Unit 6: Drawing | **Application: Manual Animation**

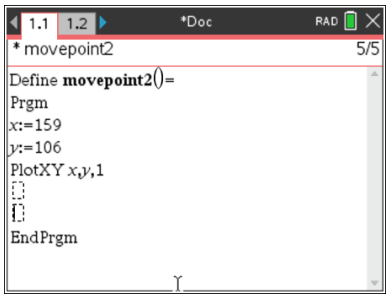
Write a program to move an object on the screen using the arrow keys (an interactive animation). This forms the basis of many video games.

Objectives:

- Use **getKey(1)** to read the keypad
- Make changes to coordinates depending on keypresses
- Erase and redraw a shape

In the previous lesson you made a point move across the screen by itself and used **getKey(0)** to end the program. Now you will write a program that lets the *user* move the point (or any shape) using the arrow keys. In this project you will use **getKey(1)** which pauses processing and waits for a keypress. Only the arrow keys ("up", "down", "left", right") and the "esc" key are important to this project.

Getting Started: Draw the Point



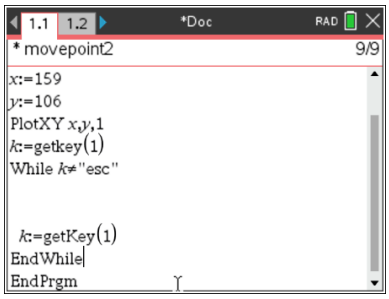
1. Start a new program.
2. **DO NOT USE SetWindow**.
   The *default* window is (0..317) x (0..211).
3. The point will start near the center (159,106) of the screen:

   x := 159

   y := 106

   PlotXY x, y, 1

The Main Loop

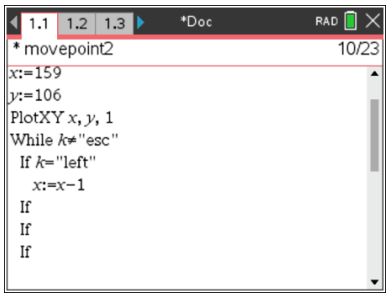


4. The main program loop continues until the "**esc**" key is pressed.

```
k := getKey(1)          wait for a keypress to start
While k ≠ "esc"         as long as it's not [esc]
        loop body
  k := getKey(1)        wait for a keypress to continue
EndWhile
EndPrgm
```

Teacher Tip:
getKey(1) pauses program processing and sits there waiting for a keypress. When a key is pressed processing continues. The program considers the four arrow keys and the esc key. Other keys are ignored.

The Arrow Keys



5. The four arrow keys are "up", "down", "left" and "right". Check **k** for each of these values and change either **x** or **y** accordingly:

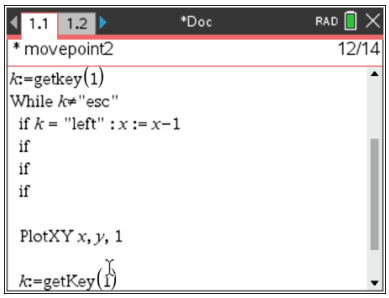
```
If k = "left"
    x := x – 1          (1 is arbitrary; it could be larger)
```

   Recall that this is the simplest form of the **If** statements (no **Then** or **Else**).
   Compete the three other **If** statements for the other arrow keys.

Re-draw the Point

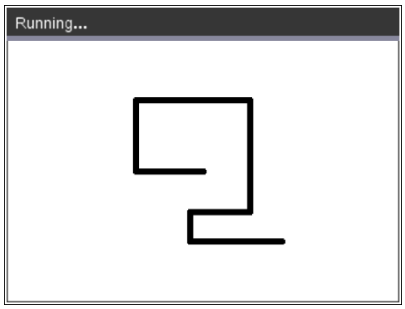
6. After the four **If** statements, plot the point again using its new coordinates.

PlotXY x, y, 1



Teacher Tip: If pressing **[esc]** does not terminate the program (it's stuck in an infinite loop) try pressing the ON/home key repeatedly to 'break' the program.

Test the Program



Use the arrow keys. Press **[esc]** to end the program. If all goes well the point moves… and leaves a trail behind.

What happens when the point gets to the edge of the screen? Currently, it goes off the screen. There are some choices to make here: you can make it simply stop at the edge and not go any further or you can make it go elsewhere, typically to the opposite edge of the screen. This last option is called wrap-around or *toral* mode.

Complete the Project

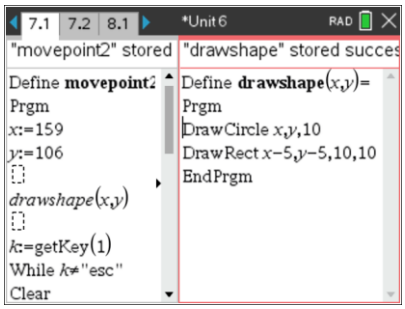
In Skill Builder 3 you learned about 'erasing a point' Add that feature to this project so that the point only appears to move.

To handle the edges of the screen, add four **If** statements to the code:

If x > 317

x:= ?

Teacher Tip: 'Toral' refers to movement behavior on a torus (a 'donut' shape that has no edges) in which 'up' gets you around to the bottom and 'left' gets you around to the right. Every 'straight' path wraps around the torus.



Can you move an entire set of shapes?

This project only deals with manually moving a point. Consider replacing the point with some *shapes*. An efficient method for doing this would be to write a separate program (a 'subroutine'). Within the same problem, write a new program called **drawshape(x, y)** and, in the **movepoint2** program, use the statement

drawshape(x, y) instead of **PlotXY x,y,1**. See an example to the right.

In the **drawshape(x, y)** program, draw some shapes using **x** and **y** as a reference point. Your object can then consist of several different pieces and use several colors.

To erase the shapes consider using the **Clear** statement (**[menu] > Draw > Clear**). **Clear** all by itself clears the screen. To erase or clear just a rectangular portion of the screen use:

Clear *x, y, width, height* *(similar to* **DrawRect***)*

Have fun!

Teacher Tip: Clear makes the region white. If the background color has been changed using a large **FillRect** statement, then use **FillRect** again to erase using the background color.

Mathematics Challenge: in the **drawshape()** example above, make the rectangle's corners on the circle (the rectangle should appear to be *inscribed* in the circle). What values are needed in the **DrawRect** statement?

Possible solution (does not address screen edges):

```
Define movepoint2()=
Prgm
x := 159
y := 106
PlotXY x, y, 1
k := getKey(1)
While k ≠ "esc"
  SetColor 255, 255, 255
  PlotXY x,y,1
  If k = "left"
        x := x - 1
  If k = "right"
       x := x + 1
  If k = "up"
       y := y - 1
  If k = "down"
       y := y + 1
  SetColor 0, 0, 0
  PlotXY x, y, 1
  k := getKey(1)
EndWhile
```

EndPrgm

To handle the screen edges, add four more If statements after checking the arrow keys:

```
If x < 0
   x := 0                    (to stop at the edge; use x := 317   for a wraparound/toral effect)
If x > 317
   x := 317                   (or x := 0 …)
…etc.
```

There are more elegant methods.

If you have access to a TI-Innovator Rover try incorporating Rover movements to match the point's movement on the screen (possibly using Send "RV TO XY …").