



15-112
Lecture 2

Animations

Instructor: Pat Virtue

Tuesday Logistics

As you walk in

Quiz will start at the beginning of lecture

- Have pencil/pen ready
- Silence phones



Quiz

Before we start

- Don't open until we start
- Make sure your name and Andrew ID are on the front
- Read instruction page
- No questions (unless clarification on English)

Additional info

- 25 min

Announcements

Same as usual 😊

HW 4 Tips

VS Code set up

- Read instructions carefully

Autograder with animations

- Read piazza post
- Video different than current CS Academy autograder

Start early

- It's a lot of code to write

Thursday Logistics

Announcements

Quiz 3 debrief

- 5 point bump applied in Gradescope

Unit 3 Sections

- 3.7-3.9 Helpful (and fun!) but won't block your from proceeding to exercises
 - You should still be able to do basic bounded motion though, 3.8

Basic Shapes

Rectangles, Ovals, Lines, and Labels

```
drawRect(x0, y0, width, height)
```

```
drawOval(centerX, centerY, width, height)
```

```
drawLine(x0, y0, x1, y1)
```

```
drawLabel(text, centerX, centerY)
```

Rectangles, Ovals, Lines, and Labels

```
drawRect(x0, y0, width, height, fill='black', opacity=100,  
        border=None, borderWidth=2, rotateAngle=0)
```

```
drawOval(centerX, centerY, width, height, fill='black', opacity=100,  
        border=None, borderWidth=2, rotateAngle=0)
```

```
drawLine(x0, y0, x1, y1, fill='black', lineWidth=2, opacity=100,  
        dashes=False, arrowStart=False, arrowEnd=False)
```

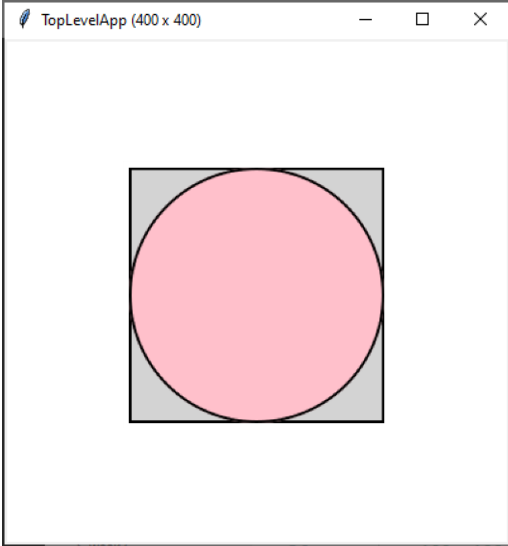
```
drawLabel(text, centerX, centerY, fill='black', opacity=100,  
        border=None, borderWidth=2, rotateAngle=0,  
        size=12, font='arial', bold=False, italic=False, align='center')
```

Poll 1

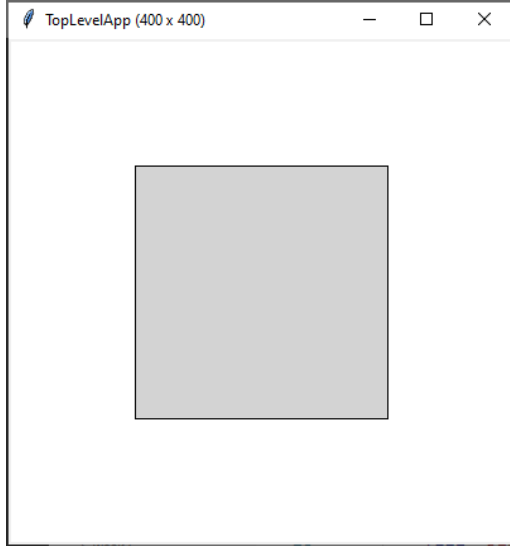
Which does this draw?

```
def drawBubble(app):  
    drawOval(200, 200, 200, 200, fill='pink', border='black')  
    drawRect(100, 100, 200, 200, fill='lightgray', border='black')
```

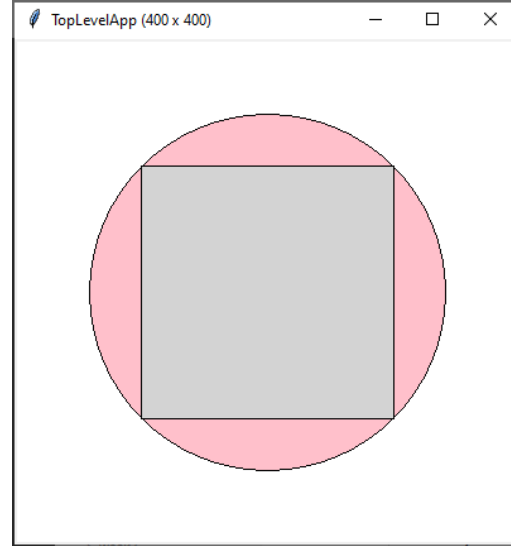
A.



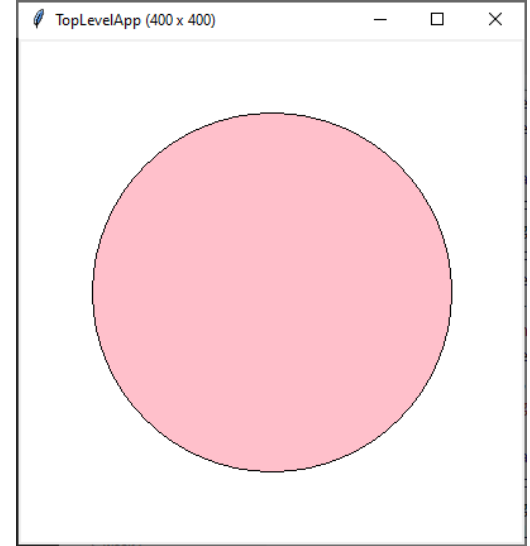
B.



C.



D.



E. I have no idea

Tip: Create Sandbox Cheatsheets for Yourself!

3.3.4 Labels

Label Alignment

This next example shows all the values you can use for alignment:

```
1 from cmu_graphics import *
2
3 def redrawAll(app):
4     drawRect(50, 100, 300, 200, fill='bisque')
5     drawLabel('Left-Top', 50, 100, align='left', size=16)
6     drawLabel('Left', 50, 200, align='left', size=16)
7     drawLabel('Left-Bottom', 50, 300, align='left', size=16)
8
9     drawLabel('Top', 200, 100, align='center', size=16)
10    drawLabel('Center', 200, 200, align='center', size=16)
11    drawLabel('Bottom', 200, 300, align='center', size=16)
12
13    drawLabel('Right-Top', 350, 100, align='right', size=16)
14    drawLabel('Right', 350, 200, align='right', size=16)
15    drawLabel('Right-Bottom', 350, 300, align='right', size=16)
16
17 def main():
18     runApp()
19
20 main()
```

align_demo.py

```
1 from cmu_graphics import *
2
3 def redrawAll(app):
4     drawRect(50, 100, 300, 200, fill='bisque')
5     drawLabel('left-top', 50, 100, align='left', size=16)
6     drawLabel('left', 50, 200, align='left', size=16)
7     drawLabel('left-bottom', 50, 300, align='left', size=16)
8
9     drawLabel('top', 200, 100, align='top', size=16)
10    drawLabel('center', 200, 200, align='center', size=16)
11    drawLabel('bottom', 200, 300, align='bottom', size=16)
12
13    drawLabel('right-top', 350, 100, align='right', size=16)
14    drawLabel('right', 350, 200, align='right', size=16)
15    drawLabel('right-bottom', 350, 300, align='right', size=16)
16
17 def main():
18     runApp()
19
20 main()
```



left-top top right-top

left center right

left-bottom bottom right-bottom

Poll 2

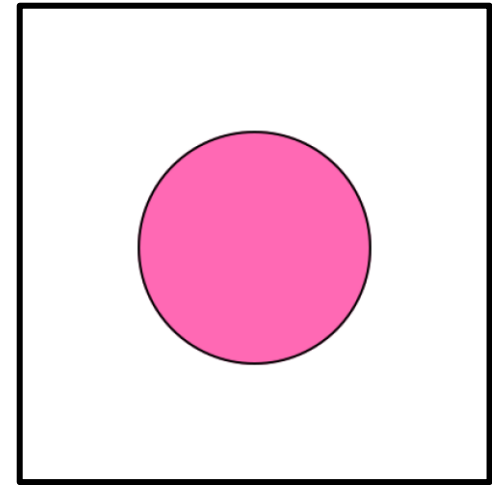
Which of these is best?

Left

```
def drawBubbleA(app):  
    drawOval(200, 200, 200, 200,  
            fill='hotpink', border='black')
```

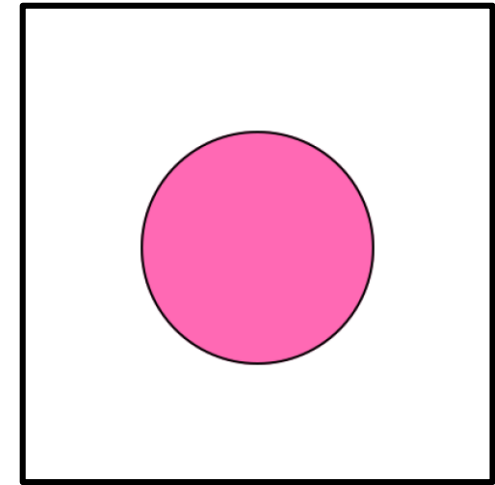
Right

```
def drawBubbleB(app):  
    cx = 200  
    cy = 200  
    width = 200  
    height = 200  
    drawOval(cx, cy, width, height,  
            fill='hotpink', border='black')
```



Poll 3

Which of these is best?



Left

```
def drawBubbleC(app):  
    cx = app.width/2  
    cy = app.height/2  
    d = app.width/2  
    drawOval(cx, cy, d, d,  
             fill='hotpink', border='black')
```

Right

```
def drawBubbleB(app):  
    cx = 200  
    cy = 200  
    width = 200  
    height = 200  
    drawOval(cx, cy, width, height,  
             fill='hotpink', border='black')
```

Poll 4

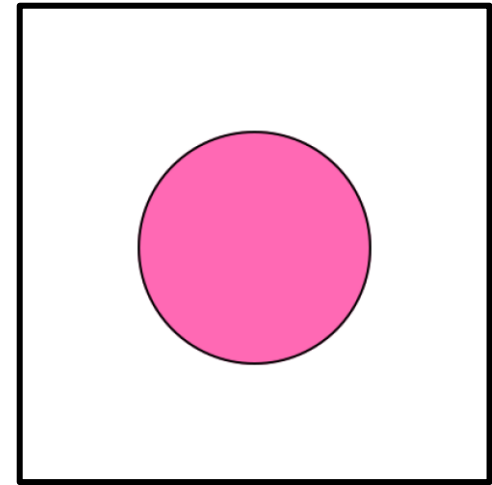
Which of these is best?

Left

```
def drawBubbleC(app):  
    cx = app.width/2  
    cy = app.height/2  
    d = app.width/2  
    drawOval(cx, cy, d, d,  
             fill='hotpink', border='black')
```

Right

```
def drawBubbleD(app):  
    cx = app.width/2  
    cy = app.height/2  
    d = min(app.width/2, app.height/2)  
    drawOval(cx, cy, d, d,  
             fill='hotpink', border='black')
```



Exercise: Bigger Smaller

Guided exercise in notes

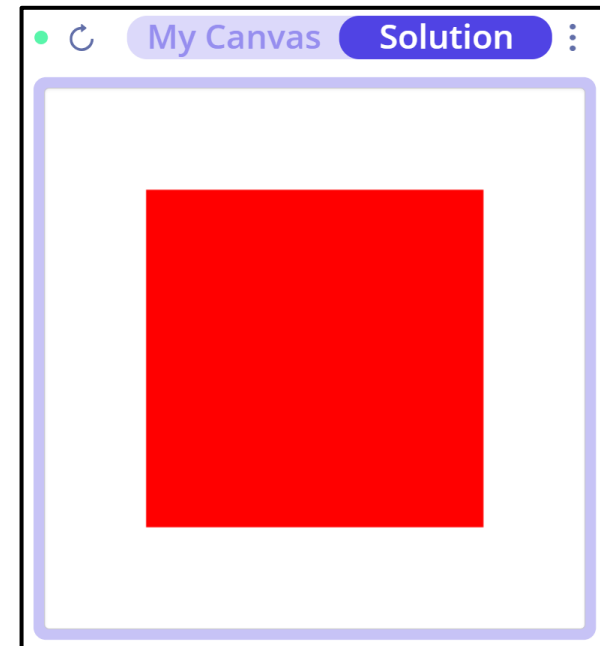
Poll 5

Exercise: Bigger Smaller

(This was not explicitly part of the pre-reading, I just want to know how much I'm repeating things from the notes)

What happens when I click on the white space outside the red square? Select all that apply

- A. Nothing (Hint: it isn't this one. Do NOT select this one)
- B. Square gets bigger
- C. Square gets smaller
- D. Square turns green
- E. Square turns pink



Rectangles, Ovals, Lines, and Labels

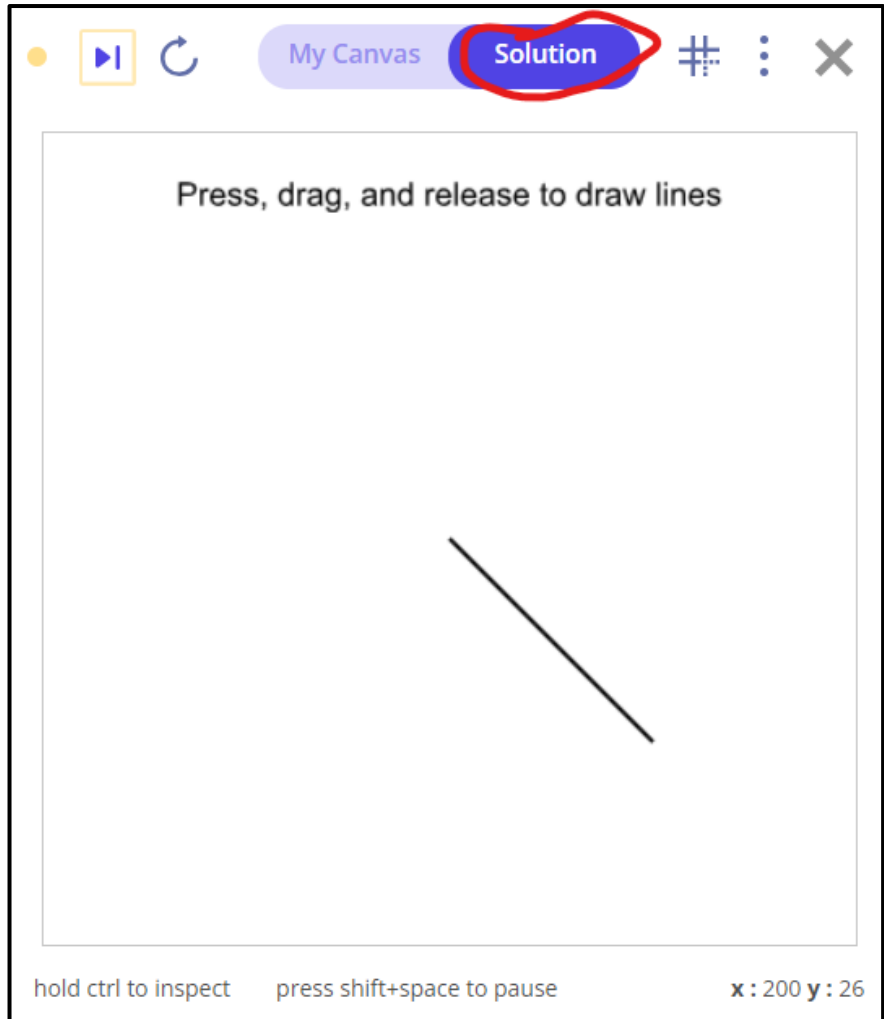
```
drawRect(x0, y0, width, height, fill='black', opacity=100,  
border=None, borderWidth=2, rotateAngle=0)
```

```
drawRect(x?, y?, width, height, fill='black', opacity=100,  
border=None, borderWidth=2, rotateAngle=0,  
align='???' )
```

```
drawRect(centerX, centerY, width, height, fill='black', opacity=100,  
border=None, borderWidth=2, rotateAngle=0,  
align='center' )
```

CS Academy Autograder

CS Academy Autograder with Animations



CS Academy Autograder with Animations

The screenshot displays the CS Academy Autograder interface. At the top, there are navigation links: "Back to note", "Sandbox", "Resources", "Docs + Colors", "My Exercises", "Teacher Portal", a settings gear, and a user profile icon. Below these is the title "3.4.1.1 checkpoint 1" and three buttons: "Stop", "Check", and "Run". A toolbar contains "Sample Solution", "Reset", "Code History", and "Saved".

The main area is a code editor with Python code. A red popup window is overlaid on the code, titled "Try again" and "Autograder Runs: 3". The popup contains the text "Almost...", "You are missing 1 shape(s)", and "Check the highlighted test case".

Below the code editor is a "Console" tab with a "Test Cases" sub-tab. A red circle highlights the "Test Cases" tab, and a red arrow points from it to the "Standard Tests" section. The "Standard Tests" section shows a test case for "Test 5":

```
# Test 5
app = runApp()
onMousePress(app, 200, 200)
onMouseRelease(app, 300, 300) # draw a line from (200,200) to (300,300)
```

ignored. We will consider a line not long enough if the distance between its endpoints is less than 10. In that case, instead of drawing the lines, draw the message 'Not long enough!' in the middle of the canvas (so centered at (200, 200)), and with `size=16`.

Important Hints:

- First run the sample solution. Draw a longer line or two. Then, draw very short lines (for example, just press and release the mouse in the same location). Notice that the shorter lines are not drawn, and instead the message 'Not long enough!' is drawn. Also, note that the 'Not long enough!' message is only drawn until the next mouse press occurs.
- You will need to add a helper function to compute the distance using the distance...

Events

Another good reference for the sandbox

3.5.1 Key Presses and Releases

Key Names

```
1 from cmu_graphics import *
2
3 def onAppStart(app):
4     app.message = 'Press keys to see their names'
5
6 def redrawAll(app):
7     drawLabel(app.message)
8
9 def onKeyPress(app, key):
10    app.message = f'You
11
12 def main():
13     runApp()
14
15 main()
```

The screenshot shows a code editor window titled "keys.py" with a toolbar containing "Stop", "CPCS Mode" (a toggle switch), and "Run" buttons. The code in the editor is as follows:

```
1 from cmu_graphics import *
2
3 def onAppStart(app):
4     app.message = 'Press keys to see their names'
5
6 def redrawAll(app):
7     drawLabel(app.message, 200, 200, size=16, bold=True)
8
9 def onKeyPress(app, key):
10    app.message = f'You pressed: {repr(key)}'
11
12 def main():
13     runApp()
14
15 main()
```

To the right of the code editor is a graphical window displaying the output of the program. The window contains the text "You pressed: 'backspace'". At the bottom of the window, it says "hold ctrl to inspect" and "x : 350 y : 106".

App Skeleton

Tip: Super convenient for sandbox

Note: Next quiz we'll provide function names

Important: Don't call event functions or redrawAll!

```
appSkeleton.py
1 from cmu_graphics import *
2
3 ### Controller
4
5 def onAppStart(app):
6     app.paused = True
7     # app.stepsPerSecond = 30
8
9 def onStep(app):
10     if not app.paused:
11         takeStep(app)
12
13 def takeStep(app):
14     pass
15
16 def onKeyPress(app, key):
17     if key == 's' and not app.paused:
18         takeStep(app)
19
20     elif key == 'p':
21         app.paused = not app.paused
22
```

```
23 def onKeyRelease(app, key):
24     pass
25
26 def onKeyHold(app, key):
27     pass
28
29 def onMousePress(app, mouseX, mouseY):
30     pass
31
32 def onMouseRelease(app, mouseX, mouseY):
33     pass
34
35 def onMouseDrag(app, mouseX, mouseY):
36     pass
37
38 def onMouseMove(app, mouseX, mouseY):
39     pass
40
41 def onResize(app):
42     pass
43
```

```
44 ### View
45
46 def redrawAll(app):
47     pass
48
49 ### Main
50
51 def main():
52     runApp()
53
54 main()
55
```

Debugging:

Even more debugging options with graphics

- Good old print debugging still works

```
def onStart(app):  
    print('onAppStart')  
    app.text = ""  
  
def onKeyPress(app, key):  
    print(f'onKeyPress: {key} ')  
    app.text += key  
  
def redrawAll(app):  
    print('REDRAW ALL')  
    drawLabel(app.text, app.width/2,  
              app.height/2, size=50)
```

- Can also create a temporary debug label: `drawLabel(app.debugText, ...)`

Poll 6

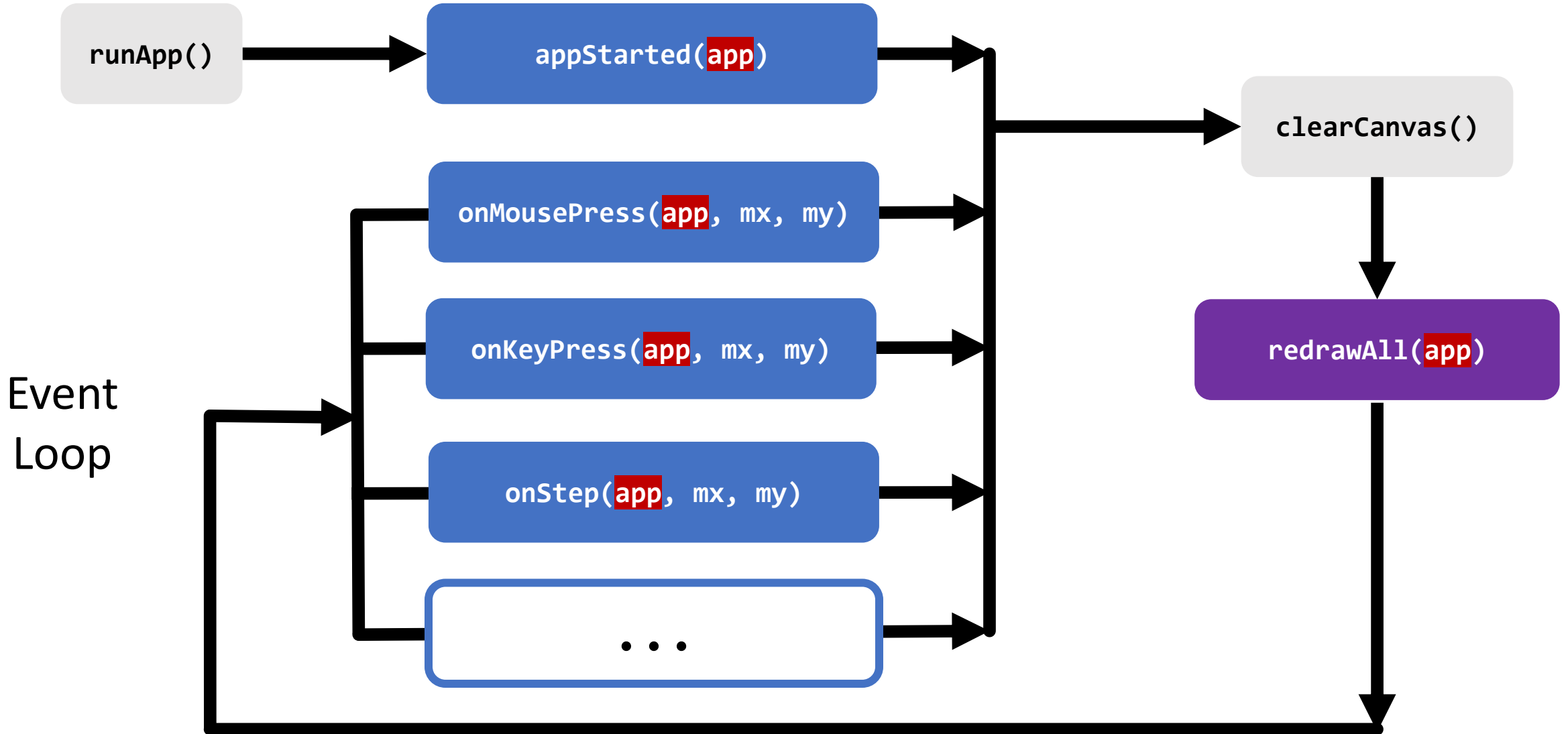
How many times will REDRAW ALL print after I start the app and press the letters A and then Q?

- A. 0
- B. 1
- C. 2
- D. 3
- E. 4
- F. 5
- G. A lot!
- H. I have no idea

```
def onStart(app):  
    print('onAppStart')  
    app.text = ''  
  
def onKeyPress(app, key):  
    print(f'onKeyPress: {key}')  
    app.text += key  
  
def redrawAll(app):  
    print('REDRAW ALL')  
    drawLabel(app.text, app.width/2,  
              app.height/2, size=50)
```

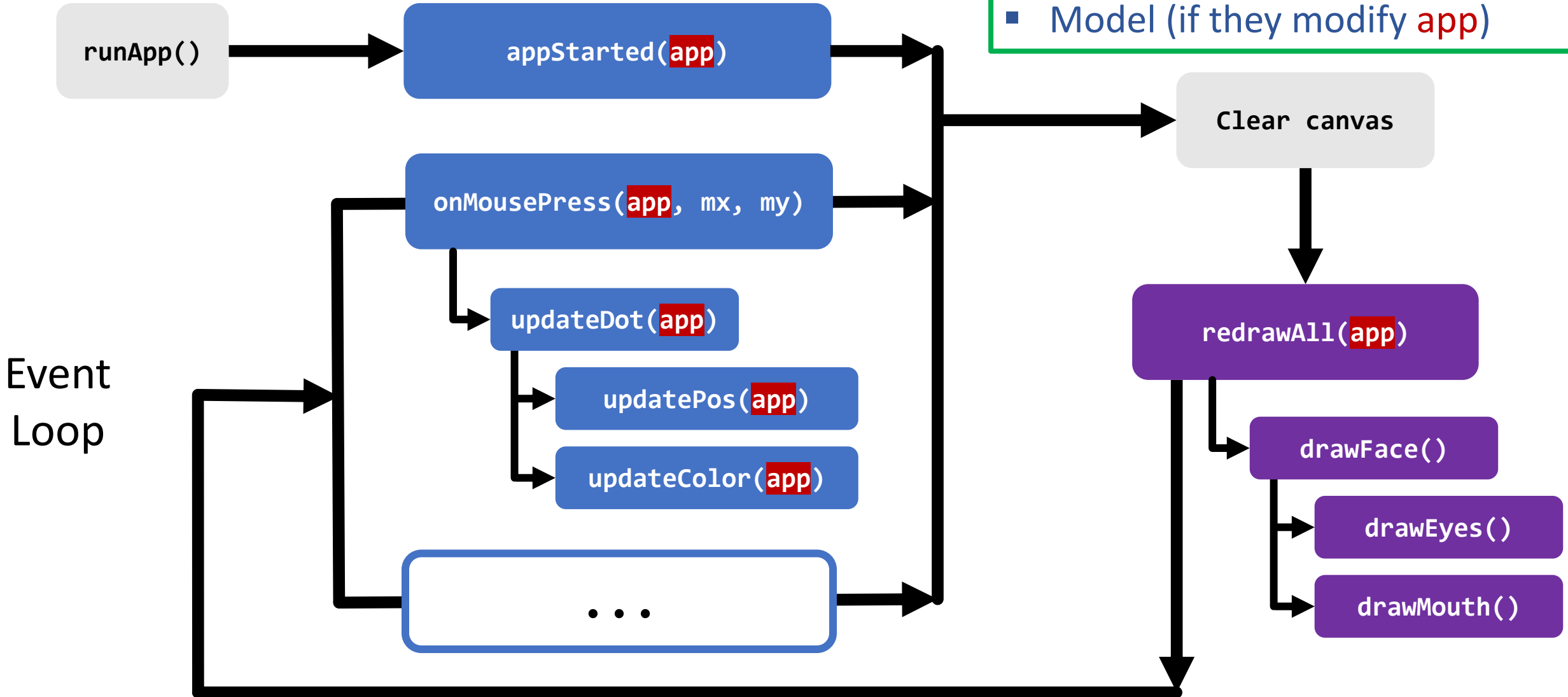
Model-View-Controller

MVC (Model-View-Controller)



MVC (Model-View-Controller)

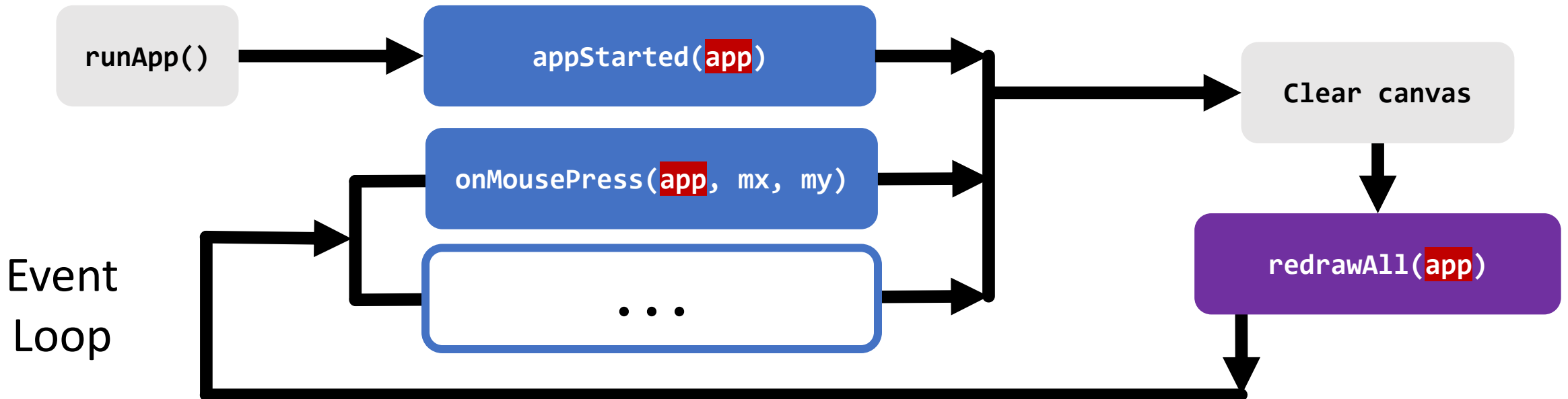
Important!
Helper functions are part of:
▪ View (if they draw anything)
▪ Model (if they modify **app**)



MVC (Model-View-Controller)

Rules

- **View** cannot change the **Model** (`app`)
- **Controller** cannot change the **View** (`redrawAll` or `draw***`)



Poll 7

Which of the following are true?

Select all that apply

- A. `onKeyPress` and `onKeyPress` can run at the exact same time
- B. `redrawAll` and `onKeyPress` can run at the exact same time
- C. Python crashes if we return a value in `onKeyPress`
- D. None of the above
- E. I have no idea

```
def onKeyPress(app, key):  
    app.text += key  
    return key
```




Animation

Examples:

Bouncing DVD Logo

Pong

Debugging: Pause Animation

Even more debugging options with graphics

(but animations can make it more challenging)

```
def onAppStart(app):  
    app.paused = True  
  
def onStep(app):  
    if not app.paused:  
        takeStep(app)  
  
def takeStep(app):  
    # Update app for one step  
    pass
```

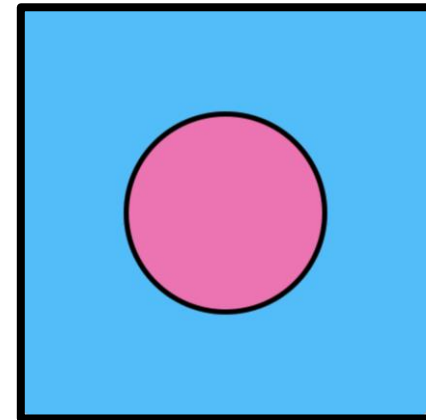
```
def onKeyPress(app, key):  
    if key == 's' and app.paused:  
        takeStep(app)  
    elif key == 'p':  
        # Flip paused stats  
        app.paused = not app.paused
```

Debugging: Pause Animation

Even more debugging options with graphics

(but animations can make it more challenging)

```
def onStart(app):  
    app.ballX = app.width/2  
    app.ballY = app.height/2  
    app.ballR = 0.25*app.width  
  
    app.dX = app.ballR  
  
def takeStep(app):  
    app.ballX += app.dX  
  
def redrawAll(app):  
    drawCircle(app.ballX, app.ballY,  
               app.ballR, fill=app.ballColor,  
               border='black', borderWidth=5)
```

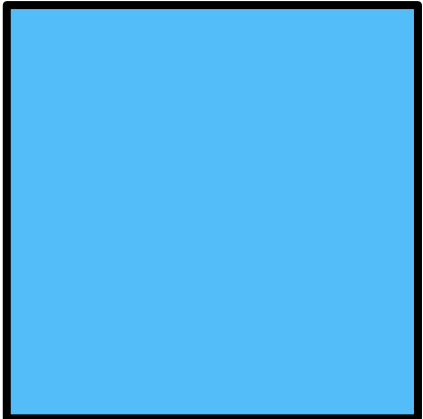


Poll 8

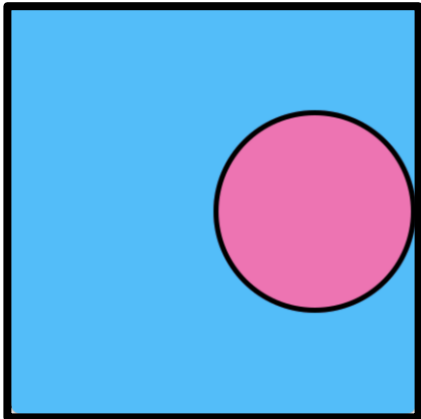
```
def onStart(app):  
    ...  
    app.dX = app.ballR  
  
def onKeyPress(app, key):  
    app.ballX += app.dX  
  
def reDrawAll(app):  
    drawCircle(app.ballX, app.ballY, app.ballR)
```

What happens after we press 's'?

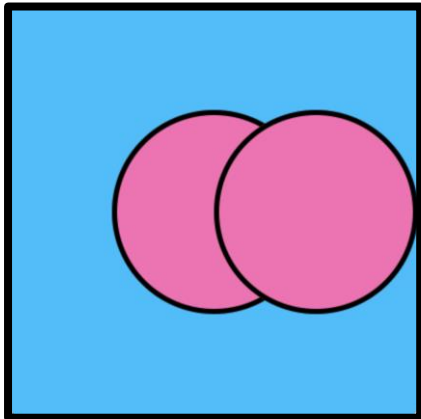
A)



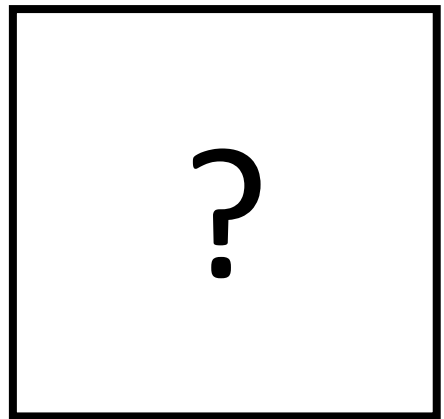
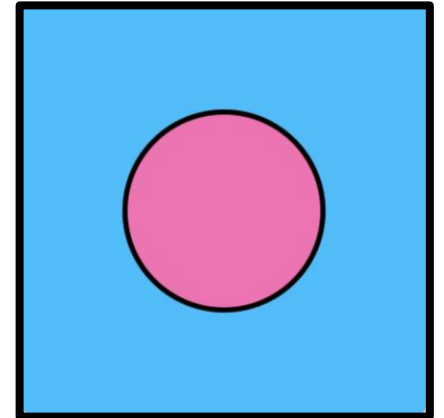
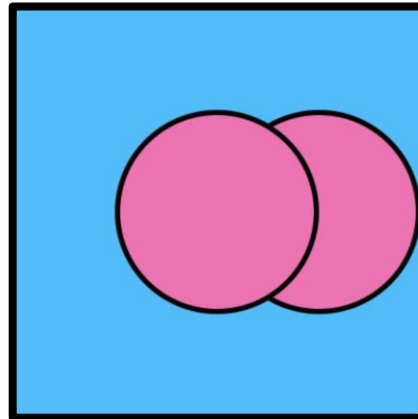
B)



C)

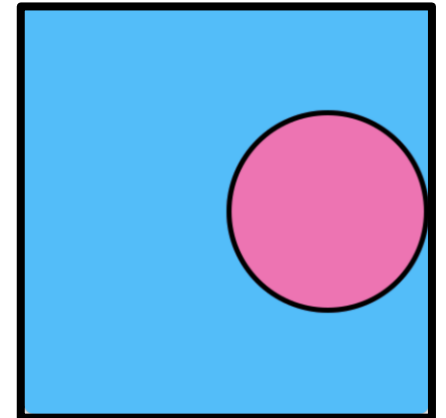
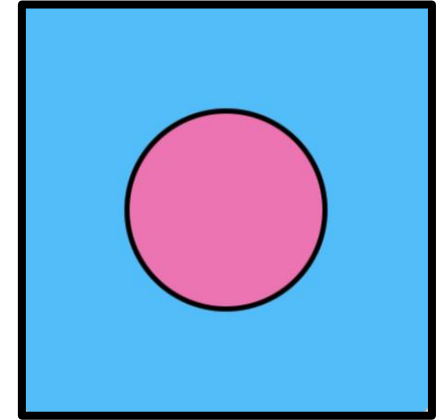


D)



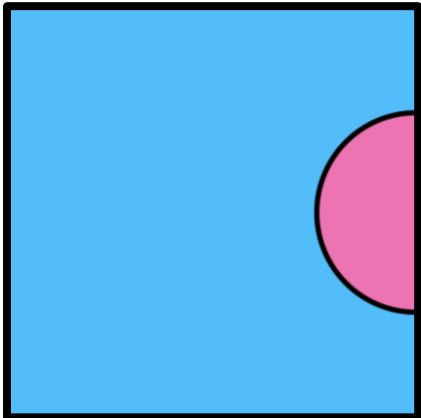
Poll 9

```
def onStart(app):  
    ...  
    app.dX = app.ballR  
  
def takeStep(app):  
    app.ballX += app.dX  
  
def reDrawAll(app):  
    drawCircle(app.ballX, app.ballY, app.ballR)
```

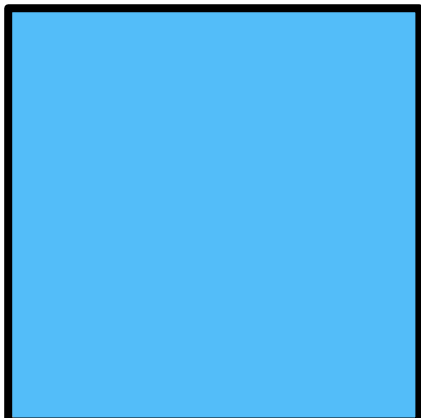


After pressing 's' once, what happens after we press 's' again?

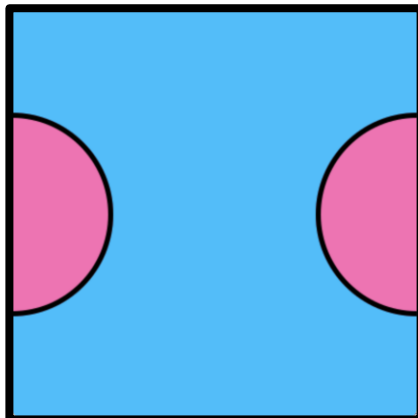
A)



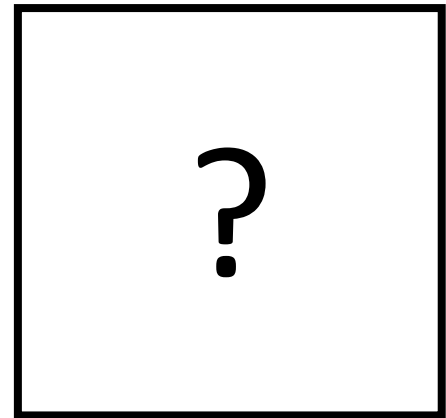
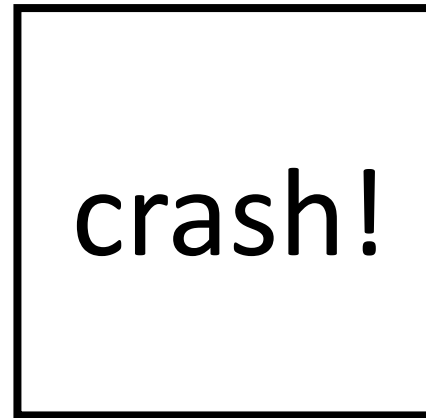
B)



C)



D)



App Skeleton

Tip: Super convenient for sandbox

Note: Next quiz we'll provide function names

```
appSkeleton.py
1 from cmu_graphics import *
2
3 ### Controller
4
5 def onAppStart(app):
6     app.paused = True
7     # app.stepsPerSecond = 30
8
9 def onStep(app):
10     if not app.paused:
11         takeStep(app)
12
13 def takeStep(app):
14     pass
15
16 def onKeyPress(app, key):
17     if key == 's' and not app.paused:
18         takeStep(app)
19
20     elif key == 'p':
21         app.paused = not app.paused
22
```

```
23 def onKeyRelease(app, key):
24     pass
25
26 def onKeyHold(app, key):
27     pass
28
29 def onMousePress(app, mouseX, mouseY):
30     pass
31
32 def onMouseRelease(app, mouseX, mouseY):
33     pass
34
35 def onMouseDrag(app, mouseX, mouseY):
36     pass
37
38 def onMouseMove(app, mouseX, mouseY):
39     pass
40
41 def onResize(app):
42     pass
43
```

```
44 ### View
45
46 def redrawAll(app):
47     pass
48
49 ### Main
50
51 def main():
52     runApp()
53
54 main()
55
```