Unit 5: The TI Modules

Application: Move it!

In this Application, you will build the core of an interactive graphical program by using the arrow keys on the keypad to move an object on the screen.

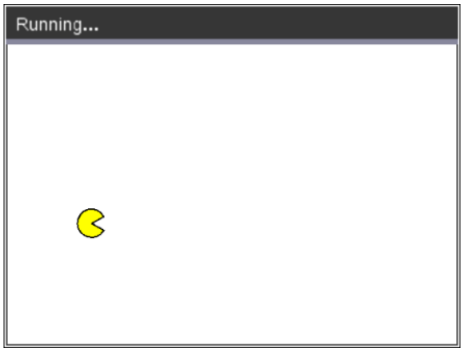
Objectives:

- Combine skills and concepts from this course in a rich and engaging graphical programming project

---

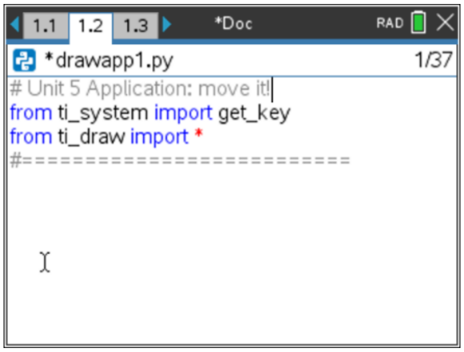
Putting it all together…

Write a program that 'listens' for your keypresses and draws something that moves on the screen in the direction of the key.



1. We'll need **get_key()** to monitor keypresses and we'll need the drawing functions to make an picture of something that will move around.

   In a blank Python file, import both **ti_system** and **ti_draw**. Or, use one of the templates and add the missing imports.



*drawapp1.py 1/37

```
# Unit 5 Application: move it!
from ti_system import get_key
from ti_draw import *
#=========================
```

2. We need to know what key is pressed so that we can act on it, so we store the key value in a variable:

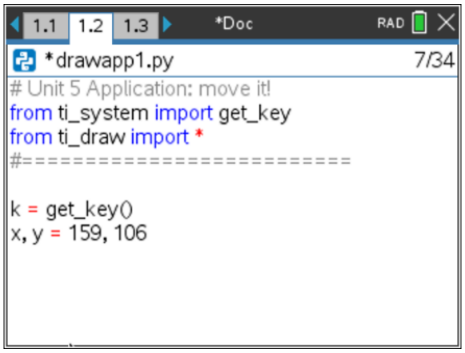
   k = get_key()

   We start our object (a circle) at the center of the screen:

   x=159
   y=106

   This can also be written: **x,y = 159,106**

   Pressing an arrow key will change one of these values and redraw the object in a new position, making it appear to move.



*drawapp1.py 7/34

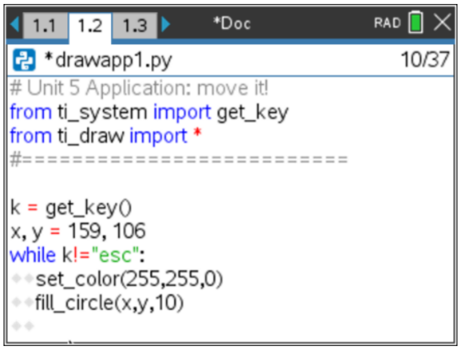
```
# Unit 5 Application: move it!
from ti_system import get_key
from ti_draw import *
#=========================

k = get_key()
x, y = 159, 106
```

Teacher Tip: Consider encouraging students to write functions to draw and erase the shape. A simple erase statement is **clear()** which clears the entire screen. Also, **clear(x,y,w,h)** clears a rectangular area.

3. Begin the main loop:

   while k != ”esc”:

   Create a colorful circle. Use a color and a radius of your choice.



```
# Unit 5 Application: move it!
from ti_system import get_key
from ti_draw import *
#==========================

k = get_key()
x, y = 159, 106
while k!="esc":
  set_color(255,255,0)
  fill_circle(x,y,10)
```

Teacher Tip: The color and radius values are arbitrary.

4. Check the key that was pressed.

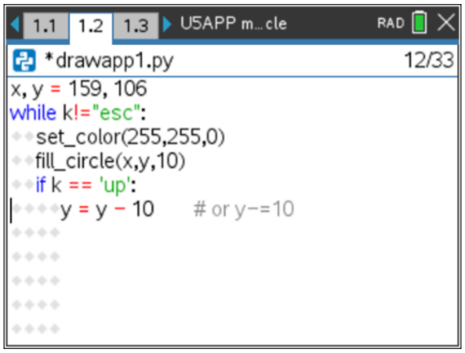
   if k == “up”:
   y = y – 10 (or **y -= 10**)

   This means: If **up-arrow** is pressed, subtract 10 from the y-value. This causes the circle to move up 10 pixels on the screen.

   Write three more **if** statements for the other three arrow keys (“down”, “left”, and “right”, changing x and y appropriately.
   All the **if** statements are part of the **while** block.

   At the bottom of the block, read the keypress again (**k = get_key()**) *inside* the **while** loop since the first **k = get_key()** statement is not part of the loop.

   When done, test your program.



```
x, y = 159, 106
while k!="esc":
  set_color(255,255,0)
  fill_circle(x,y,10)
  if k == 'up':
    y = y − 10      # or y−=10
```

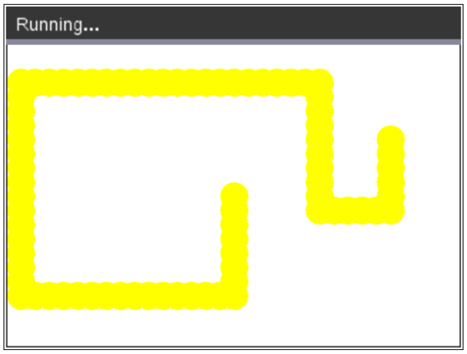
5. Use the arrow keys to move the circle. Does the circle move? If so, congratulations!

   But it leaves a trail behind.

   Erase the old circle before drawing the new one by using:

   set_color(255,255,255)
   fill_circle(x,y,10)

   before changing the values of x and y.

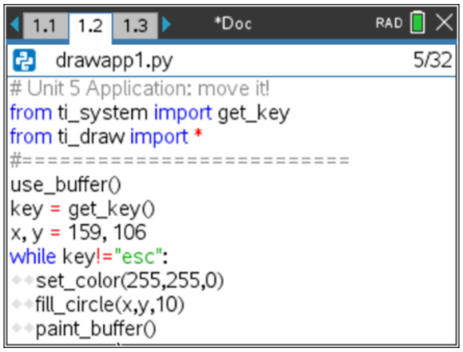


6. The circle now ‘blinks’ because it is being constantly drawn in yellow, and then immediately in white. We only want the screen to be updated when the yellow circle is drawn.

   So, at the top of your program (before the while), place the statement

   use_buffer()

   found in **TI Draw > Control**.



```
# Unit 5 Application: move it!
from ti_system import get_key
from ti_draw import *
#==========================
use_buffer()
key = get_key()
x, y = 159, 106
while key!="esc":
  set_color(255,255,0)
  fill_circle(x,y,10)
  paint_buffer()
```

   And, right after the yellow circle is drawn, place the statement

   paint_buffer()

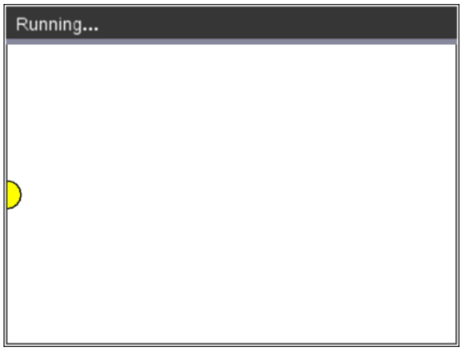
   and run the program again.

   Better? The screen is repainted *only* after the yellow circle is drawn.

Teacher Tip: A little about **use_buffer()** and **paint_buffer()**:
These two functions give you more ‘Control’ over exactly *when* the screen drawings are updated.
use_buffer() tells the computer to draw things on a memory buffer, a piece of memory that is off the screen.
paint_buffer() is the signal to transfer everything in the buffer to the screen.
So, erasing the yellow circle and drawing a new one takes place off the screen and is painted all at once so that all you see is a yellow circle (in a different location).

7. Finally (for now), what happens when the circle goes off the screen? It just keeps going, going, going. Modify your program with more **if** statements to detect the edge of the screen and act. You have two options here:

   a. When you get to the edge, stay there, and do not go off the screen (if x<0 then make x=0, etc.).
   b. When you go off one edge of the screen, wrap around to the opposite side (if x<0 then make x=317, etc.).



   Good luck!

   Extra challenge: Modify your program to let the circle keep moving in the last direction even when a key is not pressed.

   Sample Code for this project on next page.

Teacher Tip: Sample code for this project:

```
# Unit 5 Application: move it!
from ti_system import get_key
from ti_draw import *
#==========================
use_buffer()
key = get_key()
x,y = 159,106
while key!=”esc”:
  set_color(255,255,0)
  fill_circle(x,y,10)
  set_color(0,0,0)
  draw_circle(x,y,10)
  paint_buffer()
  set_color(255,255,255)
  fill_circle(x,y,10)
  draw_circle(x,y,10)
  if key == ‘up’:
     y = y – 10  # or y-=10
  if key == ‘down’:
      y+=10
  if key ==’left’:
      x-=10
  if key ==’right’:
      x+=10
   if x<0:x=317
   if x>317:x=0
   if y<0:y=211
   If y>211:y=0
   key = get_key(0)
```