Unit 6: Drawing | **Skill Builder 1: Basic Shapes**

In this lesson, you will learn about using the **Draw** features in the programming world of **TI-Nspire™ CX II**.

Objectives:

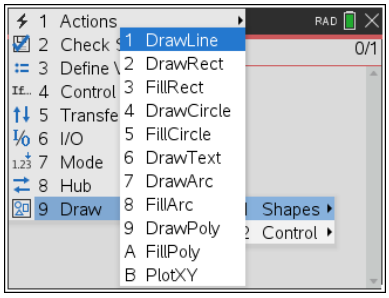
- **Draw** some shapes
- Handling the drawing screen
- Running and ending a program using Draw commands
- Changing the drawing color

Teacher Tip: Unit 6 requires the TI-Nspire CX II with OS 5.0 or higher. The programming language and OS have **Draw**ing features built in.

- The Program Editor contains a Draw menu. *(Note: This unit is not appropriate for the TI-Nspire or TI-Nspire CX series of handhelds.)* The **Draw** tools and other enhanced features of OS 5.0 are also available in the **TI-Nspire CX *Premium* Computer Software.**
- A program that uses **Draw** statements can only run from a Calculator app. It will not run in a Notes app Math Box.
- **Draw**ing takes place in a special new drawing screen that appears 'over' the Calculator app to display the drawing. This drawing screen is not an app but it 'covers' the whole screen. You cannot draw on a Graphs app or a Geometry app. When the program is 'Finished' press any key to return to the Calculator app.

Draw a Line

1. Start a new document and add a Program Editor. Name the program **shapes.**
2. Press **[menu]** and select **Draw >Shapes** and observe the list of shapes available.
3. Select **DrawLine.**

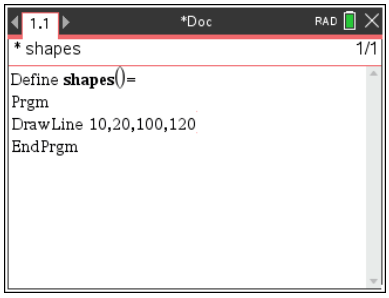


4. To draw a line, specify the endpoints, (*x1, y1*) and (*x2, y2*), of the line.
5. The format of the command is:

   DrawLine *x1, y1, x2, y2*

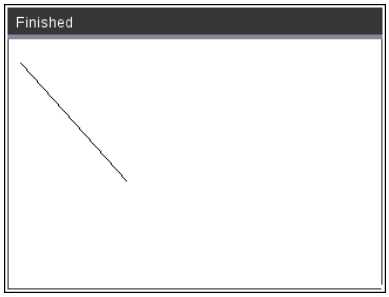
6. To get started, use these values:

   DrawLine 10, 20, 100, 120



Note: *Draw statements do not use parentheses. Using parentheses causes an error.*

7. Press **ctrl-R** to prepare to run the program and press **[enter]** to execute the program on the Calculator app.

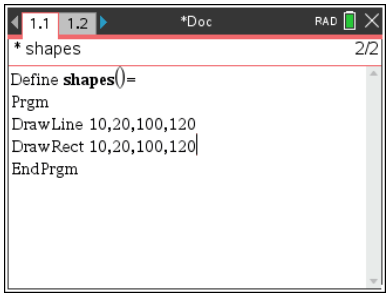


You now see a screen like the one on the right. You are seeing a special graphics drawing 'window' for displaying the Draw commands. (10,20) is the upper left endpoint of the line and (100,120) is the lower right endpoint. The title bar on the screen displays 'Finished' indicating that the program has ended but the handheld still displays the drawing. ***Press any key to close the drawing screen and return to the Calculator app.***

Teacher Tip: This default screen measuring unit is *pixels.*
If you use values that are not in the viewing window there is no error, but you may not see anything because the object is drawn off the screen.
While the graphics drawing window is active, I/O commands like **Request** and **Disp** are disabled. The **getKey( )** function is operational and it will be discussed in Skill Builder 3.

DrawLine actually produces only a line segment. Here's an opportunity to make use of good Algebra: write a program that accepts two points (4 arguments) and draws a line *through* those points and all the way across the screen. This requires knowledge of the slope of a line and the point-slope form for the equation of a line.

Draw a Rectangle

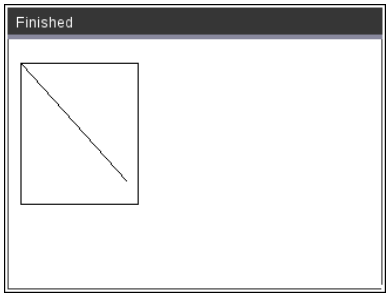


8. Return to the Program Editor and, after the **DrawLine** statement, add a line of code to draw a rectangle: choose **DrawRect** from **[menu] > Draw > Shapes**

   DrawRect 10,20,100,120

   Use the same four values that you used for the **DrawLine** statement.

   Can you predict what will happen?



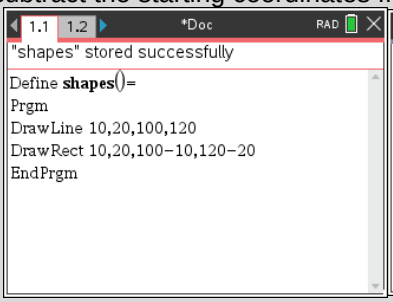
9. Press **ctrl-R** and run the program again. Why is the lower right corner off a bit?

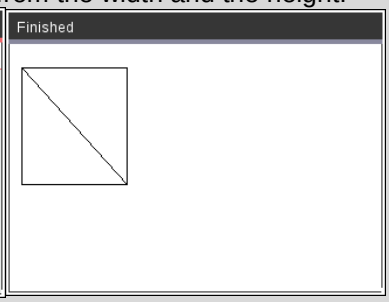
   The **DrawRect** command uses the form:

   DrawRect *x, y, width, height*

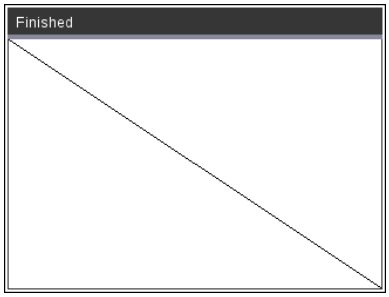
   What should the *width* and *height* of the rectangle be so that the line is a diagonal of the rectangle?

Teacher Tip: Answer**: DrawLine 10,20, 90, 100**
You have to subtract the starting coordinates from the width and the height.





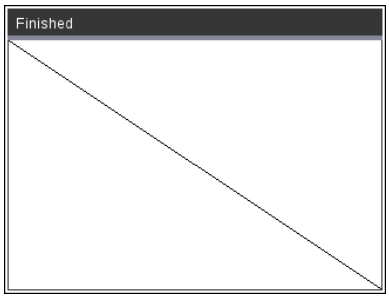
Bet you're wondering what the screen dimensions are. Before advancing, see if you can draw a line that is a diagonal of the screen.



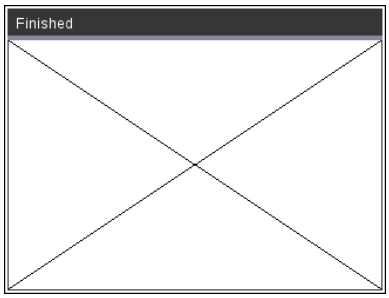
Screen Size

- The drawing screen is 318 x 212.
- The upper left corner of the screen is (0,0).
- The lower right corner of the screen is (317, 211)

10. The statement **DrawLine 0, 0, 317, 211** draws a diagonal on the screen.



11. Write the statement that draws the other diagonal of the screen.



Teacher Tip: To draw the other diagonal use the statement:
DrawLine 0, 211, 317, 0

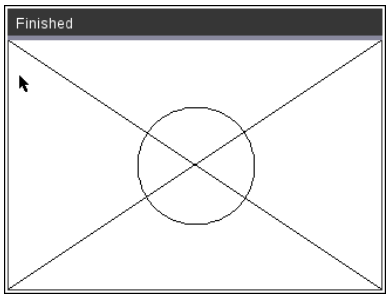
Other Shapes

DrawCircle *x, y, r* where (*x,y*) is the center and *r* is the radius

Example:

12. **DrawCircle** 159, 106, 50

FillRect and **FillCircle** are similar but they fill the shapes with the current color.



Teacher Tip: How would you draw a square? Make the width and height values the same. Remember we're talking about pixels for now.

FillRect can be used to change the 'background' color from white.

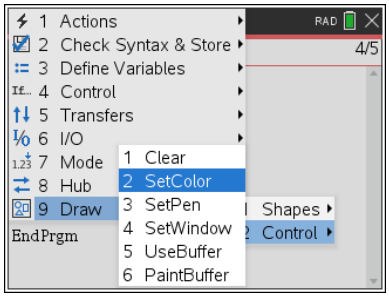
Example:

SetColor 255,255,0 *yellow*
FillRect 0,0,317,211 *default window*

to make a yellow screen.

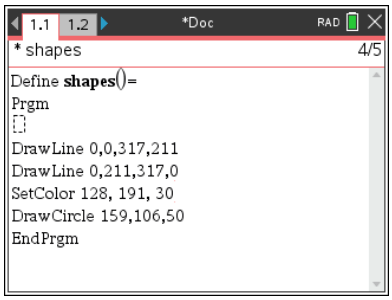
If you use **SetWindow** then the **FillRect** values change accordingly.

Changing Drawing Color



13. In the Program Editor, on a blank line choose the **SetColor** statement from **[menu] > Draw > Control > SetColor**

R, G, B



Color is determined by three values: the amount of **R**ed, **G**reen and **B**lue to mix together. Each of these three values must be in the range 0..255.

SetColor *red_value, green_value, blue_value*

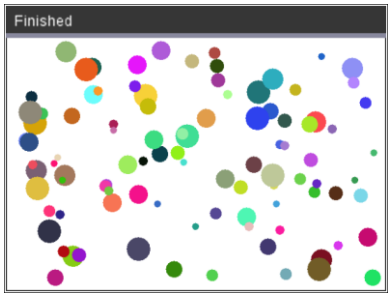
Example:

SetColor 128, 191, 30

Teacher Tip: Certain colors such as yellow, cyan, and magenta have special RGB values:

Yellow is a mix of red and green
Cyan is a mix of green and blue
Magenta is a mix of red and blue

The higher the numbers (closer to 255), the brighter the color.



14. Use the **SetColor** statement *before* drawing something.
15. Try writing a program that draws 100 random filled circles in random colors. Use a **For** loop.

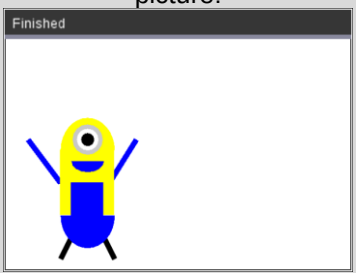
If you want a circle and its interior to be two different colors use two different pairs of statements:

Setcolor 255, 0, 0	*red*
FillCircle 100, 100, 50	*interior of circle*
SetColor 0, 0, 0	*black*
DrawCircle 100, 100, 50	*the circle itself*

… draws a black circle with a red interior. But if you *draw* the circle first, then the *filled* circle, you will not see the black circle because the red covers it. Try it.

Teacher Tip: Possible solution to draw 100 random circles:

```
Local i, r, g, b, x, y, ra
For i, 1, 100
  r := randInt(0, 255)
  g := randInt(0, 255)
  b := randInt(0, 255)
  © we want the circles to be on the screen…
  x := randInt(10, 308)
  y := randInt(10, 202)
  ra := randInt(3, 10)      © can't use r here because its already used for red
```

```
  SetColor r, g, b
  FillCircle x, y, ra
EndFor
```

Another challenge to try: Using only the statements introduced in this lesson, draw a picture: