Unit 5: Graphics — **Skill Builder 3: Lines, Text and Colors**

In this third lesson for Unit 5 you will learn about drawing lines, text and color enhancements.

Objectives:

- Use the line, function, and text drawing statements.
- Use colors in graphics statements.
- Develop formulas to utilize graphics in programs.

Drawing Lines and Curves

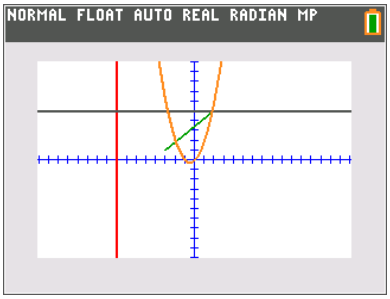
Line(X,Y,W,Z) draws a *segment* between points (X,Y) and (W,Z). See CATALOG HELP for the optional features.
Vertical A draws the vertical line X=A.
Horizontal B draws the horizontal line Y=B.
DrawF X^2+X draws the function. This is different than graphing the function.

See the examples to the right. Note the optional colors found on the [PRGM] COLOR menu. Color is not available on the TI-84 Plus.

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:U5S0
:FnOff
:PlotsOff
:ClrDraw
:ZStandard
:ZSquare
:Line(2,5,-3,1,GREEN)
:Vertical -8,RED
:Horizontal 5,DARKGRAY
:DrawF X²+X,ORANGE
```

This program…

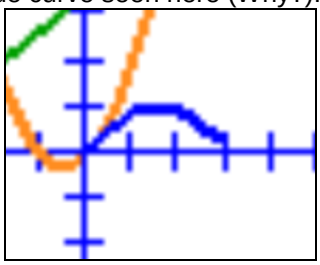
…does this:



Tip: to draw part of a function divide the function by the interval desired:

```
DrawF sin(X)/(X≥0 and X≤π)
```

draws the blue curve seen here (Why?):



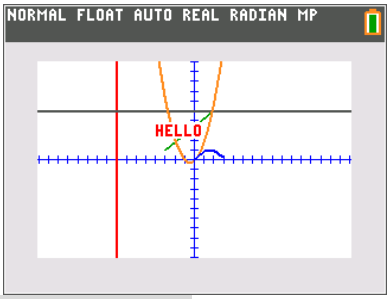
Teacher Tip: The **DrawF** feature is not the same as graphing a function using the [Y=] and [GRAPH] method. A function that is drawn cannot be traced or 'calculated' ([CALC]). The **DrawF** command only draws the pixels that the function determines. A repaint of the screen will erase any drawn function. It will not be redrawn unless the command is issued again. Notice that the **Line( )** statement only draws a line segment. But **Vertical** and **Horizontal** draw complete lines from edge to edge of the screen. This is a teachable moment! A project follows that has students program a complete line rather than a segment.

Text Drawing

The **Text(** drawing statement is unique because it uses *pixel* values rather than window (point) values for positioning the text. There is also a separate **TextColor(** statement that sets the color of the next text drawn.

```
NORMAL FLOAT AUTO REAL RADIAN MP
DRAW POINTS STO BACKGROUND
3↑Horizontal
4:Vertical
5:Tangent(
6:DrawF
7:Shade(
8:DrawInv
9:Circle(
0:Text(
A:TextColor(
```

Text(50,100,"HELLO") displays HELLO in the same spot on the screen regardless of the WINDOW settings. Row 50, column 100 of the pixels represents *the upper left corner* of the text to be drawn.



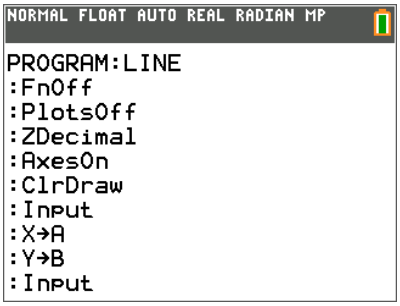
Note: Remember your screen's pixel dimensions: TI-84 Plus: 96 columns x 64 rows and TI-84 Plus C/CE: 265 columns x 165 rows.

Teacher Tip: This is an awkward statement because it is the only one of the Draw commands that utilizes *pixel* coordinates rather than *WINDOW* coordinates. An interesting project is discussed below.

Programming with Line(and Algebra

This programming activity 'enhances' the **Line(** statement.

The **Line(** statement only draws a *segment* between two points. We'd like to see a *line through* the two points and extend all the way across the screen regardless of which two points are selected. This activity makes use of Algebra concepts, so be prepared!



1. Start a new program. We will call it **LINE**.
2. Add the usual graph setup statements to the beginning of the program.
3. Use two **Input** statements *without variables* to get the coordinates of two points on the screen. Input determines the values of **X** and **Y** so we need to store the first two values in other variables, **A** and **B**, so that we can get the second set of coordinates into **X** and **Y**.
4. Calculate and store the *slope* of the line. `:(Y-B)/(X-A)→M`
5. Now we need the two points at the left and right sides of the screen for the line statement. The *x*-coordinates of these points are **Xmin** and **Xmax**.
6. We need to compute the *y*-coordinates.
7. The equation of the line is ***y* = M*(*x* - A)+B** (*point-slope form*).

Your Task…

1. Substitute **Xmin** and **Xmax** (the *names* not the values!) into the equation for *x* and store the results in the two variables **Q** and **R** representing the *y*-coordinates.

Answer: M*(Xmin-A)+B → Q
M*(Xmax-A)+B → R

2. Use the **Line(** statement to draw a line between the left and right side of the screen. `:Line(Xmin,Q,Xmax,R)`

Extensions

1. Add a loop in this program to allow you to draw many lines without having to re-run the program which erases the screen.
2. This program fails when the line is vertical. Why? Incorporate an **If…** structure to handle this special case.

Teacher Tip: Because the slope of the line is undefined. The program tries to divide by zero, causing an error.

Programming Points to Pixels

Imagine this: you use the **Pt-On(** statement to draw a point (**A**,**B**) on the graph screen. You now would like to *label* the point with some text. Where do you draw the text?

Write two formulas (one for **C** and one for **D**) that convert WINDOW coordinates to pixel coordinates for the **Text(** statement. This table (TI-84 C/CE values) may help:

WINDOW	pixel
Xmin	0
Xmax	264
A	**?**
Ymax	0
Ymin	164
B	**?**

Complete the **?** *so that the point is labelled* P:

```
NORMAL FLOAT AUTO a+bi RADIAN MP
PROGRAM:POINT
:?→C
:?→D
:Pt-On(A,B)
:Text(C,D,"P")
```

Note: remember that in the ***Text****(command the first argument is the ROW number which corresponds to the y-coordinate or the point!.*

Teacher Tip: This is another example of using a linear relationship. Think of (Xmin, 0) and (Xmax, 264) as two points on a line.

Then the slope of the line is

(264-0)/(Xmax-Xmin)

so the linear transformation for A is

264/(Xmax-Xmin)*(A-Xmin) → D *(equation of line*)*

Similarly, for B we get a slope of

(164-0)/Ymin-Ymax)

so B's transformation is

164/(Ymin-Ymax)*(B-Ymax) →C *(equation of line*)*

* remember that Pixel-oriented commands use the format (column#, row#, text). This is why A goes to D and B goes to C.

```
NORMAL FLOAT AUTO REAL RADIAN MP
PROGRAM:ALABEL
:Input
:X→A:Y→B
:Pt-On(A,B)
:264/(Xmax-Xmin)*(A-Xmin)→
D
:164/(Ymin-Ymax)*(B-Ymax)→
C
:Text(round(C,0),round(D,0
),"P")
```

round() is necessary because the command only allows whole numbers in the proper range. Let students discover this!

For a TI-84 Plus use 95 instead of 264 and 63 instead of 164.