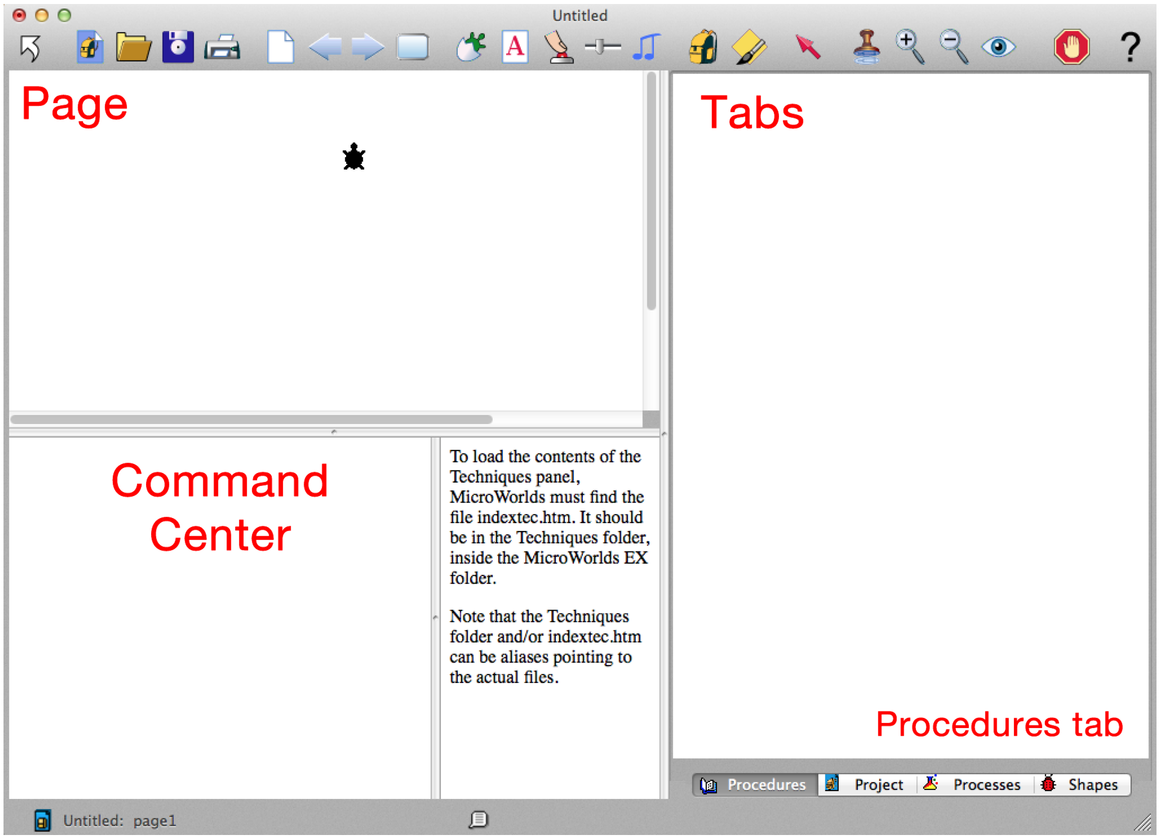
Introduction to Logo Programming in MicroWorlds EX



To load the contents of the Techniques panel, MicroWorlds must find the file indextec.htm. It should be in the Techniques folder, inside the MicroWorlds EX folder.

Note that the Techniques folder and/or indextec.htm can be aliases pointing to the actual files.

MicroWorlds EX (including its variants *Robotics* and *Coder)*, is a multimedia programming language built upon the Logo programming language. It may be used across the curriculum to bring creative ideas to life with mathematical thinking and computer science concepts used in service of student expression.

Anything you can do through direct action with the mouse can also be programmed.

The Language
The Logo language consists of terms built into the language, called **primitives.** Primitives are procedures that may not be changed.

Procedures are terms that a user defines and adds to the language. Projects may have procedures and turtles may have procedures in their backpack. These procedures are saved with each **project** and are not available to other projects without the user taking deliberate action to copy and paste them or by modifying an existing project. **Primitives** behave in exactly the same fashion as a **procedure**, but cannot be modified by the user.

You may not name a user-defined procedure with the same name as an existing primitive. Doing so will result in the following error message:

```
____ is already defined
```

Error messages
Logo/MicroWorlds is known for its clear error messages. Students should be encouraged to read and think about the error messages that appear in the **Command Center** and use that feedback to debug their processes.

Getting Started

Type the following commands one at a time into the command center and notice the error messages generated.

```
Go
Walk
Move
```

Each command generates `an` *`I don't know how to ____`* error message

That means that the word you typed is not (yet) in the vocabulary of MicroWorlds or the turtle.

Type:

```
Forward
```

And the error message changes to *`Forward needs more input`*

That indicates that `forward` is a recognized command, but more information is necessary. Specifically, a number representing how many turtle steps forward.

`Now type:`

```
Forward 40
Forward 20
```

and the turtle travels 60 steps

```
Back 30
```

leads the turtle to back up half the distance it has traveled.

You may combine commands on one one line as long as they are separated by a space. Not including a space will cause an error message or an unpredictable result.

```
forward 40 back 20 rt 90 fd 20 bk 40 fd 20 lt 90
```

When you press RETURN, all of those instructions are run in order from left to right.

Can you figure out what `FD, BK, RT, LT,` and `CG` do?

All error messages can be explained by MicroWorlds simply by choosing Last Message… from the Help menu

Procedures
Anytime you combine lots of instructions in a sequence or intend to use that sequence more than once, you should write a procedure.

Procedures are created in the procedures tab in MicroWorlds and later in turtle backpacks.

All procedures begin with the word *TO* and end with the word *END* – each on their own line.

To is followed by the name of the procedure you are creating and any inputs it may require. You are adding a verb to the language. Typing that word will cause Logo to take action.

I might do the following example with kids.

Click on the hatching turtle tool in the MicroWorlds toolbar and then click in the center of the page to place it there.

The turtle is like you except that she has a pen stuck in her bellybutton and when she moves with her pen down, she leaves a trail.

Let's put the turtle's pen down!

Type `PD` in the command center.

Then type the following procedure in the procedures tab, reinforcing the ideas that:

- procedure names are arbitrary because the turtle doesn't have a large vocabulary and
- procedures begin with `TO` and end with `END`.

```
to atticus
fd 35 rt 60
end
```

Type `atticus` in the command center and see what happens. Ask kids to predict what shape it has started to draw.

Next, add the following procedure below (or above) or above the `atticus` procedure in the procedures tab. Placing a carriage return or two between procedures is a good habit to introduce. It makes reading programs easier.

```
to olivia
atticus
atticus
```

```
atticus
end
```

Type `CG` (clear graphics) and then `Olivia` into the command center

Now what happened?

Let's go totally crazy now! Create the following procedure and run it.

```
to jamie
olivia olivia
end
```

You might also write the Olivia procedure in the following manner.

```
To Olivia
Repeat 3 [atticus]
End
```

Just be sure NOT to have two procedures with the same name. That confuses MicroWorlds.

> MicroWorlds will show you its entire vocabulary of primitives by selecting Vocabulary from the Help menu. However, teachers should refrain from teaching too much programming vocabulary too difficulty!
>
> Powerful problem solving and computational skills are developed by building complexity out of a handful of simple objects – in this case primitives. This is not a race!

Commands and Reporters

The Logo/Microworlds language has two kinds of procedures/primitives, **commands** and **reporters**. Commands tell the system to *do* something and reporters report.

Type `2 + 2` into the command center (be sure to use spaces around the arithmetic symbols)

You should receive the error message, *I don't know what to do with 4*. In this case, the + sign is a reporter that takes two numbers as inputs, adds them together and reports the result. This is called an *infix* reporter because the function or reporter is between the inputs. Logo tolerates some infix reporters (+- * / = < >), but doesn't really like them and often requires the use of parentheses for denoting groups or order of operations.

Logo would really prefer `sum 2 2`, but humans hate that. Sum is what computer scientists call a *prefix* reporter (or sometimes, function).

Typing `show 3 * 4` asks the multiplication symbol to multiply two numbers and report it to `show` whose job it is to display that information in the command center.

Serious computing and interactivity will ultimately require reporters even those most beginner experiences with turtle graphics and the Logo-inspired programming environment, Scratch. Reporters are invaluable helpers who can work together to solve more complex problems.

Commands and reporters may have zero, one, two, or more inputs.

Among these reporters, how many inputs do they each use?

`FD`
`CG`
`RT`
`REPEAT`
`LT`
`SHOW`
`BK`
`PD`
`PU` (penup)

Procedural inputs may be numbers, words, or lists. Numbers require no punctuation, words get a set of quotation marks to the left of the input name and lists are surrounded by square brackets [ ].

We will return to turtle graphics (aka: turtle geometry) at a later point. In the meantime, I would like to introduce you to a form of programming called word and list processing. In fact, while MicroWorlds, Scratch, and Snap! are dialects of Logo, Logo is a dialect of LISP, the preeminent language for artificial intelligence programming since it was created in 1958! That's like a billion in computer years and LISP is still in use.

Why do you need to know all of this? LISP stands for list processing. That's what we will be doing now in a school language arts context.

The Random Gossip Project

Type the following in the command center.

```
Show pick [apple cherry lemon]
```

Try it again or type `repeat show pick [apple cherry lemon]`

What does the primitive `pick` do?

We can use this random picking of items in a list in a number of different projects. We can generate random poetry of all sorts. Doing so gives kids direct experience with parts of speech, because the computer is prone to making hilarious English errors, all while developing important computer science concepts.

Defining Reporter Procedures

Reporters are programmed in virtually the same fashion as commands, except that they **always** contain the primitive, output. Reporters still begin with `To` and end with `End`.

Here is a procedure to try.

```
to person
output pick [  Gary Amy Sylvia James    ]
end
```

You may of course insert different or additional names in the list.
Type `person` in the command center.

What happened?

Type:

```
Show person
```

That should do the trick. `Person` reports an item in its list of names and `show` displays that name in the command center.

Next, write a procedure called `verb` (or doeswhat).

```
to verb
output pick [jumps swings swims [strategizes with] creates flies
likes punches slaps runs kickboxes walks [laughs at]     ]
end
```

Again, you may put any action words you wish in the list.

Do you notice how we handle lists within lists?

In the command center, try:

```
Show verb
Repeat 10 [show verb]
```

Eating and Smushing

Lots of computer programming consists of taking things apart and smushing them back together again. Let's put a sentence of random words together.

```
to gossip
show (sentence person verb person)
end
```

`Gossip` is now a super-procedure that makes everything go. Type it in the command center.

```
Gossip
Repeat 5 [gossip]
```

The parentheses work just like they do in a math equation. In this case, they force the reporter, `sentence`, to accept multiple inputs when it really only likes two.

If you really want to get fancy, try this version of `gossip`.

```
to gossip
show (sentence person verb word person ".)
end
```

How is this version different? What does the reporter, *`word`*, do?

Now add an adjective procedure to your procedures in the procedures center. You should be able to figure out how to it all by yourself now, but here's one more example.

```
to adjective
output pick [hairy mean stinky moldy insane warm snug funny       ]
end
```

How do you test `adjective` to make sure it works?

Challenges:

1. Where would you use your new `adjective` reporter in the `gossip` procedure? Here is a hint. You can use it one or two different places.
2. How would you program the person procedure so that the computer has a greater probability of insulting a particular person – at random of course?

Extension Activity

Add adverbs to your program. You can do it. Where does the `adverb` reporter get inserted in the `gossip` procedure?

Super Duper Challenges

1. How could you modify this program to create random poetry or haiku?
2. Can you program a randomly generated MadLib?