The Action Block

The action block in programming is the most basic type of structure. It is a set of statements that are performed sequentially. In this lesson, you will explore solutions to several problems that employ action blocks.

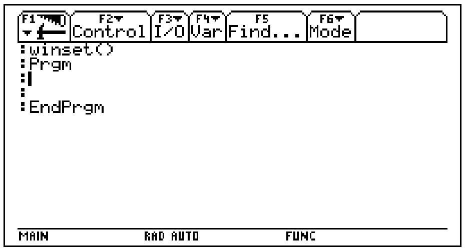
Instructions

Part A — Drawing a Specific Object

The following instructions tell you how to draw a wagon. You begin with a program that sets the values of the Window variables. Next, you enter a program that draws a box using the specified Window variable values. The final program adds wheels and a handle to the box to draw a wagon.

Setting the Values of the Window Variables

1. First, reset memory to the default settings. From the Home screen, press [2nd] [MEM] [F1], select **3:Default**, and then press [ENTER] [ENTER].
2. To start a new program in the Program Editor, press [APPS], select **7:Program Editor**, and then select **3:New**. Press ⌄ ⌄ and type **winset** as the name of the new program variable.
3. Press [ENTER] [ENTER] to store program name and display the new program template.



```
:winset()
:Prgm
:
:
:EndPrgm
```

4. Type the program lines as shown to the right. (Notice the indentation pattern.) Press [ENTER] at the end of each line.

   To type →, press [STO▸].

   The © symbol indicates a program comment, which you can use to explain or describe various aspects of the program. Any information to the right of the © is not executed when you run the program.

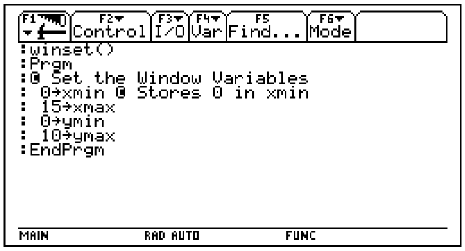
   To type ©, press [2nd] **X.**

```
winset()
Prgm
© Set the Window Variables
 0→xmin © Stores 0 in xmin
 15→xmax
 0→ymin
 10→ymax
EndPrgm
```

5. After you have entered all the program lines, your screen should look like the one to the right. Now press [♦] [HOME].

6. To run the program, type **winset()** and press [ENTER].

   Although **Done** is displayed on the Home screen, nothing seems to have happened. This program changes the Window variable values for function graphing. To see the results, press [♦] [WINDOW]. The values for **xmin**, **xmax**, **ymin**, and **ymax** should match the values in the program.



```
:winset()
:Prgm
:© Set the Window Variables
:  0→xmin © Stores 0 in xmin
:  15→xmax
:  0→ymin
:  10→ymax
:EndPrgm
```

Drawing the Box

1. Start a new program in the Program Editor, and name it **drawbox**.

2. Enter the program lines as shown to the right, making sure to indent for readability.

3. Press [♦] [HOME]. Type **drawbox()** and press [ENTER].

   Where is the box located on the screen? How have the menu items at the top of the screen changed? How would you get back to the Home screen?

   The box is located toward the lower left part of the screen. The menu items now relate to graphing since you are on the Graph screen. You can return to the Home screen by pressing [♦] [HOME].

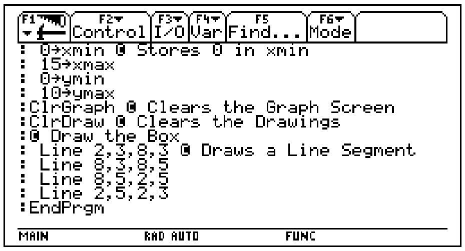
```
drawbox()
Prgm
© Set the Window Variables
 0→xmin © Stores 0 in xmin
 15→xmax
 0→ymin
 10→ymax
ClrGraph © Clears the Graph Screen
ClrDraw © Clears the Drawings
© Draw the Box
 Line 2,3,8,3 © Draws a Line Segment
 Line 8,3,8,5
 Line 8,5,2,5
 Line 2,5,2,3
EndPrgm
```

Drawing the Wagon

You can continue to learn programming techniques and commands by building on the previous program. The new program adds wheels and a handle to the box in the **drawbox** program.

You can begin a new program named **drawagon** and type all of the program lines shown below. However, since the new program uses the same lines you entered in the **drawbox** program plus some additional lines, you can follow these steps to copy the lines from the **drawbox** program and then enter the new lines.

1. Press [APPS], select **7:Program Editor**, and then select **2:Open**. Press ⌄ ⌄ and then › and highlight **drawbox** as the name of the program variable.



```
: 0→xmin © Stores 0 in xmin
:  15→xmax
:  0→ymin
:  10→ymax
:ClrGraph © Clears the Graph Screen
:ClrDraw © Clears the Drawings
:© Draw the Box
: Line 2,3,8,3 © Draws a Line Segment
: Line 8,3,8,5
: Line 8,5,2,5
: Line 2,5,2,3
:EndPrgm
```

2. Press [ENTER] [ENTER] to display the program.
3. Move the cursor to the beginning of the first line following **Prgm**.
4. Hold down [↑] and press ⌄ 12 times to highlight the 12 lines in the program. Then press [♦] **C** to save a copy of these lines.
5. To start the new program, press [F1] and select **3:New**. Press ⌄ ⌄ and type **drawagon** as the name of the new program variable.
6. Press [ENTER] [ENTER] to store the program name and display the new program template.
7. Press [♦] **V** to paste the lines from **drawbox** to **drawagon**.
8. Type the remaining program lines as shown to the right. Press [ENTER] at the end of each line.

   ***Note:** The **ClrGraph**, **ClrDraw**, **Line**, and **Circle** commands can be entered in lowercase letters; but after you run the program and return to the Program Editor, they appear either with an initial capital letter for a single-word command or with capital letters at the beginning of both words for a compound-word command. Program names are always displayed in all lowercase letters.*

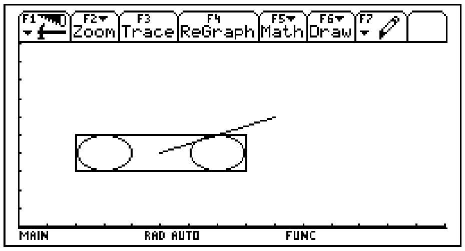
```
drawagon()
Prgm
© Set the Window Variables
 0→xmin © Stores 0 in xmin
 15→xmax
 0→ymin
 10→ymax
ClrGraph © Clears the Graph Screen
ClrDraw © Clears the Drawings
© Draw the Box
 Line 2,3,8,3 © Draws a Line Segment
 Line 8,3,8,5
 Line 8,5,2,5
 Line 2,5,2,3
© Draw the Wheels and Handle
 Circle 3,4,1 © Center (3,4) Radius 1
 Circle 7,4,1
 Line 5,4,9,6
EndPrgm
```

9. Return to the Home screen, and run the **drawagon** program.

   What do you notice?

   One benefit of writing programs that draw objects is that you can observe mistakes easily. It is clear that the wheels and handle are not in the correct place. Plan what to do and how to do it by thinking carefully about how you would place these objects correctly.

   Consider how you would solve the problems presented in the examples before you enter the programming lines. Then think about the results obtained after running the program and ask yourself if the results are reasonable.



10. To place the wheels and handle in the correct location, press [APPS], select **7:Program Editor**, and then select **1:Current**. Now edit the wheels and handle program lines as shown to the right.

    Did you think about how you could correct the mistakes before replacing the lines?

```
Circle 3,2,1 © Center (3,2) Radius 1
Circle 7,2,1
Line 8,5,12,7
```

11. Return to the Home screen, and run the program again.

    Are the wheels and handle correctly placed now? Thinking carefully about solutions often saves time.

Part B — Making a Program More Flexible

When you make a program flexible, it can be used to solve a greater variety of problems. This next section shows you how to add program code that:

- Lets the user choose the exact location of the box.
- Lets the user choose the Window variable values for the screen and choose the exact location of the box.
- Lets the user choose the size of the box and choose the exact location of the box.

Letting the User Choose the Location of the Box

1. Start a new program in the Program Editor and name it **placebox**.
2. Enter the program lines as shown to the right.

   Note: *In this book, when a line is too long to fit on the screen as in the eighth program line here, it is continued and indented on the next line. However, you should type the line without breaking it and let the calculator wrap the line.*
3. Run the **placebox** program.
4. Enter the x- and y-coordinates of the lower left corner when requested. Think about the currently defined Graph screen and where you might locate the corner so that the entire box is displayed.

   Was the box displayed where you expected?

```
placebox()
Prgm
© Draw a Box at a Location You Choose
© Set the Window Variables
 0→xmin © Stores 0 in xmin
 15→xmax
 0→ymin
 10→ymax
© Choose the Location
 Input "Enter x-coordinate of lower left
  corner",x
 Input "Enter y-coordinate",y
ClrDraw © Clears the Graph Screen
© Draw the Box
 Line x,y,x+2,y © Draws a Line
  Segment
 Line x+2,y,x+2,y+2
 Line x+2,y+2,x,y+2
 Line x,y+2,x,y
EndPrgm
```

Letting the User Choose the Window Variable Values and the Box Location

Let's continue to extend this example. Suppose you would like to set up a specific Graph screen while the program is executing and then draw the box on that screen. Once again, think about how you can accomplish this before you enter the program lines.

You need a request to the user to input each of the main Window variable values: **xmin**, **xmax**, **ymin**, and **ymax**. Then you must be able to input the desired values.

The following program extends the **placebox** program. As you enter the program, you can either type the commands **Input**, **Line**, and **ClrDraw** from the keyboard or paste them in from the appropriate menu. For example, to paste the **Input** command in your program, place the cursor at the appropriate position, press F3, and then select **3:Input**.

Before you begin the new program, compare it to the **placebox** program. Remember that you can copy lines from one program to another as you did with the **drawagon** program in Part A of this lesson.

1. Start a new program in the Program Editor and name it **chuzwin**.

2. Enter the program lines as shown to the right.

   Notice that the four **Input** lines for the Window variable values are almost identical. Therefore, after typing the first one, you could follow these steps to copy and edit it instead of typing the other three lines.

   a) Type the first **Input** line and then move the cursor to the beginning of the line.

   b) Hold down $\boxed{\uparrow}$ and press ⌄ once. This highlights the line.

   c) Press $\boxed{\blacklozenge}$ **C** to copy the line.

   d) Move the cursor to the beginning of the next line.

   e) Press $\boxed{\blacklozenge}$ **V** to paste the copy of the line here.

   f) Edit the line to match the program at the right.

   g) Repeat these steps for the other two **Input** lines.

   Enter the remaining program lines.

3. Return to the Home screen, and run the **chuzwin** program.

   Did your choice of *x*- and *y*-coordinates place the box in the Graph screen? Is the box scaled as you thought it would be? Can you think of *x*- and *y*- coordinates that place the box toward the lower right corner of the Graph screen?

```
chuzwin()
Prgm
© Draw a Box at a Location You Choose
© Choose your Window Values
 Input "Enter xmin value",xmin
 Input "Enter xmax value",xmax
 Input "Enter ymin value",ymin
 Input "Enter ymax value",ymax
© Choose the Location
 Input "Enter x-coordinate of lower
  left corner",x
 Input "Enter y-coordinate",y
ClrDraw © Clears the Graph Screen
© Draw the Box
 Line x,y,x+2,y © Draws a Line
  Segment
 Line x+2,y,x+2,y+2
 Line x+2,y+2,x,y+2
 Line x,y+2,x,y
EndPrgm
```

Letting the User Choose Box Size and Box Location

Are there other ways to extend the program? Perhaps you would like to be able to choose the length and width of the box along with the placement of the lower left corner of the box. In addition, suppose you would like to ensure that the entire box actually appears in the Graph screen.

Again, think about how you can accomplish these tasks before typing in the program.

1. Start a new program in the Program Editor and name it **chuzbox**.
2. Enter the program lines as shown to the right.
3. Return to the Home screen, and run the **chuzbox** program.

   Experiment with any values you desire for the width and height of the box. Do the same for the location of the lower left corner.

   Was the entire box displayed in the Graph screen? Can you choose values for the dimensions of the box and location of the corner whereby only a portion of the box is drawn in the Graph screen? Try several combinations.

   By thinking about what the program does, you can be sure that the entire box is always displayed in the Graph screen since the Window variable values are determined by the numbers you input.

```
chuzbox()
Prgm
© Choose a Box Size and its Location
© Program also Determines Viewing Window
© Choose your Box Size
 Input "Enter width of box",x
 Input "Enter height of box",y
© Choose the Location
 Input "Enter x-val of low left
  corner",xx
  Input "Enter y-val of corner",yy
© Set the Window Values
 xx-4→xmin
 xx+x+4→xmax
 yy-4→ymin
 yy+y+4→ymax
ClrDraw © Clears the Graph Screen
© Draw the Box
 Line xx,yy,xx+x,yy © Draws a Line
  Segment
 Line xx+x,yy,xx+x,yy+y
 Line xx+x,yy+y,xx,yy+y
 Line xx,yy+y,xx,yy
EndPrgm
```

Putting It All Together

An action block is a set of programming commands that are performed sequentially. You can string simple commands together to perform fairly complicated tasks. You can join these tasks in a program to solve problems. In this lesson, you created an action block from a set of commands and then combined a set of action blocks to graph simple objects in a several locations on the screen.

Good Structured Programming Techniques

Comments describe what the program does and what sets of commands do. You also can use comments to describe what a particular statement does.

Indenting some program lines makes the program easier to read and understand. Readable programs are easier to work with when you return to them after some time.

Program Editing

You enter and edit programs in the TI-92 Program Editor. You can copy and paste to move characters, words, or lines from one place to another. These editing features are on the **Tools** menu (F1) along with their shortcut diamond equivalents.

Extending Programs

As you saw in Part B, sometimes you can work out the program details to solve a general problem by first considering a specific instance of that problem. Once you have worked out the flow of the specific problem, you can extend that program solution to handle more general situations. The final program evolves from these extensions as you come closer and closer to the general case.

Summary of Commands

©	Characters following © are comments and are not executed.
Circle x,y,r	Draws a circle with its center at (x,y) and radius of r.
ClrDraw	Clears the drawing from the screen.
ClrGraph	Clears the Graph screen.
Input "First Coordinate",a	Prints the words between the quote marks (" ") and then pauses until the value for a is entered.
Line $x1$,$y1$,$x2$,$y2$	Draws a line between the points ($x1$,$y1$) and ($x2$,$y2$).
a $\boxed{\text{STO▸}}$ b	Stores the value on the left (a) in the variable on the right (b).
Prgm…EndPrgm	Designates the beginning and end of a program.

Practice Problems

Think about how you might solve the following exercises before you write the program lines. In addition, run each program to be sure it gives the desired results.

Problem 1

Modify the **winset** program in Part A by doubling the values. Run the program after you modify it, and check to see that the Window variable values have changed to the values in your modified program.

Problem 2

Modify the **winset** program in Part A by setting the minimum values to the negatives of the maximum values. Run the program after you modify it, and check to see that the Window variable values have changed to the values in your modified program.

Problem 3

Modify the **drawbox** program in Part A so that the width of the box is four times the height. The values you use should ensure that the entire box is displayed in the Graph screen.

Problem 4

Modify the **drawagon** program in Part A so that the wheels are 25% bigger, yet still appear at an appropriate place on the body of the wagon.

Problem 5

Modify the **placebox** program in Part B so that each side of the box is six units in length.