MicroWorlds Project Builder

Information
for
the
Curious

by

Dr. Robert D. Sweetland

Fall 1996

Introduction

Getting Started

1. Turn on the computer and monitor if needed.
2. Follow the directions the teacher gave you to start MicroWorlds Project Builder.
3. You should see the title page of MicroWorlds Project Builder.
4. You may press ENTER or wait. After a few seconds the title page will disappear and you will see three windows and six pull down menus across the top of the screen.

WELCOME TO MicroWorlds Project Builder.

When you are done using MicroWorlds Project Builder you can exit by selecting **Quit** on the **File** menu and logging off the computer.

For practice you may want to leave MicroWorlds Project Builder and return before you continue.

The three windows from the largest to the smallest are the:

1. project window,
2. command center, and
3. tool palette.

The pull down menus are: File, Edit, Font, Pages, Gadgets, and Help.

You will begin to explore commands by entering them into the command center and seeing the results in the project window. Later you will explore the menu items and the tool palette.

Notes:

Project Window

The turtle is in the **project window** and is heading toward the top of the screen (heading zero).

Look at the **command center**. If there is a blinking vertical line you can type commands. If there is not move the cursor into the box and click the mouse button.

Type **fd50** and press **ENTER**.
The computer will print. *I don't know how to fd50*.

Type **fd**, press the space bar, and type **50**. Then press **ENTER**.
If you are real observant you will notice the turtle has moved up.

The turtle did not draw a line because the pen was not down.

Type **pd fd 50** and press **ENTER**
You should see a line drawn in the **project window**.

When you type a word in the **command center** and press **ENTER** the computer will try to execute the command. If there is not a command with that name it will say: I don't know how to...

Since there is no command **fd50** it gave you the error message.

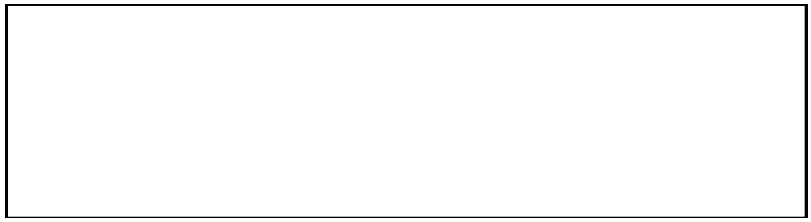
Spaces are very important in Logo.

If you make a typo you can erase the error with the **BACKSPACE** and retype.

For example: type **fd12**, press **BACKSPACE** twice to erase **12**,
press the **SPACE** bar once, and type **12.**

See? It's easy to correct a mistake.

After the **fd 12** Press **ENTER**.
Draw what you see?



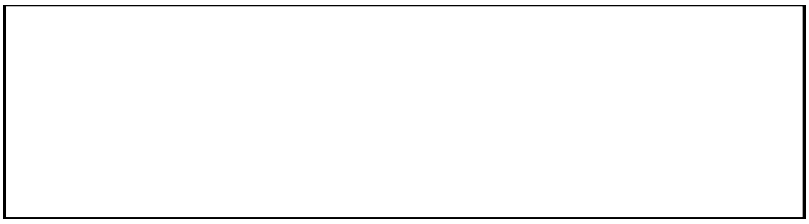
When you want to clear the graphics type **cg**.
Type it now and press **ENTER**.

cg clears the graphics and moves the turtle **HOME** (position 0 0 and heading 0).

Type

fd 40 press **ENTER**
rt 90 press **ENTER**
fd 40 press **ENTER**
rt 90 press **ENTER**
fd 40 press **ENTER**
rt 90 press **ENTER**

In this box draw what you see on the screen.



What do you think **fd** means?

Type **fd** and press **ENTER**
What message was printed on the screen?

What do you think inputs mean?

Some commands in Logo do not need inputs. Like **cg**.
Some commands in Logo need inputs. Like **fd**.
Remember you always need a space between commands and their inputs.

Type **cg** and press **ENTER**.
Type **fd 50** and press **ENTER**.
Type **cg fd 50** all on one line and then press **ENTER**.

You can put more than one command on a line with a space between each.
Whenever you want the commands to start press **ENTER**.
From now on **Enter** will not be printed. You may press it after each command or at the end of a line of commands. It's your choice. **You're the boss**.

Type **bk 75**.

What do you think **bk** means?
Try **cg ht fd 20.**
What happened?

Now try **st fd 20**.
What do you think **ht** means?

What do you think **rt** means?

Type **rt**.
What do you think you could do to give the command inputs?
Did you think of something like **rt 90**?

Try a few.

Brainsqueezer:
Think of a series of commands to make a square.
If you need a hint, here it is.

(HINT)
cg
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90
fd 50 rt 90

Notes:

Making Squares
Write a series of commands to make a square that is larger than 50 by 50.

If **rt** turns the turtle to the right, what command do you think will turn the turtle to the left?

Did you make your square by turning right or left?

Write commands to make a square by turning the turtle the opposite way.

More Commands (PU, PD, clean, home)
Type:
cg
fd 45 pu fd 45 rt 90
fd 45 pd fd 45

The turtle will move without leaving a trail if you pick the **p**en **u**p and move.

The turtle will draw after the command **pd** with a forward or backward move.

Clear the graphics and use **fd**, **rt**, **pu**, and **pd** to draw this shape.

Record the commands here:

Type **cg rt 45 fd 70**
Type **clean**

What happened?
What do you think **clean** means?
Type **cg fd 87 lt 90 fd 62**.
Type **home**.
What does **home** do?

How could you move the turtle home without drawing a line?

Notes:

Review

So far you know:

cg - clears the graphics from the **project window** and positions the turtle with a heading of zero.

ENTER - causes the commands to be executed

ht - hide turtle

st - see turtle

BACKSPACE - erases one space to the left

fd - forward

pu - pen up

bk - backward

pd - pen down

lt - left turn

home - turtle centered heading zero

rt - right turn

clean - clears the graphics from the **project window**

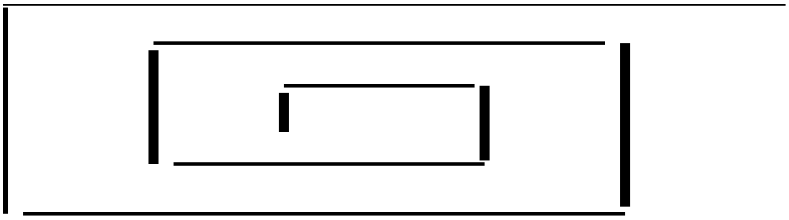
Circle the following commands that need inputs?

CG HT ST FD PU BK PD LT RT CLEAN

Notes:

Practice

Try this.



Hint:

```
cg
fd 10 rt 90 fd 15 rt 90
fd 20 rt 90 fd 25 rt 90
```

Finish this or use your code.

When you finish you may want to save your project and print it.

Save your project from the **File** pull down menu by selecting **save project**.
Type the name in the box and click the save button.
To change directories or disks ask another student or your teacher for help.

You can print the design in the **project window** by making sure the project is displayed on the screen and select from the **File** pull down menu **print page**.
(Printing graphics (pictures) requires more time (sometimes 2 or more minutes) than text, so relax and wait for the system to work.)

Insert your design in this packet.

Repeats for Less Code:

Clear your screen and type this.
fd 40 rt 90
fd 40 rt 90
fd 40 rt 90
fd 40 rt 90

Nice square!

How many sides?

How many corners?

The sides and corners are all alike.
They repeat.

There are many ways to solve a problem
Clear your screen again and type this:

repeat 4[fd 40 rt 90]

You just told turtle to repeat everything inside the brackets four times.
Which way do you like best?

Write a list of commands that will make a square that is 85 turtle steps on each side:

Patterns to Repeat

Clear the screen for more turtle moves.
Type each line and watch what happens.
rt 90
rt 90
rt 90

Will **repeat 3[rt 90]** do the same thing?

How far is all the way around?

Hint

You can use the **command center** with the following symbols to do calculations + addition, - subtraction, * multiplication, / division.
Example Type **show** 90 + 90 + 90 + 90 and press **ENTER** (must have spaces).

What was printed?

How many times would it take to turn the turtle all the way around with **rt 60**?
Hint same only use 60 six times.

Use a **repeat** and write a list of commands to make a hexagon?

How much would each turn be if you wanted the turtle to turn all the way around in five turns?
Hint type **show** 360 / 5 in the **command center** press **ENTER.**

Triangles have three sides and three corners. How much would each turn be if you wanted the turtle to make a triangle?

Hint see hint above and change 5 to 3.

Use **repeat** and write a list of commands to make a triangle?

Write one for a pentagon?

Procedures

Clear your screen.
There are three windows on the screen:
1. **Project window**. To draw graphics with the turtle and write text (You haven't learned how yet).
2. **Command center**.
3. **Tool palette**.

There is another area called the **procedure page**.
Select **procedures** on the **Pages** pulldown menu.
When you do this the turtle will disappear and the title in the window will say *Untitled (Procedures)*.

Click in the procedures window to get a blinking line.
The **procedure page** is where you can teach the turtle your own commands.

Here is how to teach the turtle to make a square.

Get a blinking line in the **procedure page (**put the arrow in the window and click the mouse).

Type **To square** and press **ENTER**.
This is the **title line** for a **procedure** called **square.**

Type the commands to make a square.
repeat 4 [fd 50 rt 90] and Press **ENTER**

type **end** and Press **ENTER**.

Your complete procedure should look like this:

To Square	(Title line must be on a line by itself.)
repeat 4[fd 50 rt 90]	(Commands can be one or more lines.)
end	(**End** must be on a line by itself.)
|	(This line has the cursor.)

If not use the arrow keys, or mouse, to move the cursor. Delete all mistakes (**backspace** or **delete** keys), and type the corrections. If you have a space in front of the T in **To** you will get an error message.
When the procedure is like the sample you are ready to try it.

Use the **Pages** pull down menu and select **Page1**.

Move to the **command center** (click on its window)

Type **Square** and press **ENTER**. Surprised?!
The computer, or turtle, will use the list of commands whenever you type **Square**.

Every procedure has three parts. The **title**, a **list of commands**, and an **end**.
The **title** is always one line.
It starts with the word **To** and is followed by the name of the procedure.
The **list of commands** in the procedure can be many lines long.
The **end** must be on a line by itself.

The following is one way to use the **Square** procedure to make squares.

Slide the mouse until the arrow is on the turtle, click and hold the button down, slide the mouse on the mousepad, and the turtle will move. This is called dragging and you are dragging the turtle. You can drag the turtle to any spot on the screen and release the mouse button.

Then click in the command center at the end of the line with **square** and press **ENTER**.

The turtle should draw another square. Repeat the procedure to make several squares in different places.

Notes:

Super Procedures

Let's use the **square** procedure in another procedure to make squares.

Go to the procedure page (Pages, procedures) or click on the page icon with **to** on it in the **tool pallete**.

Move the cursor (arrow keys or mouse) below the **square** procedure that is already there.

If the **square** procedure is not there you will have to retype it.

Type the following procedure:
To squares
repeat 4[square lt 90]
End

Change from the procedure page to page 1.
Move to the **command center** and type **cg squares**.

You have now written two procedures and used one procedure (**square**) in the list of commands of another procedure (**squares**). You can do this as often as you like.

Let's use it to make a flag.

Go to the **procedures page**.

Start with the following procedure:
To pole
fd 90
end

Move to **page 1** and the command center to try the procedure.
Type **cg pole** and press **ENTER**.

To raise the flag, use the **pole** and **square** procedure in the **flag** procedure.
Move to the **procedure page** and add the following procedure:

To flag
pole
rt 90
square
lt 90
bk 60

end

Try the FLAG procedure (**cg flag**).
After you're done admiring your work go to the **procedure page** and write the following procedure to make six flags in a circle.

To flag6
repeat 6[flag rt 60]
End

Is the flag too big?

Go to the **procedure page** and change the size of **fd** (30) in the **square** procedure.

You can put **cg** in the **flag6** procedure before the **repeat** line so you do not have to type it each time.

If you haven't saved the project do so now (**File, Save**).

Print the picture.
Make sure the page with the flags are showing and select **file, print page**.

To print the procedures make sure the procedure page is showing and select **file, print page**.

Put both copies in this packet.

BrainBuster:
Can you make one change so the **flag6** procedure will make the following?

Super Procedure Narratives, and Diagrams:

A super procedure is a procedure that calls on other procedures to make a whole picture, design, scene, or do anyother thing that a computer can do. When the name of the super procedure is entered in the **command center**. The computer takes over and runs the list of commands the way the programmer organized them in procedures. The **flag6** procedure is a super procedure. When you typed **flag6** and pressed **enter** the computer took over and ran the list of commands in several procedures to make six flags.

Narrative to explain control of super procedure **flag6**:

When you type **flag6** in the command center the Logo language starts to do the list of commands in the **flag6** procedure. It starts with the **cg** command and clears the graphics screen.

The next commands are in the repeat statement where the **flag** procedure and a **rt 60** will be repeated six times. The first command in the repeat is to call the **flag** procedure. So the computer calls the **flag** procedure. When the **flag** procedure is called the first line in that procedure is **pole,** so the **pole** procedure is called.

Nothing has been drawn yet. The first command in the **pole** procedure is **fd 90** so now a line of 90 turtle steps is made. Since this is the only command in the **pole** procedure it ends.

When **pole** ends control returns to the **flag** procedure from where it left, before the **square**. Control then goes to the **square** procedure where it makes a square by repeating a **foward** 50 and **right** 90 four times. When the **square** procedure ends the **flag** procedure continues with the command **lt 90** and **bk 60**. Next the **flag** procedure ends and control goes back to the **flag6** procedure.

The next command to be executed is a **rt 60**. All of this makes one flag and a little more. The **repeat** command will then repeat the above process (except for the **cg**) five more times. Then the **flag6** procedure will end and control will be given to the command center.

Notes:

The diagram below shows the order in which the lines are drawn by the **flag6** super procedure.

The diagram below shows the control of **flag6** super procedure in a diagram.

Here is a list of procedures. Can you tell which is the super procedure? Figure out how the super procedure works and write a narative to explain how it works, diagram to show order of the lines drawn, or a diagram to show the control of the super procedure.

```
To circle
repeat 36[fd 1 rt 10]
end

to plus
rt 90 fd 10 bk 5 lt 90 fd 5 bk 10 fd 5 lt 90 fd 5 rt 90
end

to move
pu fd 20 rt 90 fd 20 lt 90 pd
end

to getset
pu
lt 90 fd 100 lt 90 fd 100 rt 180
pd
end

to circleplus10
cg
getset
repeat 10[circle plus move]
end
```

Notes:

Procedures for Curves

Open a new project (**File, new project**) and name it **Sunshine**.
Let's create Sunshine!
Type in these three procedures.

```
To piece
repeat 10[fd 30 bk 30 rt 1]
end

To ray
repeat 2[fd 60 bk 60 rt 1]
end

To sunshine
pd ht repeat 30[piece ray]
end
```

The **sunshine** procedure would be considered a super procedure if all you wanted to make was a sun.

For a circular design try these two procedures.

```
To ring
repeat 20[fd 7 rt 18]
end

To ringright
repeat 10[ring rt 36]
end
```

Which procedure uses the other procedure to make a design?

Let's suppose you wanted a **ringleft** procedure.
Highlight the entire **ringright** procedure. Then use the **Edit** pulldown menu **Copy**, move the cursor below the **Ringright** procedure and use **Edit**, **Paste**. You could also use the short cut keys **ALT C** (copy) and **ALT V** (paste).

Delete **right** from the title and type **left**.
Then delete **r** from **rt** and type **l**.

Go to the flipside and try **ringleft**.
If all went well you should have the same design only drawn to the left.
You can use the edit functions to change procedures without typing.

Tool Palette

The **tool palette** has eight choices: From top left they are selection arrow, shapes center, drawing center, procedure page, turtles, text boxes, buttons, and sliders.

Arrow and Shapes Center
Clicking on the **arrow** will close a tool palette window.
Click on the **shapes center** (moon) and window will display choices of shapes.
Click on the **arrow** again and it will close the window.

Click on the **shapes center** (moon) again and open the shapes window.
To change the shape of the turtle click on a **shape** (tree), and slide the arrow into the graphics window, as you do it will change to a pointing hand. Slide the hand onto the turtle and click. The turtle should change to the shape you selected. To change the turtle back click on the turtle shape in the shape center.

The **shapes center** tools are:
Arrow to select turtles or turn off a tool.
Turtle to turn the shape back to the turtle.
Scissors to cut turtles.
Stamp to stamp shapes.
Magnifying glass + to grow shapes.
Demagnifying glass - to shrink shapes.

To **stamp** a shape select a shape (house) with the **arrow** by clicking on it. The turtle will change to the house. Click on the **stamper** tool and it will turn to a stamp. Slide it onto the turtle on page 1 and click. It will turn into a house. Then click on the **arrow tool** and use it to drag the house (turtle) away and leave the stamped shape. Next click on the **turtle tool**. It will change from the to an arrow. Then click on the house, that is the turtle, on page 1. This will change the house back to the turtle.

Magnifying glass + demagnifying glass - Select a shape and change the turtle to that shape (tree). Click on the **magnifying glass** and slide the magnifying glass onto the tree shape and click on it a few times. The tree will grow. Then stamp the tree shape (stamp tool), drag the turtle to another location (arrow tool) and switch the turtle to another shape (click on another shape and click the arrow on the turtle). You should notice that the shape is as large as the stamped shape. You can click on the **demagnifying glass -** and click on the turtle shape to shrink the shape. You need to remember where the turtle is so you don't loose the turtle in the stamped shapes.

Scissors the scissors can be used to cut a turtle from the page. If you cut a turtle it will be gone. Cut the last one and there will be no more. If you cut all the turtles

and decide you need a turtle, or more, click on the turtle button in the **tool palette** to hatch some.

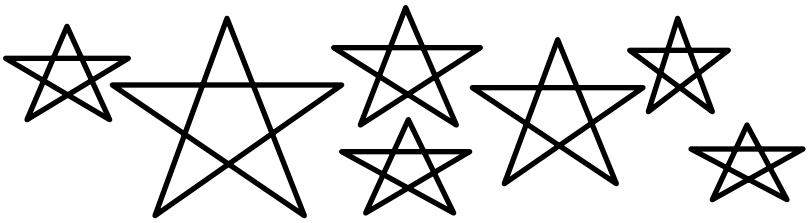
Drawing Center is the paint brush. These tools are the standard tools in most drawing programs. Be adventurous.

Procedure Page is the page picture with “to”. When it is clicked it will switch you back and forth from the procedure page to the work page.

Turtle Button looks like a turtle hatching from an egg. When you click on it and click on the work page it will put another turtle on the page. Turtles can be cut from the shapes page with the scissors. New turtles are numbered according to the order in which they are hatched (t1, t2, t3...). If you put the arrow on a turtle and click the left mouse button its number will be shown.

You can talk to a particular turtle with the following code pattern:
t1, setsh “house fd 100 t2, setsh 12 fd 2.

Buttons, Sliders, and Variables



Open a **new project** (**file, new project**) and name it star.
Stars are made of 5 lines and 5 turns..... a natural for a **repeat**.
What do you get with **repeat 5[fd 30 rt 72]**?

A pentagon is not a star.
For the star's turn, does turtle turn more or less than the pentagon?

Enter the following procedure.
to star :turn
cg pd ht
repeat 5[fd 30 rt :turn]
end

Let’s see what the colon turn (**:turn**) does in the procedure.
What happens after you enter the procedure and try **star**?

Try different inputs (**star 72**, **star 23, star 100**).

Let's see how to use a **button** and a **slider** to experiment with the **star** procedure.

Click on the **button** tool in the **tool palette** (second box from the bottom right)
Move the pointer into the **project window**.
When you do you will see the arrow tool change to a hand with a **button**.
Position it where you want the button (on the left or right side of the window) and click the mouse button.
A button box will pop-up on the screen. In the **instruction** box write **star turn** and click **OK**.

Click on the button. You should get an error message. That's okay you need to add a **slider** for the input.

Click on the **slider** tool in the **tool palette** (box on the bottom right).
Move the pointer into the **project window**.
The arrow tool will change to a hand with a **slider**.
Position it where you want the slider (left or right side of the window) and click the mouse button.

A **slider** box will pop-up on the screen.
Put **turn** into the **name** box.
Put an **X** in the show name box.
Type **125** in the Minimum box.
Type **150** in the Maximum box.
Click **OK**.

Click on the **star button**.
You should get five lines drawn on the project page.
You can drag the turtle off the lines and see what it looks like.
You can change the value for the input to star by dragging the slider or clicking on the right or left of the slider.

Experiment until you find the best input to make a star.

What is it?

Notes:

Animation

There are many ways to animate the turtle. Here is one procedure. Open a new project. Enter these procedures and give it a try:

To Fly
repeat 10 setsh “bird1 fd 2 setsh “bird2 fd 2]
end

Birds usually don’t fly straight up. See how you can change it.

Notes: (see the project sqdance)

Text Boxes and Conditionals

Open a new project.
Enter these procedures:

```
to rect :s1 :s2
repeat 2[ fd :s1 rt 90 fd :s2 rt 90]
end

To grow :size
if :size > 100 [stop]
rect :size :size * 2
grow :size + 10
end
```

Try to write a procedure that will use an input to make different sizes of squares.

You can use a **text box** to write notes of what you did.

Click on the **text box** button in the **tools palette** (bottom row with the letter A). When you move the pointer to the project window it will change to a pen. Move it to the top left hand corner of where you want the text box. Click and drag to cover the area where you want text.
Click on the box and you can type whatever you want into the box.

You can change the font, font size, styles, and colors. You can also change the size of the text box by dragging its handles.

Print a picture of the square(s) with your name and size of input(s), a list of the procedure, and insert it in this packet.

"Species" Sample

(need microworld "species")

Three scientists, who worked separately, discovered three strange new life forms that live on the ocean's floor in three different geographical regions. Each scientist claims to have found a **new species**. Each named their species: D, T, and S after Dr. Denau, Dr. Trebor, and Dr. Setaoc who located each. Organisms from each geographical region (species?) begin life as sporelike cells and at regular intervals undergo a metamorphosis into a new form. Organisms from each geographical region (species?) and their life stages change as they go through different time periods. Computer models of each organisms from the three geographical regions (species?) can be viewed on the computer by typing D 1, T 4, S 3, and so forth. If **D** 2 is entered the computer program will model the organisms collected by Dr. **D**enau for its second time period of life. If **S** 3 is entered the computer program will model the organisms collected by Dr. **S**etaoc for its third time period of life. If **T** 6 is entered the computer program will model the organisms collected by Dr. **T**rebor for its sixth time period of life.

These distinguished scientists have had a dispute over who was the first to discover this species. Unfortunately this has caused no coordination of their individual studies and each to claim the organisms from each geographical region are of a different species. Your task is to see if the orgnisms are similar enough to be one species or if they are unique enough for each to be considered a separate species. Examine the data created by the computer models, make a decision, and support it with evidence.

1. Open the microworld "species".
2. Decide what time period (era) you want to model and set the slider for that era, then click on the species button (D, S, or T).
3. Experiment with various combinations and collect evidence to create and support a theory.

Feel free to make a copy of this Microworld to add to your resource file.

Creating a Sample Microworld “TurtleWorld”

Open a new page.
Create a button and call it erase.
Enter the following procedure on the procedure page for it.
to erase
cg pu ht
bk 100 pd
end

Create another button and call it forward 5.
You might want to position this button so that it and four others will be on the opposite part of the page. This will help so the erase button does not get pushed accidently.

Copy the forward 5 button four times, so that all five buttons will be the same size.
Click on the arrow tool on the tool palette.
Use the arrow tool to select the forward 5 button (slide the arrow to a corner of the button, click and drag past the opposite corner and release the mouse button). There should be a square at each corner.
Go to edit and click on copy.
Go to edit again and click on paste.
Repeat the paste three more times.
Label each button with one of the following:
forward 10, forward 25, right 30, right 90.

Congratulations you have created your first MicroWorld “TurtleWorld”.

Explore.

Notes:

Brain Buster

Create a microworld that creates different sizes of squares, SquareWorld.

Hint: You could use a button called erase and another called square length. A slider called length. And a procedure called square with a variable length.

Open a New project and name it **PI**.

Create a button and name it **circle circumference** (if you forget how look back in the packet).

Create a button and name it **eraseall**.

Create a button and name it **mark diameter**.

Create a button and name it **erasediameter**.

Create a slider and name it **circumference** with the maximum = 500 and the minimum = 100.

Create a slider and name it diameter with the maximum = 200 and minimum = 50

Write the following procedures.

```
To circle :circumference
repeat 360[fd 1 rt (360 / :circumference)]
rt (90 + (360 - ((360 / :circumference) * :circumference))
end

To eraseall
cg ht pd
end

To mark :diameter
fd :diameter
make “lastdia :diameter
end

To erasediameter
pe bk :lastdia pd
end
```

Ready to explore? If you would like you may use the data sheet on the next page to record your data.

PiWorld Data Sheet

Collect the following data for each circle.

How many turtle steps around the circle (circumference)?

How many turtle steps across the center of the circle (diameter)?

CIRCLE 1: DATA

Circumference:

Diameter:

How do they compare?

CIRCLE 2: DATA

Circumference:

Diameter:

How do they compare?

CIRCLE 3: DATA

Circumference:

Diameter:

How do they compare?

CIRCLE 4: DATA

Circumference:

Diameter:

How do they compare?

When you are finished how could you combine your results with other results from the class in a chart or graph?

One way would be to have several groups collect data for the same circumferences record the data and calculate the group mean.

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Circumference 360						
Diameter 360						
Relationship						
Circumference 180						
Diameter 180						
Relationship						
Circumference 120						
Diameter 120						
Relationship						
Circumference 90						
Diameter 90						
Relationship						

	Class average
Circumference 360	
Diameter 360	
Relationship	
Circumference 180	
Diameter 180	
Relationship	
Circumference 120	
Diameter 120	
Relationship	
Circumference 90	
Diameter 90	
Relationship	

Another way would be to have each person plot their data (circumference and the diameter) on a graph.

MicroWorlds Primitives by Category

Graphics

back (bk)	bg	cg	clean
color	colorunder	distance	fill
forward (fd)	freeze	glide	heading
home	ht	left (lt)	newturtle
pd	pe	pensize	pos
pu	restore	right (rt)	setbg
setc	seth	setpensize	setpos
setsh	setsize	setx	sety
shape	size	snapshape	st
stamp	towards	turtletype	unfreeze
who	xcor	ycor	

Objects

ask	get	hideframe	newbutton
newslider	newtext	remove	set
showframe	talkto (tto)	turtlesown	

Text Editing

bottom	cb	cd	cf
cleartext (ct)	clipboard	copy	cu
cut	delete	eol	eot?
fontsize	found?	hidetext	insert
paste	print (pr)	search	select
setfont	setfontsize	setstyle	settc
show	showtext	snaptext	sol
tc	textcount	textitem	textpick
textwho	top	unselect	unsnaptext

Words and Lists

ascii	butfirst (bf)	butlast (bl)	char
count	empty?	equal?	first
fput	identical?	item	last
list	list?	lput	member?
number?	parse	pick	sentence (se)
word	word?		

Screen Management

cc	getpage	newpage	newprojectsize

Disk Access

chdir	currentdir	erfile	getproject
loadpict	loadshapes	loadtext	lock
merge	pagelist	pictlist	projectlist
savepict	saveproject	savetext	textlist
unlock			

Flow of Control and Logic

and	cancel	carefully	dolist
done?	dotimes	errormessage	forever
if	ifelse	launch	not
or	output (op)	repeat	run
stop	stopall	stopme	waituntil

Workspace

recycle	space

Assigning

clearnames	let	local	make
name	name?	shownames	thing

Math

+	-	*	/
=	>	<	abs
arctan	cos	difference	greater?
int	less?	minus	product
quotient	random	remainder	rerandom
round	sin	sqrt	sum
tan			

Input/Output

announce	answer	key?	mousepos
printtext	question	readchar	

Time

resett	timer	wait

Sound

note	rest	setinstrument	soundlist

Special

the name of a melody
the name of a sound
the name of a page
the name of a text box
set combined with the name of a text box
the name of a slider
set combined with the name of a slider
the name of a turtle variable
set combined with the name of the turtle variable

MicroWorlds Primitives in Alphabetical Order

abs *number* Reports the absolute value.
and *true/false 1* true/false 2 (and true/false 1 true/false 2 true/false 3 ...).
announce *word/list* Displays the message in a message box.
answer Reports the contents of the last answer typed in the question dialog box. Reports it as a long word. **Parse** can be used to change it to a list of words.
arctan Reports the arc tangent -90 through 90.
ascii *char* Reports the ASCII number for the character.
ask *who nstruction-list* Gives temporary instructions to who (turtle(s), text boxes...).
back (**bk**) *number* Moves turtle back number.
bg Reports the background color by number.
bottom Moves the cursor to the bottom of the text.
butfirst (**bf**) *word/list* Reports all but the first element of the word or list.
butlast (**bl**) *word/list* Reports all but the last element of the word or list.
cancel *instruction-list* Stops the process given as input by **launch** or **forever**.
cb Cursor back one character.
cc Clears all text from the command center.
cd Cursor down one line.
cf Moves the cursor forward one character.
cg Clear graphics, sends turtles to home, clears all text printed with the label command.
char *number* Reports the text character with the ASCII code of number.
chdir *pathname* Stands for change directory.
clean Erases all graphics, does not affect the turtles or text.
clearnames *word/list* Clears all global variables memory.
cleartext (ct) Clears the text in the current text box.
clipboard Reports the contents of the text buffer if empty reports the empty word
color Reports the number representing the color of the turtle
colorunder Reports the number representing the color under the center of the turtle
copy Puts a copy of the selected text into the clipboard
cos *degrees* Reports the cosine for degrees.
count *word/list* Reports the number of elements in the word or list.
cu Cursor up one line.
currentdir Reports the name of the current directory.
cut Deletes selected text and puts a copy on the clipboard.

delete Deletes the character to the right of the insertion point in the current text box.
difference *number1 number2* Reports the result of subtracting number 2 from number 1.
distance *turtle-name* Reports the distance between the current turtle and the turtle indicated.
dolist *range instruction-list* Runs the instruction list for each item in a list. The first input, range, is a list with a temporary variable name and a list of items. The second input is a list of instructions that uses the variable name included in the firest input. In the following example, the instruction remove :I is run for each item of the first list. "I" successively takes the value t1, t2, and t3. dolist [I [t1 t2 t3]] [remove I].
done? *instruction-list* Reports true if the process indicated is complete. The process must have been launched using launce or forever. Input must be an exact copy of the instruction list.
dotimes range instruction-list Runs the instruction list for each value specified in the range. The first input is a list with a temporary variable name and a maximum number. The second input is a list of instructions that uses the variable name included in the first input. dotimes [I 8] [setbg :I wait 5] sets the background color for each value of I, from 0-7. dotimes [I 10] [show :I] writes 0, 1, ...9 in the command center.
empty? *word/list* Reports **true** if word/list is an empty word or list, otherwise **false.**
end End of a procedure definition.
eol Moves the cursor on the page to the end of the current line.
eot? Stands for end of text. Reports true if the cursor, in the current text box, is at the end of the text.
equal? *word/list 1 word/list 2* Reports **true** if word/list 1 or word/list 2 are the same otherwise **false** words and lists are not equal.
erfile *name/pathname* Erases the file of any type if it is not locked.
errormessage Reports the last error message trapped by **carefully**. If it reports an empty word, then there was no error.

Everyone *instruction-list* Makes all turtles on the current page run the instruction, one after another.

false Used as input and output to conditionals.
fill Fills a shape on the screen with the turtle's color. The turtle must be inside the shape.
first *word/list* Reports the first element in word/list.
fontsize Reports the font size used at the insertion point in the current text box. If the text currently selected has more than one font an empty list is reported.
forever *instruction-list* Runs the instruction list repeatedly as an independent parallel process. Use cancel menu item, the stopall menu item. or **ctrl - break** to stop the processes.
forward (**fd**) number Moves the turtle forward the distance of number.
found? Used after **search** or **replace** to report if the desired word was found (**true**) otherwise **false.**
fput *word/list list* Forns a new list by putting the word/list into the list in the first position.
freeze *word-or-list-of-page-elements* Freezes a text box, button, turtle, or slider so that it cannot be moved, resized, or removes with the mouse. The input is the name of a text box, turtle, button, or slider; a list containing many names; or the name of a page.

get *object property* Reports a property of an object in the current project. The first input is the name of a turtle, text box, slider, button, color, or page. The second input is a property name.
The following are properties of each object.
Turtles: rule, on?, own
Pages: turtles, texts, buttons, sliders
Sliders: pos, showname?, limits, value
Buttons: pos, size, rule, on?
Text: pos, size, showframe?, showname?, text
Colors: turtlerule, turtlemode, mouseclick
getpage *pagename* Displays pagename.
getproject *project-name* Saves the current project (if not locked) and gets the project name in the current directory.
glide *distance speed* Makes the turtle glide over the distance indicated at the speed indicated. Max speed is 99.
greater? *number1 number2* Reports true if the first number is greater than the second number.

heading Reports the turtle's heading in degrees 0-359.9999.
hideframe Hides the fram of the current text box.
hidetext Hides the current text box. You cannot type in a hidden text box, but primitives such as **print**, **insert**, and **ct** still work.
home Moves each turtle to its home position and sets heading to zero.
ht Hides the turtle.

identical? *word/list 1 wordlist 2* Reports **true** if the two words or lists are equal numbers, identical words, or identical lists otherwise **false**. Case sensitive.
if *true/false instruction-list* If the first input is **true** the instruction list is run, if it is **false** it is not.
ifelse *true/false instruction-list 1 instruction 2* The first input must be a condition that tests **true** or **false** If **true** listtorun 1 is run if **false** listtorun 2 is run.
insert *word/list* (**insert** *word/list1 word/list2 ...*) Inserts a word or list in the page at the cursor position without a carriage return.
int *number* Reports the integer portion of number.
item *number word/list* Reports the element number of a word or list.

key? Reports **true** if a key has been pressed otherwise **false** the character will remain in the character buffer until **readchar**, or **key?.**

last *word/list* Reports the last component of the word or list.
launch *instruction-list* Runs the input as an independent process. If **launch** is used in the command center the cursor reappears as soon as the process is launched. Use **cancel** or **ctrl-break** to stop the process.
left (**lt**) *degrees* Turns the turtle(s) degrees left.
less? *number1 number2* Reports true if the first number is less than the second.

let *list-of-names-and-values* Creates one or many temporary variables. The variables will exist only while the procedure containing the **let** instruction and procedures called by this procedure are running. The input is a list of paired variable names and values. Example of code in procedure: **let** [d 10 s 5 r 4] **fd** :d **rt** : 4.

list *word/list 1 word/list 2* (**list** *word/list 1 word/list 2 word/list 3 ...*) Reports one list made from inputs. Must be in parentheses if more than two inputs.

list? *word/list* Reports **true** if word/list is a list otherwise **false.**

loadpic *name/pathname* Loads a graphic PCX file onto the current page. 640 x 480 mode with 256 colors or less.

loadshapes *name/pathname* Loads the shapes found in the designated project. These will replace all shapes in the current project.

loadtext *name/pathname* Loads a text file into the current text box or on the procedure page.

loadtext *name/pathname* Loads an ASCII (text) file onto the current page.

local *word-or-list* Makes the variable(s) local to the procedure.

lock *name/pathname* Locks the file.

lput *word/list list* Reports a new list with word/list at the end of list.

make *word word/list* Creates a variable name and stores the value of word/list.

member? *word/list-1 word/list-2* Reports **true** if word/list 1 is an element of word/list 2 otherwise **false.**

merge *pathname word-or-list-of-types* Copies pages, melodies, procedures, or sounds from another project into the current project. The first input must be the name of a project in the current directory or a fullpathname. this is the project you want to cpy from. To copy all the objects of one type, the second input must be sounds, procedures, elodies, orpages. If the input is pages, they must be in the same project size s the current project.

minus *number* Reports the additive inverse of number.

mousepos Reports the page coordinates of the current mouse position.

name *word/list word* Creates a variable name and assigns it the value of word/list The variable keeps its value as long as it is not cleared.

name? *word* Reports **true** if word is a name of a variable otherwise **false.**

newbutton *name [x y] instruction-list* Creates a new button with the name and instruction specified, at the position [x y] indicated. Name limit 32 characters. Position is top left corner of button. Created in the once mode. Sized to fit the instruction-list.

namepage (**np**) *pagename* Names the page showing.

newpage Saves the page that is showing and displays a new page.

newprojectsize *word-or-list* Sets a new size for the following new projects. Input can be the word **standard** or **large** or a list of two numbers for width and height in turtle steps. There must be a completely empty project on the screen before using **newprojectsize**.

Newslider *name [x y] [min max current]* Creates a new slider using the specified name at the position indicated. The position is the center of the slider. The last input is a list of three numbers representing the minimum, maximum, and the current value of the slider. -9999 and 9999.

newtext *name [x y] [xsize ysize]* Creates a new text box using the name and size specified at the position x y indicated. The postion is the top left corner of the box. The maximum xsize and ysize is the size of the page in the project.

newturtle *name* Creates a new turtle with the name indicated. appears at position 0 0 and is hidden.

not *true/false* Reports the logical inverse of its input **true** or **false**

note *number-or-list duration* Plays a note using the current instrument. The firt note must be a MIDI note number, or a listof numbers (a chord only available with a sound card). The second number is the duration in tenths of a second. Middle C is 60. Maximum note 127 and maximum duration is 255.

A	**A#**	**B**	**C**	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**
				1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30	31	32

33	34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92
93	94	95	96	97	98	99	1009	101	102	103	104
105	106	107	108	109	110	111	112	113	114	115	116
117	118	119	120	121	123	124	125	126	127		

number? *word/list* Reports **true** if word/list is a number otherwise **false. ONLY FOR NUMBERS BETWEEN 999999999999 - -999999999999.**

or *true/false 1 true/false 2* (**or** *true/false 1 true/false 2 true/false 3*...) Reports **true** if any of the inputs are **true** otherwise **false.**
output (**op**) *word/list* Used only within a procedure. Stops the procedure and outputs word/list. Output makes a procedure act like a reporter.

pagelist Reports a list of all the names of all the pages in the current project.
parse *word* Converts a string of characters into a list. Also changes long words into lists of words. Text boxes report/return long words.
paste Inserts the text in the clipboard buffer into the page at the cursor position.
pd Pendown.
pe *pen erase* Erases any line it crosses.
pensize Reports the number representing the size of the current turtle. Maximum 100.
pick *word-or-list* Reports an element chosen randomly from the word or the list. Pick from a word reports a character. Picking from a list reports an element of the list (word or list).
pictlist Reports a list containing the names of picture files in the current directory. PCX type files.
pos Reports the coordinated of the turtle's position as a list [x y].
print (pr) word/list Prints a word or list in the current text box followed by a carriage return.
printtext Prints out, on the printer, the contents of the current text box or the procedures page, depending on what is currently showing. Text format is not used for printing.
product number1 number2 Reports the result of multiplication. If more than two inputs are used they must be enclosed in parentheses.
projectlist Reports a list containing the names of MicroWorlds projects in the current directory.
pu When the turtle moves it will not draw or erase.

question *word/list* Opens a dialog box displaying the question and an area to type the answer. Answer can then be used to report what was typed in the dialog box as the answer to the question.
quotient *number1 number2* Reports the result of dividing number1 by number2.

random *number* Reports a random integer less than number. Maximum 9999.
readchar Reads a character from the keyboard and reports it. If there is none then it waits until there is. You must click outsie of a text box, the command center, or procedures page for **readchar** to recognize the charater typed.
recycle Garbage collection frees unused memory.
remainder *number 1 number 2* Reports the remainder of number 1 divided by number 2 zero is a remainder.
remove *name* Deletes the named button, turtle, slider, text box, page, sound, melody, or turtle variable created with **turtlesown**. If the named object is not on the current page, MicroWorlds will look for it on other pages. If the input to **remove** is a procedure, the procedures page in the current project will be cleared. You cannot remove the only page of a project. In this case, the page will be cleared.
repeat *number listtorun* Runs the listtorun number of times.
rerandom Reproduces the same sequence of numbers generated by **random**.

resett Resets the timer to zero when MicroWorlds is started.
rest *duration* Inserts a rest in a sequence of notes. In tenths of a second with a maximum of 255.
restore Restores the background to the way it was the last time a snapshot command was issued. Everything else is not changed.
right (**rt**) *degrees* Turns the turtle right degrees.
round *number* Reports the input rounded to the nearest integer.
run *listtorun* Runs listtorun. List can be in a text box. **run text1**.

savepic *name/pathname* Saves the background of the current page as a PCX file. The turtles, button, text boxes, and sliders are not part of the background. Stamped images, stamped text, and text typed with **turtletype** are.
saveproject Saves the current project without closing it. The project must have a name in order to work.
savetext name/pathname Saves the text in the current text box or on the procedures page as a file with .TXT. Formatting (font, style, color) will not be saved. Text can be loaded with **leadtext**.
savetext name/pathname saves the text on the current side of the page as a ASCII file
search word Looks for the first occurrence of word in the current text box and highlights it. The search starts from the cursor if word is found then it is selected must **unselect**, will find word if in part of word.
select Starts the select mode if it is followed by **cf**, **cb**, **cu**, **cd**, **sol**, **eol** a block of text will be selected if is followed by **search**, **replace**, **print**, or **unselect** the select is turned off.
sentence (se) *word/list 1 word/list 2* (**sentence** *word/list 1 word/list 2 word/list 3* ...) Reports a list made from the inputs. If more than two inputs must enclose all in parentheses.
set *object property-value* Sets a property for an object to the specified value. First input is a name of a turtle, text box, slider, button, color, or page. The second input is a property name and the lst is the value. See **get** for the list of objects and properties.
setbg *name-number* Sets the backgkround color to the color number represents.) or “white is the original background color.
setc *name-number* Sets the color of the turtle and its pencolor to the color represented by the number. Original color is “black or 9.
setfont *word* Sets the font of the selected text in the current text box. If no text is selected the cursor uses that font for typing.
setfontsize number Sets the font size for the current text box. If no text is selected it sets the cursor to use that size for typing. Input must be 10, 12, 14, 20, 24, 28, or 48.
seth *degrees* Sets the heading of the turtle to degrees.
setinstrument *name* Sets the instrument to play with note. Name must be one of the following words: piano, bottle, harp, xylophone, violin, or organ (1...6).
setpensize *number* Sets the pen size of the turtle. Original = 1 and maximum is 100.
setpos *[x y]* Moves the turtle or turtles to the given screen position
setsh *number* Sets the shape of the turtle to the shape represented by the number
setsize *number* Sets the size of the turtle. Original is 40. Range 5-150. Multiples of 40 are suggested.
setstyle *word/list* Sets the font style (plain, bold, italic, underline) in the current text box.
settc *name/number* Set text color in current text box. Can use names or numbers.
setx *number* Sets the x coordinate of the turtle to the number The y coordinate is not changed
sety *number* Sets the y coordinate of the turtle to the number The x coordinate is not changed
shape Reports the turtle's shape name or number of the current turtle.
show *word/list* (**show** word/list 1 word/list 2 ....) Prints the word or list in the command center A list is printed without its outer brackets.
showframe Shows the frame of the current text box.
shownames Prints all the variables in memory and their values in the command center.
showtext Makes the current text box visible.
sin *degrees* Reports the sine of degrees.
size Reports the size of the current turtle.
snapshape Copies the background behind the turtle into the current shape of the turtle. Will not work with the original shape of the turtle. E.g. **setsh** 23 **snapshape**.
snapshot Takes a snapshot of the background. The next restore command will restore the background to the snapshot.
snaptext *text-box-name* Makes the text box transparent, shrink the box to the size of the text.
sol Moves the cursor to the start of the current text box line.
soundlist Reports the list of available sounds (if there are sounds available).

space Reports the amount of free memory in bytes. Use recycle before space for better accuracy
sqrt *number* Reports the square root of number (number must not be negative).
st Show turtle.
stamp Stamps a copy of the turtle on the background in the current **pu**, **pd**, **pe**, or **px.**
stop Stops the procedure that is running (used only in a procedure).
stopall Stops all procedures and returns control to the command center (can be used in a button, stop rule, or command center).
stopme Stops the process in which this command was run.
sum *number1 number2* or (**sum** *number1 number2 number3*) Reports the sum of its inputs.

talkto (tto) *turtle/text-boxs* Makes the turtle or text box current.
tan *number* Reports the tangent of the input.
tc Reports the color of the text in the current text box. If there is more than one color it reports an empty list.
textcount *text-box-name* Reports the number of lines in the current text box.
textitem *line-number text-box-name* Reports the designated "line" of the named text box as a long word. **Parse** will chang the long word into a list.
textlist Reports a list containing the names of text files in the current directory (TXT).
textpick *text-box-name* Reports a randomly chosen line from the named text box. Reports empty lines if they exist and are selected. Returns a long word.
textwho Reports the name of the current text box.
thing name Reports the value of the named variable. Can replace the colon eg. **show** :age or **show thing** age
timer Reports a number representing the time elapsed (tenths of seconds) since the program started, or the lastest resett command.
to Shows the beginning of a procedure
top Move the cursor to the beginning of the text in the current text box.
towards turtle-name Sets the heading of the current turtle to aim toward position of the turtle given as input.
true Used as input and output to conditionals.
turtlesown *own* Assigns a variable to all the turtles in the current project. This variable can then be set to a specific value for each turtle. When this command is executed it also creates a special word of **set(**followed by word) to use to set the turtle for the word. (e.g. **turtlesown "speed** creates a **setspeed** command as in **t1, setspeed 12**). Use **remove** to remove a turtle variable for all turtles in a project. **turtlesown** initializes the variable to an empty list.
turtletype *word/list* Sets the turtle to type the given word or list, one character at a time. The turtle will drop a letter at each **fd** or **bk** command following the **turtletype** command until there are no more letters to drop.
type *word/list* (**type** *word/list1 wordlist 2* ...) Prints the word/list in the command center without a carriage return.

unfreeze *word/list-of-page-elements* Unfreezes the button, text ox, turtle, slider, or all objects on a page, so they can be changed using the mouse.
unlock pathname Removes the protection of lock on a page.
unselect Undoes highlight from a **select** or **search**.
unsnaptext *text-box-name* Reactivates a text box.
wait *number* Pauses for 10 ths of a second.
waituntil *true-or-false-list-to-run* Waits until the instruction list reports **true** before going to the next instruction. The input must be an instruction list that reports **true** or **false**. e.g.

```
to bounce
t1,
forever [fd 1]
waituntil [(distance "t2) > 100]
      cancel [fd 1]
      rt 180 bounce
      end
```

who Reports the name of the current turtle.
when *letter listtorun* Programs a control-key event whenever the control key and the letter key are pressed listtorun is run only ten keys can be used N O P Q R V W X Y Z Are saved only on a page where the control keys are inside a procedure. Note if **readlist**, **readlistcc** or **readchar** are waiting to read the keyboard control-keys will be ignored.

clearevents **when** "z [**print** "hello]

word *word/list1 word/list2* (**Word** *word/list1 word/list2 word/list3*...) combines all inputs into one word and reports it.

word? *word/list* Reports **true** if input is a word otherwise **false**.

xcor Reports a number representing the horizontal position of the current turtle.

ycor Reports a number representing the vertical position of the turtle.

+ addition	- subtraction
/ division	* multiplication
= equals	> greater than
< less than	

Super Procedure Project:

Project must include:

1. At least five different and **new** designs. One design is considered a procedure and a complete figure (circle, square, car (if simple) house (if simple). If house is the super procedure then a window and door would be considered a design if they are a procedure.
2. Must have a border all around it. This would be a separate procedure and is not one of the five need for the five designs.
3. Must have a super procedure. When the title of this procedure is entered in the command center the entire project will run. This procedure is not one of the five for designs.
4. Must have a print-out of all the procedures and only the procedures that are needed. If you have extras be sure to erase them before you print the final copy.
5. Must have a print-out of the picture created by the superprocedure.
6. Must be saved on the class disk.

Don't let all of the must haves scare you. It's not that complicated.

Helpful Hints:

- If each design is made so the turtle starts and stops at home it maybe easier to keep track of the turtle's position. You could put all the moves in one procedure or make several move procedures to locate the turtle on the screen before you use a design procedure. Move procedures do not count as one of the required procedures.
- Save your work often. When you start to forget what you have done it's time to print-out the procedures. If you change them on the printout and it starts to get messy, it time for a new printout.
- Use top down planning. Start with the super procedure. Enter it so you won't have to list procedure after procedure each time you want to see how the project is coming.
- Write each proceedure in the order they are in the super procedure. Then debug each design one at a time. Don't be afraid to change your plan.
- If a procedure is getting too long (a screen) or complicated break it into smaller procedures and call it from another procedure.

- Be CREATIVE Have FUN!!!!

Logo promotes the following:

1. Provide for experiential learning. Logo is an environment that allows students to view the results of their interactions with the computer on demand
2. Supports learning for students that are not able or willing to learn in a traditional classroom
3. Allows students to gain confidence in working with computers by being in control of the computer
4. Serve as a vehicle for learning computer literacy: writing structured programs students learn proper techniques, learn how to design programs by breaking down a problem into smaller tasks, learn about branching and conditionals, learn about variables and recursion using the power of a programming language, learn how to manipulate data as numbers, words, and lists.
   - Logo is an easy language to learn: the commands are English-like and easy to remember, friendly and helpful error messages, and it is interactive so the user gets immediate feedback.
   - Logo promotes good programming habits: it encourages the development of short programs (procedures), that are then used as building blocks by other procedures.
   - Logo allows creation and reuse of procedures allows addition of new commands, lets procedures be run and tested independently, is easy to transfer procedures from one project to another, encourages the development of a library of often-used procedures.
   - Logo is easy to write and maintain: programs are easy to debug with tracing and pausing tools, and you can use long and descriptive variable and procedure names.
   - Experienced programmers should be aware that: Logo is derived from LISP, the language of artificial intelligence, Logo is recursive, Logo offers dynamic variable scoping, Logo offers a full range of disk and file handling commands.
5. Encourages top-town thinking.
6. Encourages students to think of a mistake as fixable.
7. Promotes the use of taking another point of view to solve problems.
8. Encourages metacognition.
9. Is a mathematical environment that enables students to do mathematics.

Learn how mathematicians make algorithms to solve problems, problem solving, learn number values, number sequences, similarity of two dimensional figures, rotations, angles, decimal numbers, fractions, infinity and limits, positive and negative numbers, coordinate systems, variables, estimation, geometric shapes, perimeter and area, symmetry, probability, algebra, geometry, trigonometry, calculus, and fractals.

10. Allows creativity.
11. Is motivational.
12. Provides satori, the joy of learning or solving a problem, accompanied with the struggle of the problem.
13. Helps users learn how to be systematic, possibly before quantitative.
14. Allows communication with a language that is understandable by adults.
15. Language Arts: sentence structure by generating Logo sentences that follow the rules of grammar and parts of speech, creative writing through writing stories and poems, word structure through writing stories and poems and programs with rhyming words, to pluralize nouns, and conjugate verbs.
16. Social studies: learn directions by translating the turtle's heading into compass points, cartography by making maps using Logo graphics, creating surveys, graphing data, and creating foreign language primitives and procedure names.
17. Science robotics with Lego Logo and controlling of other devices through Logo, attaching sensors for light, touch, temperature, sound to the computer and reading and processing the data with Logo, and running simulations.
18. Fine arts: using Logo's graphics for computer art, using Logo's sound-generating abilities for music, creative movement by choreographing the turtle, and multi-media.
19. Animation.