format of this game and other games they’ve seen on computers. Although simpler, this game includes some of the basic features of popular video and computer games. The structure of the programming in this project is the same as that in Chapter 3, the Plant Wiz Quiz. The main difference is that in this project, each question procedure calls its own wrong or correct procedure. Students are introduced to recursion, another powerful programming and math concept.

Before Starting on the Computer

Although this project outlines the type of game that the students will build, they still need to do a great deal of work on their own. In order to start their research, have students read the planning section of the project several days before they begin their on-computer work.
Briefly review the general idea of the game. It is important that you help students make the programming connection between this project and the Plant Wiz Quiz. You may want to draw a diagram on the blackboard showing the flow of the project.

Explain to students they must select and do research about the plants on which the clues will be placed, where these plants grow, and where these plants do not grow. Encourage them to be original in their choice of plants and locations and not merely copy those in the project book.

This project involves both a scientific research component and a creative writing component. The game is based on an adventure story the students create. Although a model is presented for students, encourage them to think of their own story line. The only constraint is that to solve the mystery presented, clues must be scattered on plants around the world. Let students prepare their story idea and introduction before beginning their work on the computer.

- Give the turtle the same name as its country.

If a country has more than one word in its name, students should use the country’s initials, instead. For example, instead of United States, use USA.

If students use a multi-word name (such as United States) and type:

United States, st
I don’t know how to united

This message appears.

- Now, go to a new page – Page1.

It’s important for students to leave the page names as Page1, Page2, etc… First, each page number corresponds to the number used for the question procedure on that page. Second, students may unwittingly give clues about
what answer to pick for each question in the story. For example, if the first correct answer is Mexico, students may naturally think to name the page containing the illustration of the Mexican desert, Mexico. The first answer, Mexico, would appear in the page list in the Pages menu. Leaving the page names as Page1, Page2, Page3, avoids this problem.

In this game, each question needs its own correct and wrong procedures because something different will happen for each correct or wrong answer. For example, each correct answer leads to a new clue.

This part of the project is different from the Plant Wiz Quiz project. Since each correct answer leads to a new clue, the instructions after a correct answer is picked will be different for each question. The overall structure of each correct and wrong procedure will be the same, though.

If students begin to get confused, show them that each question procedure contains:
- this instruction—question clue
- a subprocedure named correct
- a subprocedure named wrong

All the numbers (the x's) should be the same within a question procedure (for example, question1 contains clue1, correct1, and wrong1).
Students may need to explore ways to synchronize all the events in this procedure. For example, the star may still be flashing when the announce instruction is run. This can be fixed in one of several ways. The instruction in the Mexico-turtle's dialog box may be changed so that the ht st instruction line doesn't repeat as many times. Or a wait instruction may be added. Students should experiment with various strategies until they are satisfied with the results.

Although some sounds are already included in this project, remind students that if they want to use different sounds, they must use Import in the File menu and find the sound they need in the Media folder (Windows) or Sounds and Movies folder (Macintosh). They should follow these steps for each sound they need that is not already available in the project.
In this project, \texttt{question1} runs the \texttt{wrong1} procedure and then \texttt{wrong1} runs the \texttt{question1} procedure. This is an example of recursion. Since students should already have mastered the idea of superprocedures and subprocedures, recursion can be introduced as a procedure that runs itself as a subprocedure. In this case, the procedure, \texttt{question1}, does so indirectly, by running another subprocedure (\texttt{wrong1}) that runs \texttt{question1} as a subprocedure.

You may want to discuss recursion with the class as a whole, using a simpler example. For instance, ask students what happens in this procedure:

\begin{verbatim}
  to drive
  fd 1
  drive
  end
\end{verbatim}

Try it on the computer. To stop, press the \texttt{stop} keys (\texttt{Ctrl+Break}/\texttt{-}).

MicroWorlds runs one line of a procedure, then goes on to the next line of the procedure and runs it, and so on, until the end of the procedure. Demonstrate to the class how MicroWorlds would run the \texttt{drive} procedure. The first line of the procedure gives the instruction to move the turtle forward 1 step. The second line of the procedure gives the instruction to run \texttt{drive}. The first line of \texttt{drive} tells the turtle to move forward 1 step. The second line of the procedure gives the instruction to run \texttt{drive}, and so on.
You may want to tell this recursive riddle: “If you had two wishes, what would your second wish be? Answer: Two more wishes.”

Since wrong1 is running question1, we need to add a stop in the question1 procedure. Edit the question1 procedure:

It is absolutely necessary to end the recursive procedure with stop in order to avoid having the correct1 procedure run twice. The reason this happens may be very confusing to students. You may want to draw a diagram that shows what happens as each line of these procedures are run (if stop is not added).

```
to question1
  question clue1 Type: 1
  if answer = 1 [wrong1]
    to wrong1
    map
    Siberia, clickon
    oops
    announce [Cacti grow in the desert. Siberia has tundra, forest, and grassland areas.]
  end
  to question1
  question clue1 Type: 2
  if answer = 1 [wrong1]
    if answer = 2 [correct1]
    end
  end
end
```

```
if answer = 2 [correct1]
end
```

Now, this line runs.

If the wrong answer is given (answer = 1) the complete subprocedure wrong1 must run before the last line of the question1 procedure (if answer = 2 [correct1]) runs.
MicroWorlds doesn’t forget about this last line—it merely waits until all the instructions before that line are complete before running it. MicroWorlds holds on to that line—saving it for the right time. In this case, MicroWorlds saves that line until wrong1 runs all its instructions. The final instruction in wrong1 is to run question1 a second time. The second time question1 runs, instead of selecting 1 as the answer to question, students select 2. This gives answer a new value. As question1 runs this second time, the following happens:

```plaintext
if answer = 1 [wrong1]     goes to the next instruction.
if answer = 2 [correct1]    This is true, so correct1 runs.
```

This means, the second time question1 runs, it runs all its lines of instruction, runs correct1, and the second question1 ends. But, still waiting to run is the last line of the first question1. The last line is:

```plaintext
if answer = 2 [correct1]    Now, this is true, so correct1 runs again!
```

This will be confusing for students, so go through the procedures line by line, asking the students to describe what happens each time.

If the wrong answer is 2, this doesn’t occur. Ask your students if they can figure out why this is so.

---

**AND SO ON**

Follow the same steps for the next pages, until the player reaches the solution. In the sample story the player is looking for a map. This special map can be found in the picture on the next to the last page. If the player answers a final question correctly, he or she finds the missing botanist on the last page.

Each page is constructed following the same sequence of steps as the first two pages. If students are not sure what to do, they can follow the instructions for Page1, substituting the number that corresponds to the page.
number on which they are working for the number 1 in the question (including the clue text box name), correct, and wrong procedures.

You may want to set a limit to how many clues they’ll need for the game—5 should be the maximum. Students should think of their own ending for their games. If they need some help, the following is an example of one way to end:

On the next-to-the-last page, students draw or import the appropriate background for the plant that held the previous clue, but also include a picture of a map of the country where the mystery will be solved (in the example, where the botanist is found). The last question procedure, which runs when students click on a plant from the country shown on the map (and not found in the region pictured in the background on the page), asks what country is shown on the map. The question procedure for this next-to-the-last page is the same as for the other pages. The last page shows, through graphics, animation, and sound the successful conclusion to the mystery.

**Action**

Design an animation on each location page to show players that they are on the right track.

Getting the action to occur at the right time may be difficult for some students. Have them think about each instruction in their correct procedure in order to figure out where the instruction for the animation should be. Since it will usually be on the page after the one with the clue, the instruction preceding the animation instruction should be a page name (Page2 or Page3, etc.).
WHERE TO GO FROM HERE

LCSI has an Internet Web site where you can see projects that other students and teachers did, and find new project ideas. Sharing projects on the Web motivates students to take pride in their projects.

The web site address is: http://www.lcsi.ca

As well, take a look at the projects included with the MicroWorlds CD ROM, called MicroWorlds Samples. These projects will give you an idea of the range of projects you can do with MicroWorlds.
REFERENCES

Most of the information found in the module came from the World Wide Web through the following sites:

**Arizona State University**
http://photoscience.la.asu.edu/photosyn/study.html

**University of Alberta**

**Okanagan University College**
http://www.arts.ouc.bc.ca/geog/G11/8k.html

**Living Earth** (an international organization for environmental education)
http://www.gn.apc.org/Living_Earth/index.html

**University of California Museum or Paleontology**
http://www.ucmp.berkeley.edu/help/taxaform.html

**Missouri Botanical Garden**
http://www.mobot.org/mbgnet/index2.html

Also of interest:

**Internet Directory for Botany: Arboreta and Botanical Gardens**
http://www.helsinki.fi/kmus/botgard.html

**Eduzone**
http://www.eduzone.com
# Table of Contents

## The Living World – Plants

### Chapter 1  Flower Facts  .................................................. 2
- Tell It Like It Is .......................................................... 2
- Take Control .............................................................. 4
- More Words .............................................................. 5
- Getting Things Started ................................................. 6
- A Better Button .......................................................... 7
- On Your Own ............................................................. 9

### Chapter 2  Flower Helpers ............................................... 10
- Gardening ................................................................. 10
- Birds and Butterflies ................................................... 11
- Hovering ................................................................. 13
- On Your Mark ........................................................... 13
- More Helpers ............................................................ 15
- Get Set ................................................................. 15
- Everyone ............................................................... 16
- Putting It All Together ............................................... 18
- On Your Own .......................................................... 18

### Chapter 3  The Plant Wiz Quiz ....................................... 19
- Planning ................................................................. 19
- Asking Questions ...................................................... 20
- What If? ............................................................... 21
- Making Your Quiz .................................................... 22
- Tricks of the Trade .................................................... 23
- Questions, Question, Questions ................................. 24
- Using Announce ....................................................... 24
- Add Some Flash ........................................................ 25
- On Your Own .......................................................... 26
The Living World – Plants

The cycle of nature links all living things. In this cycle, plants use energy from the sun plus carbon dioxide to make food. During this process, the plants give off oxygen. People and animals eat the plants and fungi for food (energy) and breathe the oxygen. They, in turn, breathe out carbon dioxide. When plants and animals die, they decay. Fungi get their food from decaying matter. Bacteria assist in the process of decomposition and this, in turn, helps return minerals to the earth. Plants use these minerals to grow.

We see plants almost everyday—whether they’re flowers, trees, or grass. Life on earth could not exist without them.
In this chapter, you add text to a diagram of a flower. The text explains what each part of the flower is.

**TELL IT LIKE IT IS**

When you look closely at a flower, you see that it’s made up of four main parts—the calyx, the corolla, the stamens, and the pistils. Some of these, in turn, have several parts of their own. At the tip of the plant’s stem is the receptacle or base of the flower. Each part helps the flower make seeds so that new plants will grow.

Before you begin, make sure you have the information that you prepared about the different parts of a flower.

Open the Flower project.

On the first page you’ll see a diagram of a flower.
FLOWER FACTS

It would be useful to have some information appear about each part of the flower, but only when you want it. That means you must make information in a text box appear and then disappear.

The largest and most noticeable part of a flower is the corolla, which consists of all the petals of the flower. Start with it.

First, create a text box. Make it large enough for a few sentences.

Type some information about the corolla.

```text
Corolla
All the petals of the flower together make up the corolla. The petals are the most colorful part of the flower. Their bright color attracts insects and birds.
```

Make sure all the text is showing. You may want to change the style, size or color of all or part of your text.

In this project, you will use several different text boxes. To avoid confusion and make sure that the correct one appears each time, give each text box a meaningful name. To do this:

- Click on the and then on the text box. When the dialog box appears, name the text box Corolla, since it has information about the corolla.
- Select transparent and click OK.
In the Command Center, type:

`corolla,`  
Don’t forget the comma.

The comma (,) stands for talk to. This instruction means that the next instruction you type will address the text box named Corolla.

Type:

`hidetext`  
The text box named Corolla disappears.

Now type:

`showtext`  
The text box reappears.

Use these commands to control when the text box will appear.

**TAKE CONTROL**

Rather than typing instructions, it would be easier if the text appears and disappears by just clicking on the corolla. Program the color of the corolla so that when you click on it the text appears for a few seconds, and then disappears.

- Open the Drawing Center.
- Select pink (the color of the petals) and click on it again.
FLOWER FACTS

In the space for mouse instruction, type:
```
corolla, showtext wait 80 hidetext
```

Click OK.

**Wait 80** makes the computer wait for 8 seconds before hiding the text box.

Hide the text box then click on the pink petals on the diagram. Do you see your text box? You can adjust the amount of time the text box is showing by changing the number input to **wait**.

MORE WORDS

Next, create a text box and type some information about the stamens. Make sure all the text shows. Name the text box Stamens and make it transparent.

```
Each stamen has two main parts. The tip of the stamen is called the anther...
```

[stamens]
Open the Drawing Center and program the green color—the one used for the stamens—so that this Stamens text box will appear when you click on the stamens in the picture.

![Instructions for green](image)

Try it.

Now, follow the same steps to program each of the other colors in the diagram to present information about the different flower parts.

Remember to try each color to make sure it works. Clicking on each plant part should display its information.

**GETTING THINGS STARTED**

Let's create another text box to give instructions about how to find the information in the diagram.

![Instructions](image)

Give your text box the name Instructions.
FLOWER FACTS

Program a button to make the text box appear:

1. Select the button.
2. Click the place on your page where you want the button to be.
3. When the dialog box appears, type:
   `instructions, showtext wait 80 hidetext`
4. Leave Do It set to Once. Click OK.

Before trying the button, type in the Command Center:
`instructions, hidetext`

Now, click on the button. (Remember to change the number input for `wait` to adjust the timing.)

A BETTER BUTTON

The button works, but you can’t see the words written on it. You could enlarge the button to see all the instructions. Or, you can make a better, clearer button, with just one word on it.

First, change the button:

1. Click on the and then on the button.
2. Select the text in the instruction line and, using the cut keys, cut it out of the dialog box.
Now type information as the instruction.
Click OK.

Make your button large enough to see the word information.
Click on it.—information

What happens? A message appears in the Command Center:
I don’t know how to information
MicroWorlds has its own built-in set of words that it already knows. Showtext and hidetext are two examples. Information isn’t a built-in word. You need to add a new word—to do exactly what you want to happen.

You can add new words to MicroWorlds vocabulary by writing procedures. Procedures act just like MicroWorlds built-in commands except you can edit or erase them and they work only in the project in which you made them.

To write a procedure called information:
Go to the Pages menu and choose Procedures.
You are now on the special page that contains all the project’s procedures. Right now it’s empty.
FLOWER FACTS

ON YOUR OWN

Use these techniques to create other diagrams. For example, import a map of your country. Use different colors to show different weather regions. (Remember not to use different shades of the same color!) Create a text box for each region and describe its flowers or trees. Program each color so that when you click on a region, information appears.

Go back to your page by clicking on the page name in the Pages menu. Now, when you use the word information, it will run all the instructions in the procedure.

Try your button — information

Congratulations! You’ve just added a word to MicroWorlds.

Remember to save your project.
Flower Helpers

In this chapter, you'll create an animated presentation showing some ways that animals such as birds and insects help plants.

GARDENING

In the Flower project, choose [New Page] from the Pages Menu. Start by creating a background for your animation. You may want to import a picture of a garden or draw your own using the Drawing Center.

Use the drawing tools and shapes from the Shapes Center to add lots of flowers to your picture!

Name your page.
BIRDS AND BUTTERFLIES

Think about how a hummingbird moves. They turn in different directions as they fly from flower to flower and they flap their wings very quickly as they move.

To get the turtle to turn, try:

```
right 30
```

Right turns the turtle to the right the number of degrees given as input.

Try different numbers with right.

```
rt 90 fd 20
```

rt is short for right.

rt 90 turns the turtle one quarter of the way around.

```
rt 180 fd 20
```

rt 180 turns the turtle to the opposite direction.

You can also turn the turtle to the left. Try:

```
lt 60 fd 10
```

lt is short for left. Try different numbers.

Now try this:

```
repeat 36 [fd 1 rt 10]
```

The turtle moves in a circle. It goes forward a little and right a little, 36 times. To move in a larger circle, type:

```
repeat 36 [fd 2 rt 10]
```

Set the turtle to the first hummingbird shape.

Try different instructions to make the hummingbird fly. For example:

```
repeat 18 [fd 2 rt 10]
```

The bird flies in a half circle to the right.

Or try:

```
repeat 18 [fd 2 lt 10]
```

Now it flies in a half circle to the left.
To see all the instructions work together, go to the Procedures page and write a \texttt{fly} procedure. For example:

\begin{verbatim}
 to fly
 repeat 18 [fd 2 setsh "humbird1 rt 10 setsh "humbird2]
 end
\end{verbatim}

Try it out. The bird flaps its wings as it moves.

Put the \texttt{fly} instruction in the turtle's dialog box. Name the turtle \texttt{hummingbird}. Set Do It to Many Times.

Try your procedure by clicking on the hummingbird-turtle. Does it look like a hummingbird flying? Is it flying in the right direction? If not, click on the hummingbird again to stop it. (If you can't catch it, press the \texttt{stop} keys: $\text{Ctrl}+\text{Break}$ / $\sigma$...)

In the Shapes Center, click on the turtle shape and then click on your hummingbird. This sets the turtle back to the turtle shape so you can see the direction in which it's heading.

Use \texttt{seth} to change the turtle's starting direction. Go to the Procedures page and edit your procedure so that the turtle is heading in the same direction each time it starts to fly. 

\begin{verbatim}
 to fly
 seth 0 \texttt{The turtle will always be pointing up when it starts.}
 repeat 18 [fd 2 setsh "humbird1 rt 10 setsh "humbird2]
 end
\end{verbatim}

Now, click on the hummingbird-turtle again.
FLOWER HELPERS

HOVERING

Get your hummingbird-turtle to hover. It should flap its wings quickly, but not move. Try this in the Command Center:

```
repeat 10 [setsh "humbird1 wait 1 setsh "humbird2 wait 1]
```

Write a hover procedure.

```
to hover
repeat 10 [setsh "humbird1 wait 1 setsh "humbird2 wait 1]
end
```

Now, edit your fly procedure so that the hummingbird moves a little and then hovers. Use hover in the fly procedure just like any of MicroWorld's built-in words.

```
to fly
seth 0
repeat 18 [fd 2 setsh "humbird1 ....The bird moves a little...
rt 10 setsh "humbird2]
hover ........................................... ... and then hovers.
end
```

Click on the hummingbird-turtle to try your procedure.

ON YOUR MARK

Each time you try your procedure, you need to drag the turtle back to its starting position before you click on it. Instead, move it back with a command.

Each turtle on a page has a position. The position is made up of the turtle’s x-coordinate (its horizontal position) and its y-coordinate (its vertical position).
You can move the turtle by setting it to a different position.

Drag the hummingbird-turtle to where you want it to start. Type:

```
show pos
-50 10
```

Pos stands for position. It reports the turtle’s position. Your answer will be different.

The first number is the turtle’s x coordinate. The second number is its y coordinate.

Show tells MicroWorlds to print whatever is reported to it in the Command Center. Here it tells MicroWorlds to print the hummingbird-turtle’s position.

Move the hummingbird-turtle to the side. Type:

```
setpos [-50 10]
```

Setpos stands for set position. Use your numbers.

The hummingbird-turtle goes to the starting position.

Now write a setup procedure on the Procedures page using this position.

```
to setup
setpos [-50 10] Remember to use the square brackets.
end
```

Go to your page and type setup in the Command Center to try your procedure.

Remember to save your project.
FLOWER HELPERS

MORE HELPERS

Usually flowers attract more than one type of bird or insect. Add another helper.

— Hatch a new turtle and place it on the flowers on the page. Set the turtle to the first butterfly shape.

Think about how a butterfly moves. Is it different from how a hummingbird moves? Butterflies flap their wings and move forward.

Try some instructions in the Command Center. Then go to the Procedures page and write a flutter procedure to get the butterfly-turtle to flutter.

Open the turtle’s dialog box with the Name the turtle butterfly and write flutter as the instruction. Set Do It to Many Times.

Click on the butterfly-turtle.

Does it look like a butterfly flying? Are the wings flapping as it moves? Is it flying in the right direction? If not, turn it so it flies in the right direction.

GET SET

Now, drag the butterfly-turtle to where you want it to start and find its position. First type:

butterfly,

Talk to the butterfly-turtle.
Since there are more than one turtle, using this instruction makes sure that the right turtle shows its starting position.

Next, type:

```
show pos
-75 -10
```

Remember, your numbers will be different.

Edit your `setup` procedure on the Procedures page adding this position.

```
to setup
hummingbird, setpos [-50 10] ; there are two turtles.
butterfly, setpos [-50 10] ; Use your turtle's position.
end
```

Go to your page and try your procedures.

**EVERYONE**

Create a button that gets each helper to automatically “click on” and, after awhile, “click off.”

Click on each turtle to stop it from moving. Type this instruction in the Command Center:

```
butterfly, clickon
```

Type `butterfly`, or the hummingbird moves.

What happens? The command `clickon` acts just like a mouse click on the turtle.

Now type:

```
clickoff
```

The butterfly stops.

Now try this:

```
everyone [clickon]
```
FLOWER HELPERS

The \texttt{everyone} command tells all the turtles to run the instructions in the square brackets. This is a quick way to give all the turtles the same instruction.

\texttt{everyone [clickoff]}

Both the turtles stop.

Write a procedure on the Procedures page that tells each of your turtles to go to its starting position, click on, flutter and fly, and then click off. Name your procedure \texttt{start}.

For example:

\begin{verbatim}
to start
setup
everyone [clickon]
wait 50
everyone [clickoff]
end
\end{verbatim}

Once you’ve written your procedure, go to your page and create a button. Put \texttt{start} in the instruction line in the button’s dialog box.

Try your button.  

\begin{center}
\texttt{start}
\end{center}
PUTTING IT ALL TOGETHER

Your project about flowering plants is nearly complete. Go to each page and try it out. Now link your pages with buttons.

Remember to save your project.

ON YOUR OWN

- Try to add another helper, such as a bee, to the second page of your project. First think about how it moves. Then, write a procedure. Remember, find the third helper's starting position and include an instruction for setting the turtle to this position in the setup procedure.

- Explain what these helpers are doing in a text box on your Helper page. For example:
  
  Birds and insects help pollinate many flowering plants. Pollen grains stick to these animals. The pollen is then carried from one plant to another as these animals look for food.

- Use the Melody Editor or Import Music to play music on your page. For example, try a few lines from “Flight of the Bumble Bee” or think of another tune. Browse the musical pieces in the Media folder. Leave the Melody or Music icon on the page and click on it to start your tune.

Honey is the only food produced by insects that people also eat!
Nectaries are the special glands in flowers that produce nectar. Honeybees suck up this nectar and store it in their honey stomachs where certain chemicals are added to the nectar. Once the bee puts the nectar into a honeycomb, the water in the nectar evaporates and the chemicals turn it into honey.
In this chapter, you will use information about the different types of plants and plant features to design a simple quiz for you and your friends.

PLANNING

In designing a quiz you need to do a great deal of planning before you even begin working on the computer. First, you need to do research on the subject of your quiz. In this case, the subject is plant facts.

Next, you need to design your quiz. A quiz can be very simple. It can be just a series of questions that are answered with either yes or no.
For example, start with five questions:
1) Do some plants manufacture their own food?
2) Do all plants have roots that grow in soil?
3) Are mushrooms plants?
4) Can some plants live without any water?
5) Is it true that plants take in oxygen and release carbon dioxide?

The answer for each question is either yes or no.

ASKING QUESTIONS
Open a new project.

Here is a simple way to ask questions in MicroWorlds. In the Command Center, type:

```
question [Do some plants manufacture their own food? Type 1 for yes and 2 for no]
```

Remember to use the square brackets.

A dialog box appears containing your question.

The dialog box will only disappear after you write an answer.

Answer the question by typing 1.

Now, in the Command Center, type:

```
show answer
1
```

Answer reports whatever your answer to the question was.
Try another question. Type:

```
question [Do all plants have roots that grow in soil? Type 1 for yes or 2 for no]
```

Pick an answer.

Type:

```
show answer
```

```
2
```

You may have picked 1.

WHAT IF?

But how will the person answering your quiz know what the correct answer to the question is? Use sounds to let them know if their answers are right or wrong.

Photosynthesis is the process by which green plants make their own food. Most photosynthesis takes place in the leaves of plants in small bodies called chloroplasts. Chloroplasts contain chlorophyll which absorbs the energy from the sun and uses it to help carbon dioxide join with water and minerals to make the protein and carbohydrates that a plant needs to live.

Try this:

```
if answer = 2 [cheer]
```

Cheer is the name of a sound in MicroWorlds.

Did you hear a cheer? If not, type:

```
if answer = 1 [oops]
```

This means:

If your answer to the question is equal to 2, make a cheer sound. If your answer to the question is equal to 1, make an oops sound.
If is a built-in word in MicroWorlds. The command if needs two inputs.

- The first is always something that will be either true or false, such as answer = 2.
- The second is an instruction and is always in square brackets. If the first input is true (the answer you chose was 2), the instruction inside the square brackets [cheer] will run. If the first input is false (the answer you chose was not 2), the instruction inside the square brackets does not run and MicroWorlds goes to the next instruction.

**MAKING YOUR QUIZ**

Use question, answer, and if to make your quiz. Use question to ask the question and if and answer to make something happen when the person taking the quiz chooses an answer.

Start by writing a question1 procedure that asks the first question and lets the player know through sound effects if the selected answer is right or wrong.

```plaintext
To question1
   question [Do some plants manufacture their own food? Type 1 for yes and 2 for no]
   if answer = 1 [cheer]
   if answer = 2 [oops]
   end
```

Try your procedure in the Command Center. Type:

```plaintext
question1
```

MicroWorlds waits for you to answer the question before going to the next instruction.
Mushrooms are fungi, a plant-like organism. Until fairly recently scientists believed fungi were plants. But because they have characteristics that are different from plants, fungi are now grouped in their own kingdom.

Some fungi are parasites, a type of organism that attaches itself to another plant or an animal in order to get food. Other fungi are saprophytes and get their food from decaying materials.
QUESTIONS, QUESTIONS, QUESTIONS

Now create a second text box. Type in your second question.

Hide this text box.

Write a question2 procedure:

to question2
    question text2 ...... Text2 reports what is in the text box named Text2.
    if answer = 1 [oops]
    if answer = 2 [cheer]
end

Put oops with the wrong answer and cheer with the correct answer.

Write a procedure for each of your other questions.

Make a button for each question procedure. Arrange the buttons on your page. You may even want to draw a background so your page looks like a quiz show on television.

USING ANNOUNCE

It would be fun if something more happened when an answer is chosen. For example, if it is a wrong answer, instead of just hearing a sound, there could be a sound plus a box with a special message.

For example, for question1, include this line:

    if answer = 1 [oops announce [Better luck next time.]]
Chapter 3

The problem with this instruction line is that it is very long and there are a lot of brackets to keep track of. This is a good time to put the instructions into their own procedure:

```plaintext
to wrong
  oops
  announce [Better luck next time!]
end
```

Write this procedure on the Procedures page. Now edit your `question1` procedure so that when the wrong answer is chosen, the `wrong` procedure runs.

```plaintext
to question1
  question text1
  if answer = 1 [cheer]
  if answer = 2 [wrong]
end
```

Edit all your `question` procedures, replacing `oops` with `wrong`.

**ADD SOME FLASH**

Add some pizzazz with a simple animation. For example, make a star flash when the correct answer is chosen.

A star looks like it’s flashing when it continually appears and disappears.

Set the turtle to the star shape. Then type:

```plaintext
ht
  ht stands for hide turtle. The turtle is hiding.
```

Now type:

```plaintext
st
  st stands for show turtle.
```

Try this:

```plaintext
repeat 20 [ht wait 2 st wait 2]
  Slow down the flashing with wait.
```
Once you’ve created one quiz, you can create others. Use these same techniques to create a quiz on another subject. Add new ideas in your correct and wrong procedures, such as playing a song if there is a correct answer or displaying a page with more information when there is a wrong answer.

Open the star-turtle’s dialog box and type in the instruction line:

```
repeat 20 [ht wait 2 st wait 2]
```

Name the turtle star. Set Do It to Once.

Write a correct procedure so that the star flashes and you hear the cheer when the answer is correct.

```
to correct
  star, clickon
  cheer
end
```

The star will keep flashing as the sound is heard.

Now edit all of your question procedures to replace cheer with correct. For example:

```
to question1
  question text1
  if answer = 1 [correct]
  if answer = 2 [wrong]
end
```

Can you think of an animation for a wrong answer? Add it to your wrong procedure.

**Remember to save your project.**

Try your quiz with your friends. Think of some new questions.
You now have all the tools you need to build a more complex game.

In this chapter, you will use your quiz design knowledge to build a game in which players solve a crime by finding hidden clues.

The game begins with a mystery. Clues will be scattered throughout the world. The plants on which the clues are attached won’t belong where they are found and will look out of place.

Each clue describes the type of place where the plant with the clue really grows. Find that place and you find the next clue. For example, if a clue is stuck to a cactus, you need to go to a place where cacti grow to find the next clue.

For this game, you’ll use question, answer, and if.
PLANNING

Remember, all games start with planning. Complete the following steps before you start.

- First, think of a crime or mystery to solve. To make it really interesting, think of a story to go with the mystery. For example:

  A valuable plant, a rare orchid, has been discovered. The botanist who found it sent a message describing her discovery, but forgot to say where it was. Now she’s disappeared. She was last seen in a rain forest in Thailand. No one knows how to find her. The only clue is a map she drew that her evil, but not very able, assistant stole. The assistant keeps delivering plants to the wrong places. Your assignment is to find the assistant and the map and rescue the botanist and her rare plant.

- Next, do research about various plants. Where do they grow? Why do they grow there? Where couldn’t they grow?

Select 5 different types of plants. List a country (or place within a country) where each plant grows and one country (or place within a country) where each plant cannot grow. You need this information to write your questions. Remember to decide on the final country where the mystery will be solved.

<table>
<thead>
<tr>
<th>Plant</th>
<th>Where found</th>
<th>Where not found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cactus</td>
<td>Mexico</td>
<td>Siberia, Russia</td>
</tr>
<tr>
<td>Baobab Trees</td>
<td>Zambia</td>
<td>Antarctica</td>
</tr>
<tr>
<td>Pine trees</td>
<td>United States</td>
<td>Greenland</td>
</tr>
<tr>
<td>Arctic Lupine</td>
<td>Greenland</td>
<td>Bolivia</td>
</tr>
<tr>
<td>Bamboo</td>
<td>China</td>
<td>Sweden</td>
</tr>
</tbody>
</table>

Solution:

Map found in Greenland. Ruthless assistant found here. Botanist found in bamboo groves of China.
Scientists divide the world into biomes. A biome is a community of plants and animals living in large geographical areas that have a similar climate. Each biome is present in several parts of the world. There are at least nine major land biomes including tundra, desert, chaparral, treeless grassland, savannah, temperate coniferous forest, boreal forest, deciduous forest, and tropical rain forest.

**THE MYSTERY**

Open the Games project. The first page is empty. Create a text box and write the beginning of your mystery story.

Name this page **Story**.

**WHERE?**

Go to the Pages menu and select Map. You'll see a map of the world. This map will let players know where each country they’ve selected is.

Look at the places on your list.

Put a turtle on each country on your list. Some countries may appear twice, but put only one turtle on each location. Set each turtle to the star shape. Whenever a player selects a country, the star corresponding to that country flashes.
When you've added a star for each country on your list, hide all the turtles:

```
everyone [ht]
```

**THE FIRST CLUE**

**Background**

Now, go to a new page – Page1.

Draw or import a background that shows the opening scene of your mystery. For example, in the story above, the botanist was last seen in a rain forest. So the first background is a rain forest. Look in the Media folder for backgrounds or draw a background with the drawing tools.

Set the turtle to the shape of the plant that has the first clue. It should be a plant that is not normally found in the rain forest. In this case, it's a cactus plant. Find the cactus shape in the Shapes Center.
Since deserts receive very little rainfall (less than 10 inches—25 cm—a year!), plants such as cacti never grow close together so they don’t compete for water. Some desert plants store large amounts of water in their leaves, stems, or roots. Others have long roots—up to 40 feet! (12 meters!)—to get water from deep beneath the ground. A third type has very tiny leaves so as not to lose water through evaporation from the surface of its leaves.

**Clue**

Create a text box and type your first clue. Don’t make it too long since it will be the text you will use for `question`. For example:

```
Keep on searching where the earth is dry.
Go to 1 - Siberia Russia or 2 - Mexico?
Type 1 or 2 only.
```

Open the text box’s dialog box and name it Clue1. Then hide the text box.

Just to make sure it works, type:

```
question clue1
```

**THE FIRST ANSWER**

- What do you need after a question?
- An answer!
- What should happen after the answer is given?
- The same thing that happened in the Plant Wiz Quiz. If the answer is correct, a `correct` procedure runs. If it is wrong, a `wrong` procedure runs.

In this game, each question needs its own `correct` and `wrong` procedures because something different will happen for each correct or wrong answer. For example, each correct answer leads to a new clue.

On the Procedures page, write a `question1` procedure similar to the procedure in the quiz.

```
to question1
question clue1
if answer = 1 [wrong1]       This is the first correct procedure.
if answer = 2 [correct1]     This is the first wrong procedure.
end
```
What can you include in the correct1 procedure?
- an animation,
- a sound,
- a message…
  … plus, you need to go to a new page for more information

Remember to import the sounds you need into the project. For example, import the cheer and oops sounds.

Write a correct1 procedure. For example:

to correct1
  map .......................................................... Go to the Map page.
  Mexico, clickon ........................................ The star for Mexico flashes.
  cheer ......................................................... There's a cheer.
  announce [You are right. .............................. A message appears.
  Cacti grow in the desert]
  page2 ....................................................... Go to Page2 to find the next clue.
end

Next write a wrong1 procedure. What do you want to happen if the player picks the wrong answer? For example, go to the Map page, the star for Siberia flashes, a sound is heard, and a message appears.

Do you want the player to go to Page2? Not until he or she picks the correct answer.

Should the player see the first question again? That would be a good idea.

Here's an example:

to wrong1
  map
  Siberia, clickon
  oops
  announce [Cacti grow in the desert. Siberia has tundra, forest, and grassland areas.]
  question1 ................................................. Run question1 again so the player gets another chance to answer.
end
You don’t need to return to Page1 to run the question1 procedure again. It will run even if you’re on the Map page as long as each text box containing a clue has a different name.

Since wrong1 is running question1, we need to add a stop in the question1 procedure. Edit the question1 procedure:

```
to question1
    question clue1
    if answer = 1 [wrong1 stop]........Stop the procedure here.
    if answer = 2 [correct1]
    end
```

Try question1.

**ASKING THE QUESTION**

Program the cactus-turtle to make the first question automatically appear.

Open the cactus-turtle's dialog box and in the instruction line type question1.

![Cactus-Turtle Dialog Box](image)

Click on the turtle to try it.
WHAT HAPPENS ON PAGE2

What happens on Page2 is exactly the same as on Page1:

1. First, import or draw a background scene that shows where the next clue is. For example, the cactus holding the first clue grows in a desert. So the second clue will be found in the desert.

2. Set the turtle to the shape of the plant that holds the second clue and place it somewhere on the background. In this story it is a baobab tree.

3. Next, write the question for the second clue in a new text box.

   For example:

   Go to grasslands that each year have a dry season and a rainy season. Go to 1 - Antarctica or 2 - Zambia? Type 1 or 2 only.

   Name the text box Clue2 and hide it.

4. Write a question2 procedure.

   ```
   to question2
   question clue2
   if answer = 1 [wrong2 stop] ...Remember stop here.
   if answer = 2 [correct2]
   end
   ```

5. Write a wrong2 and correct2 procedure. Remember to return to your Map page in order to get the star to flash.

   ```
   to correct2
   map
   Zambia, clickon
   cheer
   announce [You’re right! Baobab trees grow in the savannahs of Africa.]
   page3
   end
   ```

Savannahs are grasslands in areas that each year have a rainy and a dry season. Mostly grasses grow on them along with a few trees and shrubs. Grass stops growing during the long droughts, but as soon as the rain returns, its roots send up new growth. Only trees that can withstand long dry periods, such as baobab trees, can grow on a savannah.
There are two kinds of tundra regions - the Arctic tundra and alpine tundras. Alpine tundras are on the tops of mountains, at altitudes higher than those at which trees can grow. Arctic tundras are in latitudes north of the area where trees grow. Although no trees grow on the tundra, many small plants, such as moss, lichens, grass, and even different types of flowers, grow there.

Now program the baobab-turtle to run `question2` when you click on it.

<table>
<thead>
<tr>
<th>Name:</th>
<th>baobab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction:</td>
<td><code>question2</code></td>
</tr>
<tr>
<td>Do It:</td>
<td>Once</td>
</tr>
</tbody>
</table>

### AND SO ON

Follow the same steps for the next pages, until the player reaches the solution. In the sample story the player is looking for a map. This special map can be found in the picture on the next to the last page. If the player answers a final question correctly, he or she finds the missing botanist on the last page.

Remember, for each page you need to:
- Import or draw a background
- Set the turtle to the shape of a plant not normally found in that area
- Write a question for a clue in a text box. Name the text box and hide it.
- Write three procedures for each clue—a `question` procedure, a `correct` procedure, and a `wrong` procedure. Add a number to the name of each procedure (for example, `question4`).
- Program the turtle so that clicking on it brings up the next clue.
PUTTING IT ALL TOGETHER

Now, put everything together. First, write a procedure to begin on Page1.

```
to begin
    page1
end
```

Now go to the Story page and add a button next to your story. Type in the `begin` instruction.

Try the button and each of your answers.

Other finishing touches may include:

- Additional instructions on your Story page telling players to click on the plant that appears out of place in each scene.
- A frame for the text box with the story.

STARTING ON THE RIGHT PAGE

When you open a project in MicroWorlds, Page1 is usually the first page shown. But in this game you want the Story page to be the first page to show. Write a `startup` procedure to display Story when the project opens.

`Startup` is a special name for a procedure that runs the moment a project is opened.

Write the startup procedure:

```
to startup
    story
end
```
GREEN GAMES – LOST ON EARTH

Close your project. Then open it again to see if Story is the first page that appears.

The **startup** procedure could also include sound effects or music.

**Remember to save your project.**

---

**ON YOUR OWN**

Once you’ve completed several clues, questions, and answers, and solved the crime, you can go back to each page and add special animation features or hints.

**Action**

Design an animation on each location page to show players that they are on the right track.

For instance, on Page2, the desert, an airplane flies across the desert sky. For example:

```plaintext
to fly Airplane is the name of a turtle.
airplane, seth 270 repeat 600
[fd 1] You may want to add sounds, too.
end
```

After answering `question1` correctly, the player goes to Page2 and sees the animation. Add the `fly` procedure to the `correct1` procedure.

```plaintext
to correct1
map
Mexico, clickon
cheer
announce [You are right.
Cacti grow in the desert]
page2
fly You see the airplane on Page2.
end
```

**Hints**

For each page, write a `hint` procedure that uses `announce` to show a hint. For example, here is a hint for Page1:

```plaintext
to hint1
announce [The saguaro cactus is a very large cactus found in deserts in North America.]
end
```

Make a button to run the `hint1` procedure on Page1.

Think of other ways to add details to your game.
CHECK OUT THESE WEB SITES

If your school has Internet access, you can get more information about plants at these World Wide Web sites:

Missouri Botanical Gardens
http://www.mobot.org/MBGnet
Information about flowers and different biomes.

The Butterfly Zone
http://www.butterflies.com

Seeds of Change Garden
http://horizon.nmsu.edu/garden/welcome.html
MICROWORLDS LEARNING

Vocabulary

,  
=  
announce  
answer  
clickoff  
clickon  
forward (fd)  
everyone  
hidetext  
ht  
if  
left (lt)  
pos  
question  
repeat  
right (rt)  
seth  
setpos  
setshape (setsh)  
showtext  
st  
stop  
wait  

Special word: startup

Logo Programming Concepts

Addressing turtles and text boxes  
Controlling text and turtles with commands  
Conditional statements  
Creating games under program control  
Multiple processes  
Programming interactions  
Programming colors  
Superprocedures and subprocedures  
Text boxes as variables  
Writing Procedures
The Living World
Plants Activities
Open a new project. There are two patterns that bees use to convey information, a circle and a figure-8. Start with the circle “dance.” When a turtle makes a circle, it goes forward a little and turns a little. Try: 

```
repeat 36 [fd 1 rt 10]
```

Too small? To make the circle larger, change the input to `fd`.

```
repeat 36 [fd 5 rt 10]
```

Write a circle procedure.

Set the turtle to the bee1 shape. Make the turtle “flap” its wings.

```
setsh "bee2 wait 2
setsh "bee1 wait 2
```

Write a `flap` procedure.

Put `flap` in the instruction line in the turtle’s dialog. Set Do It to Many Times.
Now write a procedure to get the turtle to flap its wings at the same time as it flies in a circle. For example:

```plaintext
to dance1
  clickon ........................................... The bee starts flapping.
  circle ........................................... It moves in a circle.
  clickoff .......................................... The flapping stops.
end
```

Create a button to run `dance1`.

Use the Melody Editor or import music 🎵 to add bee dance music!

Can you get the bee to fly in a figure-8 pattern? (Hint: a figure-8 is two circles put together. In one, the turtle turns right. What does it do in the other?)

Create a new bee dance. What does it tell the other bees?

Watch your pet or any animals that live near you. Do they do “dances” to convey information? Create an animation of one of these dances.
SEED SCATTERER

The wind causes seeds to be scattered randomly. This is an important way of ensuring that plants continue to exist. Create an animation to show how the wind scatters seeds.

Start a new project. Draw a map of part of a city or town. It should look something like this:

Put a park on the left side of your map. In the park, there should be a few flowers. Make sure there are other green grassy areas, buildings, roads.

In the Shapes Center, draw a seed shape. It may look like this:

Set the turtle to this shape. Put the seed on one of the flowers.

Every time you click on the seed, it will turn right a random amount and go forward a random amount. When it lands on grass, it stays there by stamping its shape. When it lands on a road or a building, nothing happens.

Write a scatter procedure that tells the turtle to turn right a random amount and go forward a random amount. Your procedure may look like this:

to scatter
  rt random 180
  fd random 600
end

Open the turtle’s dialog box and on the instruction line type: scatter

Keep Do It set to Once because you want this to happen only once every time you click on the turtle.
How will you know if the seed lands on the grass? It should stamp its shape.

Program the color green (the green you used for the grassy areas) so that when the turtle lands on it, the turtle stamps its shape and then stops. (If you used both shades of green (green and lime), program them both.)

- Open the Drawing Center.
- Select green and click on it again.
- In the space for turtle instruction, type:
  
  \[\text{stamp stopall}\]

**Stopall** stops the seed after it stamps.

Before you try your `scatter` procedure, freeze the background. (This step is very important.) Type:

```
freezebg
```

Click on the seed-turtle. Did it stamp on the grass or land on a building or road?

Move the seed back to the flower and try again. Try several times. Are seeds beginning to appear all over the screen?

To erase all the stamped seeds, type:

```
clean
```

Add a text box to explain how to scatter the seeds.

---

Use **`pos`** to find the starting position of the seed. Write a **reset** procedure that uses **`setpos`** and **`seth`** to move the seed back after running **`scatter`** and to set its heading.

Write a procedure that runs the **reset** and **`scatter`** procedures 20 times. For example:

```
to scatter20
  repeat 20 [reset scatter]
end
```

Put this procedure in a button.

How many seeds appear on the screen? Run your **`scatter20`** procedure a few times and keep track of your data. On Page2 of your project, make a chart showing how many seeds appear each time.
A TREE TALE

By looking at the rings in a tree’s trunk you not only learn the age of the tree but you learn about the tree’s life. Different events effect the size of the rings. Create a series of drawings or animations showing what may have happened in one tree’s life.

Open the Tree project. On the first page you’ll see a log from a tree trunk. Look at the size of the rings and the markings. They tell the tree’s tale.

The following chart will help you understand the events that may have happened:

- Narrow rings—the tree had little light and/or water. This can be caused by over-crowding or drought.
- Wide rings—the tree had a lot of light and water.
- Black Markings—the tree was burned.
- V markings—a branch had been there.
- Rings thicker on one side—the tree does this to keep from falling. Something caused the tree to bend.

Look at the center of the log. The rings are very close together. Go to Page2. Name this page Beginning. Use the drawing tools to show what life was like for the tree when it was young. Add an animation to liven up your picture.
For example, maybe the tree was in a big forest with many huge trees. It received little light and had to compete for water. Show the small tree struggling to grow. Use the sapling shape to start. Copy this shape and edit it to show what the tree looks like as it grows.

Set the turtle to the first sapling shape and name the turtle tree.

Write a grow procedure. For example:

```
to grow
  tree, setsh "sapling
  wait 5
  setsh "sapling2
  wait 5
  setsh "sapling3
  wait 5
  setsh "sapling4
  wait 5
end
```

Go back to Page 1. Put a button with begin as its instruction on the center part of the log. Write a begin procedure to go to the Beginning page and to run the grow procedure.

Now add more pages showing what the tree's life was like. For example, the blackened mark on the log indicates the tree was burned. Maybe there was a raging forest fire. You could show this on the second page of the tree's story.

Each time you add a page, put a new button in the diagram. The button will display the page and start and stop any animation.
Butterflies are very helpful to plants, but in their caterpillar stage, some can be very destructive. The change from the caterpillar stage to the butterfly stage is called metamorphosis. Show a “metamorphosis” with animated words by creating the crawling word c-a-t-e-r-p-i-l-l-a-r that turns into the flying b-u-t-t-e-r-f-l-y.

Open the Metamorphosis project. In the Shapes Center, you will see all the letters of the alphabet. Set turtles to these letter shapes to spell the word caterpillar.

First, create enough turtles so that you’ll have one turtle for each letter in the word caterpillar. Place these turtles in a line in order of their turtle name (t1 first, t2 second, etc.) Write a caterpillar procedure that sets each shape to a letter in the word.

```
to caterpillar
  t1, setsh "c"
t2, setsh "a"
t3, setsh "t"
t4, setsh "e"
t5, setsh "r"
t6, setsh "p"
t7, setsh "i"
t8, setsh "l"
t9, setsh "l"
t10, setsh "a"
t11, setsh "r"
end
```

Try it. Make sure all your letters are in order.

```
c a t e r p i l l a r
```

When each letter is where it belongs, each turtle should have the same y-coordinate. Type:

```
everyone [sety -150]
```

To space all the letters evenly, line them up according to their x coordinates. For example:

```
t1, setx -190
t2, setx -160
t3, setx -130
t4, setx -100
t4, setx -70
...
```

...and so on.

Write a setup procedure that sets all the turtles to the right y-coordinate, places each turtle on its x-coordinate, and sets it to its letter.

```
seth 90
everyone [glide 10 2]
```
Glide tells the turtle to move forward. The first input says how far to move. The second says how fast to move in turtle steps.

Write a bump procedure to get each letter to glide along the ground. Then write a crawl procedure to get each letter to bump three times.

to bump
everyone [glide 10 2]
end
to crawl
repeat 3 [bump]
end

Try crawl.

seth 90
crawl

Now you have a crawling caterpillar. Change it to a flying butterfly.

First, write a butterfly procedure that changes the letters. Whoops! Caterpillar has two more letters than butterfly. Hide the last two turtles.

Butterflies don't crawl; they fly and flutter in the sky. Write a flutter procedure.

You may want to change the turtles' starting position to put them in the sky. Try changing their y-coordinate only.

everyone [sety 0]

Now, write a procedure to:
- Line the caterpillar-turtles up,
- Face the right direction,
- Crawl,
- Change to the letters for “butterfly,”
- Move to a different y-coordinate and flutter.

**Draw or import a background for your letter-turtles.**

Make the caterpillar crawl on the ground. You may have to change your starting position.

Explain the metamorphosis in a text box—the Crawling Caterpillar changes to the Flying Butterfly. How does this change affect the plants around it?

Think of other animals that change
(for example, tadpoles change to frogs.)

On a new page, create new animated words to show the change.
Open the Photosynthesis project. You will see a diagram of a plant.

All these processes are important to photosynthesis:
- Carbon dioxide enters the plant.
- The sun’s energy enters the plant.
- Water, absorbed through the roots, travels up from the soil.
- Food travels down from the leaves, through the plant.
- Oxygen leaves the plant.

Use one turtle to show each of these processes.

Start by showing how carbon dioxide enters a leaf to trigger the chlorophyll.

Place your turtle to the left side of the plant leaf. In the Shapes Center, there are two arrow shapes that point to the right, a long one and a short one. Set the turtle to the long arrow shape.

Try this:
```
repeat 5 [setsh "r.arrow1 wait 2
          setsh "r.arrow2 wait 2]
```
It looks like the arrow is pointing to the leaf, showing how carbon dioxide enters the plant.

Write a \texttt{co2} procedure that shows this process. For example:
\begin{verbatim}
    to co2
        setsh "r.arrow1
        wait 2
        setsh "r.arrow2
        wait 2
    end
\end{verbatim}

In order to differentiate between processes, use different colored arrow shapes. For example, to show oxygen leaving the plant, make a second set of right arrows in a different color. Remember to give these shapes new names (such as \texttt{r.arrow3} and \texttt{r.arrow4}).

Once you've created an animation for each of the processes and tried each one, put them altogether.

Type:
\texttt{everyone [clickon]}

To stop the process, type:
\texttt{everyone [clickoff]}

Write \texttt{start} and \texttt{stop} procedures. Make buttons to start and stop the animation.

Add text boxes as labels to show the names of each part of the plant.

Make a legend to clarify what each turtle represents. For example, red arrows show carbon dioxide, yellow arrows show oxygen, etc…

Click on the turtle to try it out.

Now follow the same steps to show each of the other processes. For each process, remember to:

\begin{itemize}
  \item Create a new turtle to show the process.
  \item Write a procedure for the animation.
  \item Put the procedure name on the instruction line in the dialog box of the turtle and always set Do It to Many Times.
\end{itemize}
Open the Hot Words project. You’ll see this text box.

The stem of a plant grows up, towards the light.
This is called positive phototropism.
The plant's roots will grow down, away from the light.
This is called negative phototropism.

Make the words “up” and “down” hot words. Follow these steps to make a word into a turtle shape.

- Double-space the text to have enough room for any animations.
- Change the color of the words “up” and “down” to red to show they’re hot. (You may want to change the size and the style for all the text.)
- Make the text box transparent and delete the text.
- Delete the text box.
- Open the Drawing Center and pick the .
  Select the word “up.” Copy it with the copy keys or menu item.
- Open the Shapes Center. Click on an empty shape.
  Use the paste keys to paste in a copy of the word.
  Open the Shape Editor and name the shape hotup.
- Open the Drawing Center and pick the .
  On your page, use the from the Drawing Center to erase the word “up” from the text. Move the turtle into position.
In the Shapes Center, there are shapes of a plant as it begins to grow taller. Write a \texttt{growstem} procedure to show the plant as it grows taller, towards the light.

\begin{verbatim}
to growstem
  setsh "stem1
  wait 1
  setsh "stem2
  wait 1

  ... Add more instructions.
end
\end{verbatim}

Write \texttt{growstem} in the instruction line of the up-turtle's dialog box. Remember to include an instruction to set the shape back to the word “up” once the procedure is finished. For example:

\begin{verbatim}
growstem setsh "hotup
\end{verbatim}

Now, click on the word “up” in the text. You should see your animation. (If the shapes change too quickly, change the input to \texttt{wait}.)

Create a second turtle and follow the same steps to make the word “down” a hot word. Write a \texttt{growroot} procedure. Program the down-turtle to run the \texttt{growroot} procedure and set the turtle back to the down shape.
WHAT’S IN A NAME?

When you hear the names of some plants, you get a picture in your mind of what they look like. Think about these names: Organ-pipe cactus, Dutchman’s breeches, Rattlesnake master, Red-stemmed Storksbill, and Fairy Slipper. Make an animation that shows why a plant has been given its name.

Open the Names project. In the Shapes Center, you’ll see shapes of some plants with interesting names.

Start with the organ-pipe cactus. This cactus gets its name because it looks like a set of organ pipes. There are two shapes for this cactus.

Use `setsh` to make the “pipes” look like they are moving up and down.

```
setsh "organ-pipe1" wait 2 setsh "organ-pipe2" wait 2
```

Put these instructions in the instruction line of the turtle’s dialog box. Give the turtle a name, such as cactus. Set Do It to Many Times.

Import organ music or make a tune in the Melody Editor.

Now, write a procedure to show your animation and play the music. For example:

```
to organpipe
  cactus, clickon
  organsong
  clickoff
end
```

Add a button to run the `organpipe` procedure.

Put other plants with interesting names in your project. Program each plant-turtle to run an animation showing why the plant has been given its name. Add music and sounds, too! Write a procedure for each plant and create a button to run each procedure.

Finally, draw a background for your plants.
Create a turtle shape of each plant's name so it will appear before each animation. Give each new shape a name. Get the turtle to switch from the plant shape to the word shape. For example, for the organ-pipe cactus, try:

```
setsh "organword"
```

Edit your organpipe procedure so that the turtle first switches from the cactus shape to the words and then runs the animation and plays the music. Then, edit each of your other plant procedures.

Some plants get their names because of the unusual way they react. Find out why the sensitive plant has its name. Create an animation to show the sensitive plant in action.
Some plants trap insects and eat them! The best known of these plants is the Venus’s flytrap. When a fly lands on one of the plant’s leaves and touches a special “trigger” hair, the leaf closes quickly, trapping the fly. Make an animation showing how a Venus’s flytrap catches insects for food.

Open the Trap project. In the Shapes Center, there are several different sets of plant and insect shapes. You need two turtles—one for the plant and one for the fly. You need three procedures—one to start the fly flying, one to get the plant to eat, and a third to have all the actions happen at the right time.

To begin, set the first turtle to the flytrap1 shape. Name the turtle Venus.

Create another turtle and set it to the fly1 shape. Write a `flying` procedure so that the fly-turtle will fly towards the venus-turtle. The fly-turtle should both flap its wings and move forward.

Write `flying` in the instruction line in the fly-turtle’s dialog box. Name this turtle fly. Select Many Times.
Try it. Make sure the fly-turtle is heading in the right direction.
When the fly reaches the Venus's flytrap, the leaf closes and the fly disappears. Write an **eat** procedure:

```plaintext
to eat
  venus, setsh "flytrap2
  fly, ht
wait 5
venus, setsh "flytrap1
end
```

How will the venus-turtle “know” when the fly is on it? Try this—move the fly-turtle until it is on the venus-turtle. In the Command Center type:

```plaintext
show touching? "fly "venus
true
```

MicroWorlds prints this.

**Touching?** asks if the two turtles—fly and venus—are touching. It reports either true or false.

Move the fly back and type:

```plaintext
waituntil [touching? "fly "venus]
fly, clickoff
```

Click on the fly.

**Waituntil** tells MicroWorlds to wait until whatever is written in brackets is true (the fly and venus are touching) and then do the next instruction (the fly-turtle stops flying).

Write an **action** procedure to start the **flying** procedure and to get the **eat** procedure to start at the right time:

```plaintext
to action
  fly, clickon
  waituntil [touching? "fly "venus]
  clickoff
  wait 10
  venus, eat
end
```

Put **action** in a button. Before clicking on the button, type:

```plaintext
fly, st
```

Move the fly back to its starting position. Click on the button.

---

The pitcher plant is another family of plants that traps insects. Create an animation to show the pitcher plant in action.

---

**Vocabulary**
- **clickoff**
- **clickon**
- **ht**
- **setshape (setsh)**
- **st**
- **touching?**
- **wait**
- **waituntil**

---

Add a text box explaining what is happening. Write a **setup** procedure to move the fly-turtle back to its starting position and get it to show. Put **setup** in a button.

The pitcher plant is another family of plants that traps insects. Create an animation to show the pitcher plant in action.
Do you think there’s life on other planets? If there is, do you think they have food chains and food webs like those we have on Earth? Using what you know about Earth food chains and food webs, imagine what an intergalactic food chain would be like and create an animation to show it.

Open a new project. Create a shape of one of the “links” in the food chain, for example, a plant. Set the turtle to this new shape. Name the turtle plant.

Next, make a shape of a space creature. Create a turtle, set it to this shape, and name it creature1.

Now program the first interaction in the food chain—creature1 moves towards the plant. When creature1 touches the plant, the plant eats it!

In the instruction line of creature1-turtle’s dialog box, type:

```
fd 1 wait 1
```

How will the plant-turtle know when the creature1-turtle touches it? Try this. Move the creature1-turtle until it is on the plant-turtle.

In the Command Center, type:

```
show touching? "creature1" "plant"
```

MicroWorlds prints this.

`Touching?` asks if the two turtles—creature1 and plant—are touching. It reports either “true” or “false.”

Set Do It to Many Times.
Now move the creature1-turtle back and type:

```
waituntil [touching? "creature1" "plant"]
```

Click on the creature1-turtle.

**Waituntil** tells MicroWorlds to wait until whatever is written in brackets is true—creature1 and plant are touching—and then do the next instruction.

Write a procedure to make all the action happen. Use waituntil and touching?

For example:

```
to devour1
  creature1, clickon
  waituntil [touching? "creature1" "plant"]
  clickoff
  ht
  plant, setsize 60
end
```

The creature1-turtle disappears.

The plant grows from its starting size of 40.

Before trying your procedure, find the starting position for the creature1-turtle:

```
creature1, show pos
-130 -20
```

Your numbers will be different.

Try your procedure:

```
devour1
```

Next, write a reset procedure that puts the creature1-turtle back at its starting position, gets it to show, and resets the size of the plant. The starting size for a turtle is 40.

Create 2 buttons: devour1 and reset.

Add more creatures to your food chain. Write devour procedures for each.

Remember to add the information about the new creatures' starting positions to the reset procedure.

Try your devour procedures.

Use what you know about food chains and webs on Earth to figure out what happens to the food chain member at the top of the food chain. How does it contribute to the food chain?

Write a procedure to reset your food chain members and run all of the procedures.
Think of three questions about parts of plants and three about water plants. Each question should have only one answer and they shouldn’t be true or false questions. The easiest question will be worth 100 points, the hardest 300 points.

Open the Trivia project. On Page 1, there are two text boxes with the names of the topics—Plant Parts and Water Plants. There are six turtles lined up under the two topics, each one set to a number shape. Two are set to the 100 shape, two to the 200 shape and two to the 300 shape.

Go to Page 2. There are six text boxes, each with a name (for example, Parts100). Write one question in each text box, the easiest ones in the 100-boxes, the hardest questions in the 300-boxes. For example, write the easiest question about plant parts in the Parts100 box.

Go to Page 1. Try a question. In the Command Center, type:

```
question parts100
```

(Remember to answer the question.)

There is only one right answer for this question, but there may be many wrong answers. So instead of using if, use ifelse. Type, in the Command Center:

```
ifelse answer = "roots [cheer][oops]
Use the answer for your question.
Remember to use " before roots.
```
If else says:
If the first input is true (answer = "roots"), run the first list of instructions.
Or else, if the first input is not true (answer does not equal "roots") run the second list of instructions.

Write a procedure for the first question and answer using ifelse.
For example:

to ask.parts100
question parts100
ifelse answer = "roots [cheer][oops]
end

Open the dialog box for the 100-turtle under the Plant Parts topic.
In the instruction line, type ask.parts100. Keep Do It set to Once.

Try it. Click on the 100-turtle.
Write a procedure for each of your questions. Program the correct turtle to ask each question.

Add a text box on Page1 and name it Points. Keep track of your points by adding points to your score when you answer a question correctly and subtracting points when your answer is wrong.

Keep track of your score automatically. To start, type 0 in the text box named Points.
Try this:
setpoints points + 100

Points reports the number in the text box named Points.
Setpoints changes the contents of the text box named Points. In this case, it says add 100 to the number already in the text box.

Use setpoints and points in your procedures for each question. For example:

to ask.parts100
question parts100
ifelse answer = "roots
[cheer setpoints points + 100]
[oops setpoints points - 100]
end

Edit all your procedures to add or subtract the right number of points.