Solidify it all + Python Main

Revisit animation + complete the python main function picture

Frankie Cerkvenik, CS106A, 2023

Stanford | ENGINEERING
Computer Science

Housekeeping

- Breakout is released and due Sunday, July 23
- Midterm information is released on the website! Midterm is July 26th, 5-7pm in NVIDIA Auditorium
- If you have accommodations, you should receive an email about your exam time/location soon
- If you are SCPD, you should have nominated your exam proctor through SCPD

Today

- **Recap Animation**
  - **Animation loop**
  - **Bouncing ball mechanics**

- Graphics odds and ends
  - Demo a few more graphics functions
  - cleanup_circles

- Python main - how to process input from command line

The animation loop

```
DELAY = 1 / 120

def main():
  # setup

  while True:
    # update world
    canvas.update()

    time.sleep(DELAY)# pause before updating again
```

The animation loop

```
DELAY = 1 / 120

def main():
  # setup - make all the variables you need

  while True:
    # update world
    canvas.update()

    time.sleep(DELAY)# pause before updating again
```

Stanford | ENGINEERING
Computer Science

The animation loop

```
DELAY = 1 / 120

def main():
  # setup

  while True:
    # update world
    canvas.update()

    time.sleep(DELAY)# pause before updating again
```

- The animation loop is like a loop over “frames”
- During one iteration the canvas will look one way.
- On the next loop, it will look slightly different

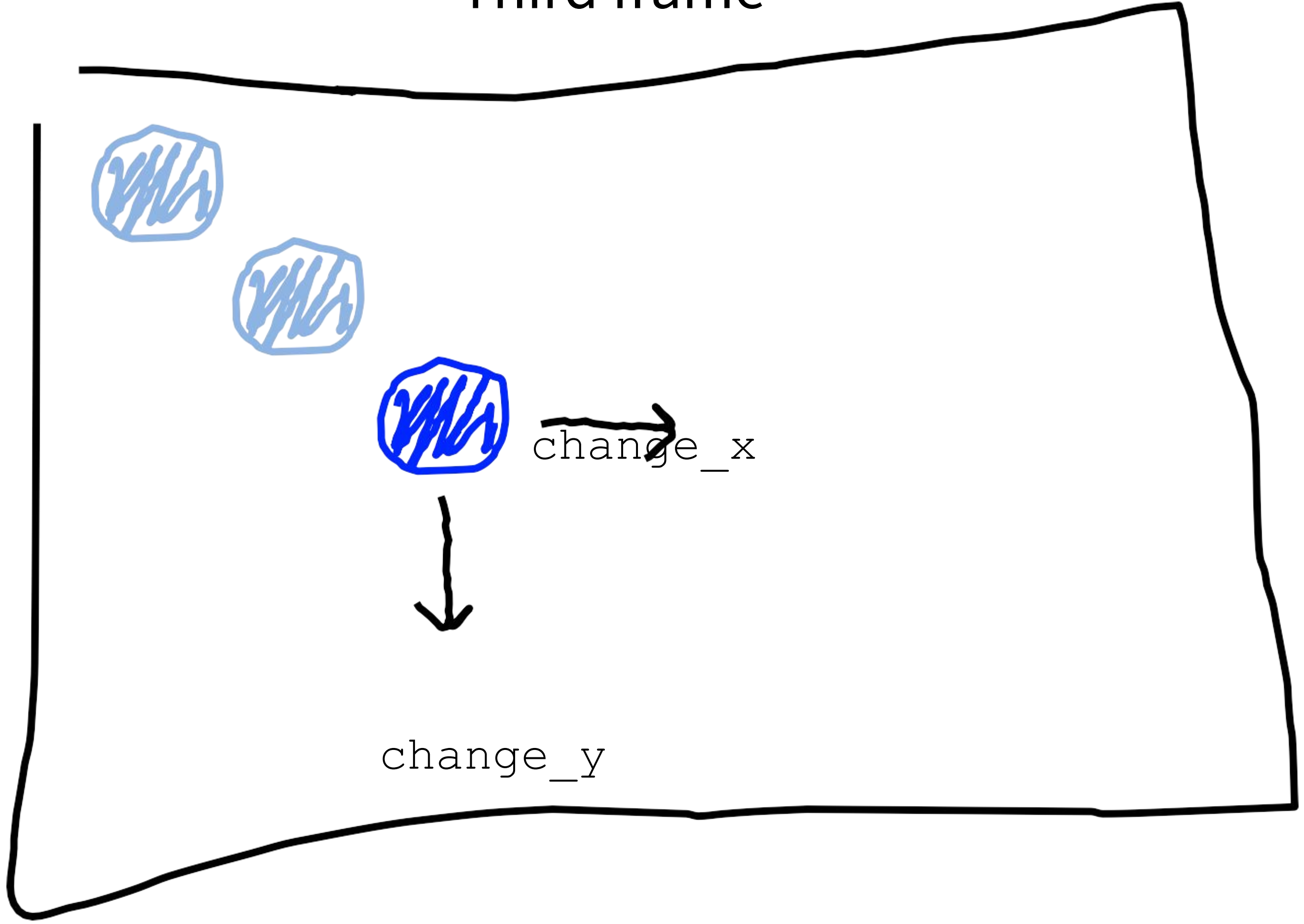
The animation loop

```
DELAY = 1 / 120

def main():
  # setup

  while True:
    # update world
    canvas.update()

    time.sleep(DELAY)# pause before updating again
```

- Pause for a fraction of a second so the user can see the update
- `DELAY` is like your "frame rate"
- Smaller `DELAY` + Smaller update to canvas = higher res animation

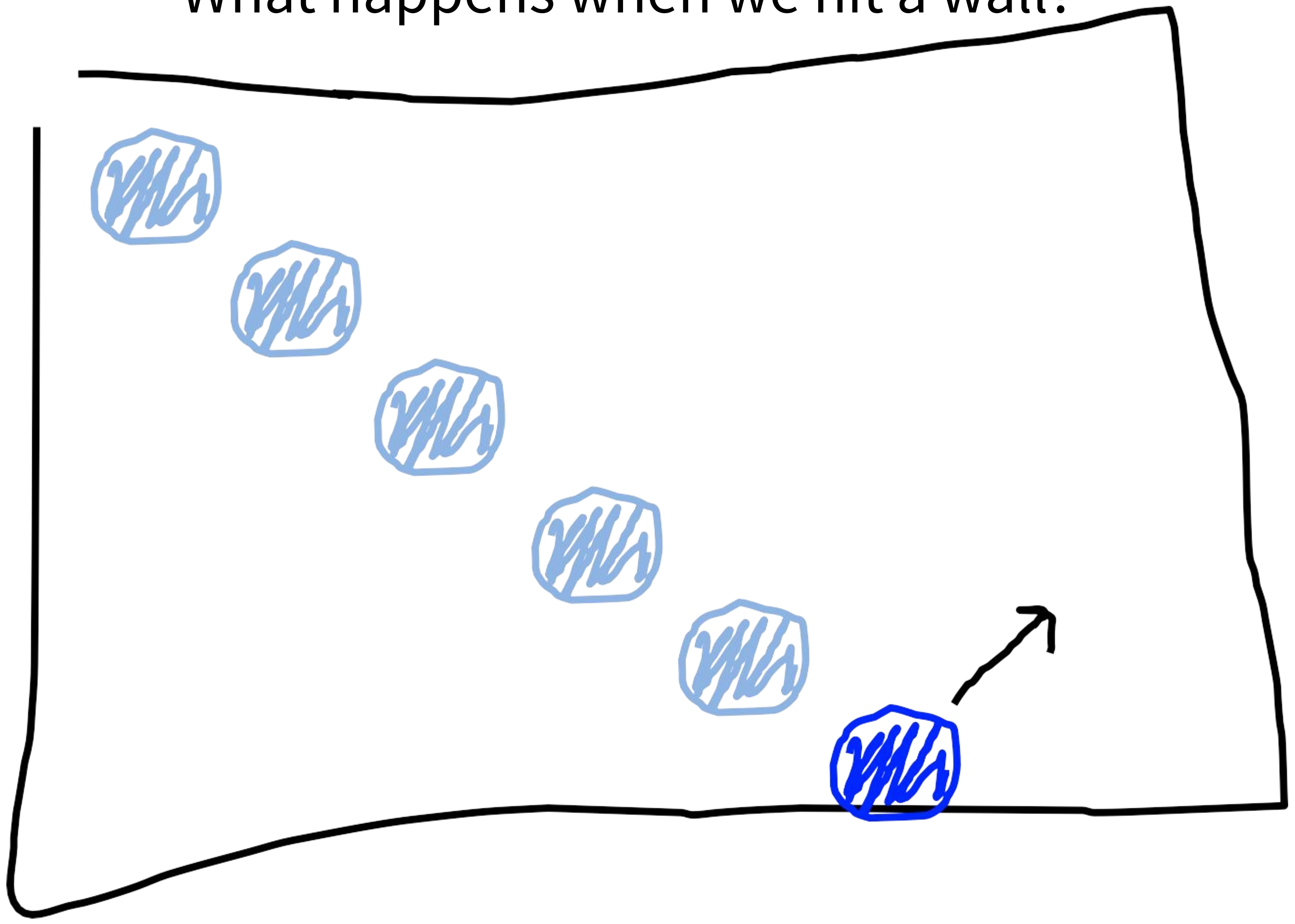
Bouncing Ball

Third frame



Bouncing Ball

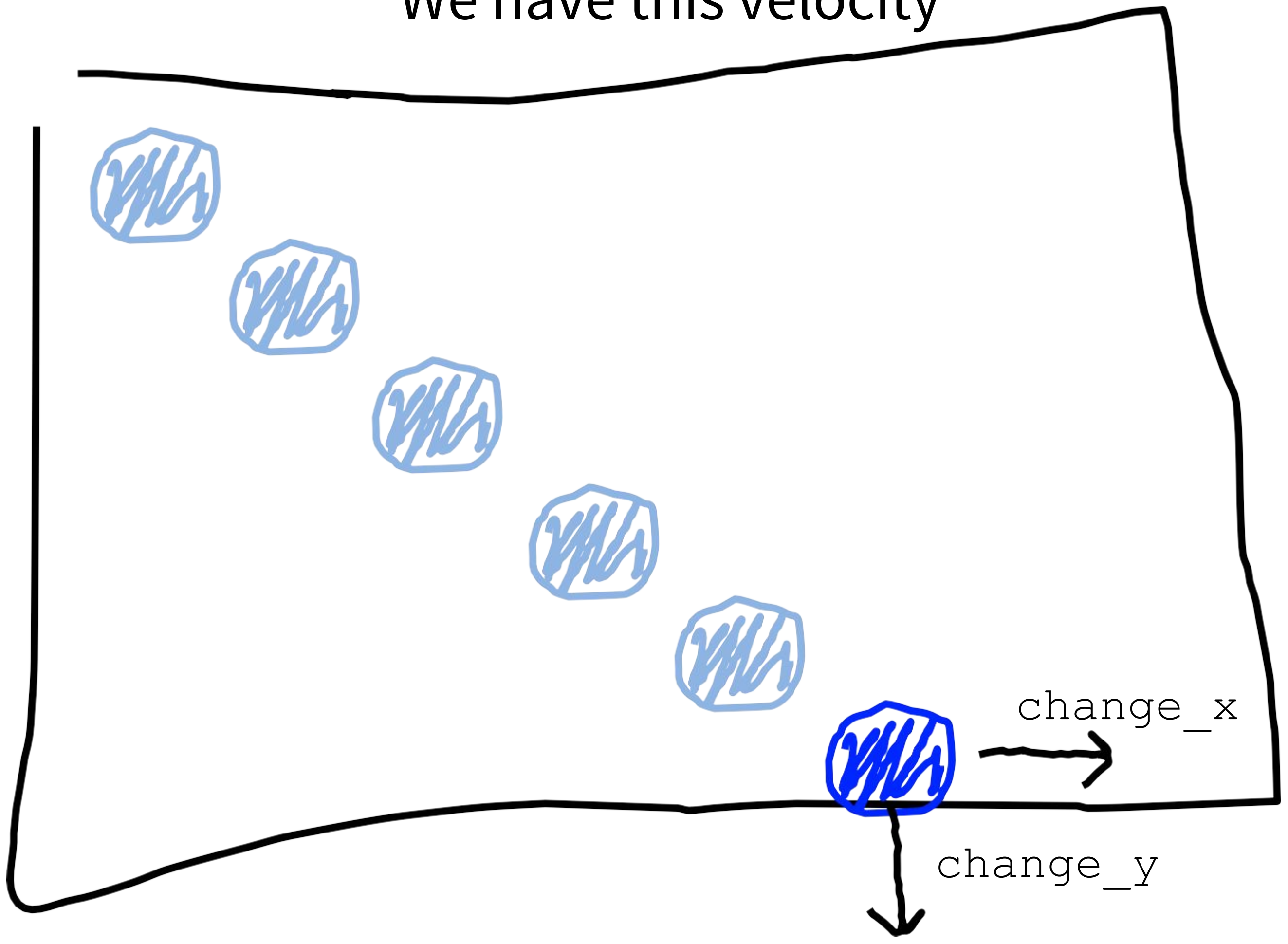
What happens when we hit a wall?



Stanford | ENGINEERING
Computer Science

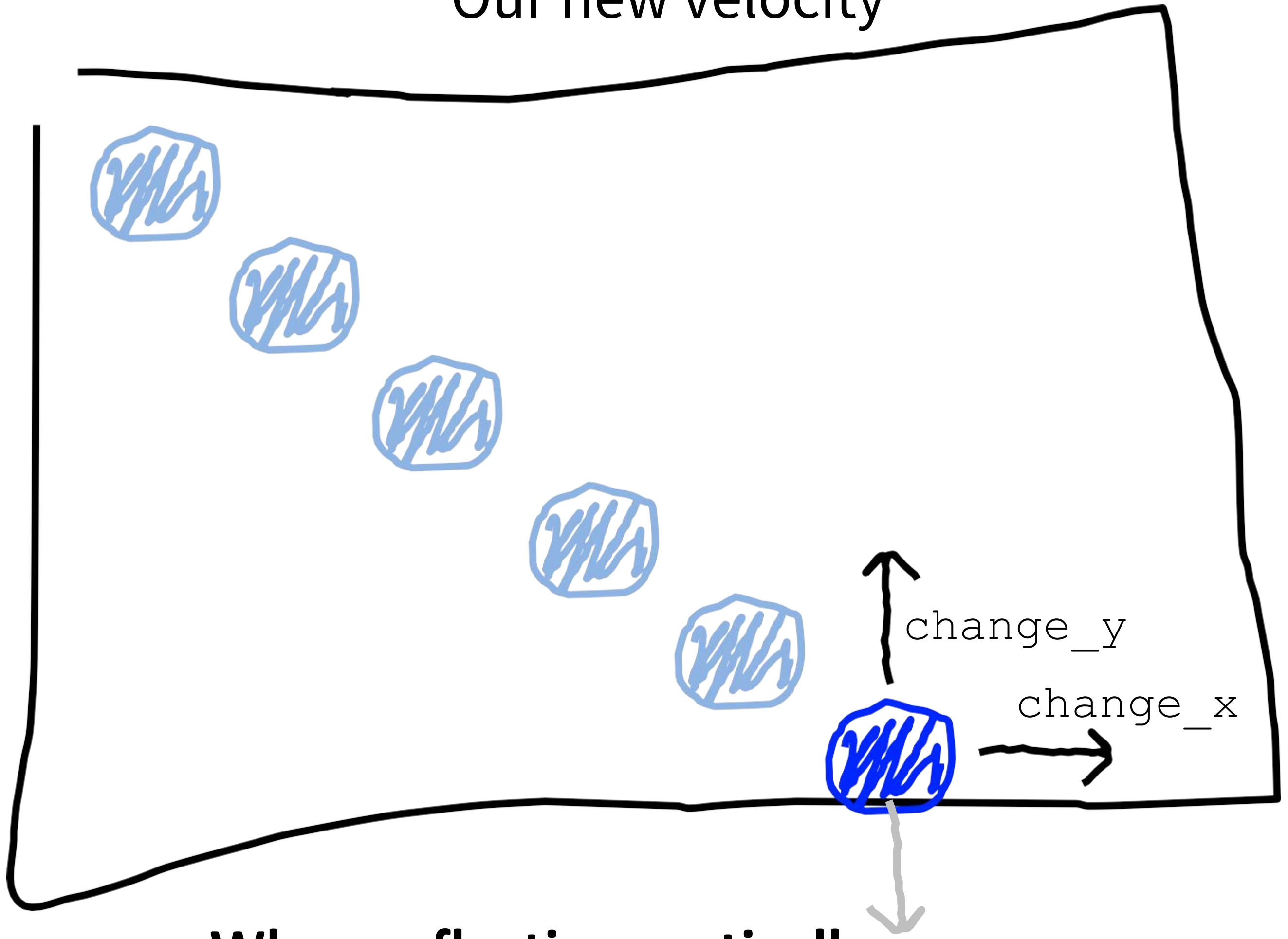
Bouncing Ball

We have this velocity



Bouncing Ball

Our new velocity

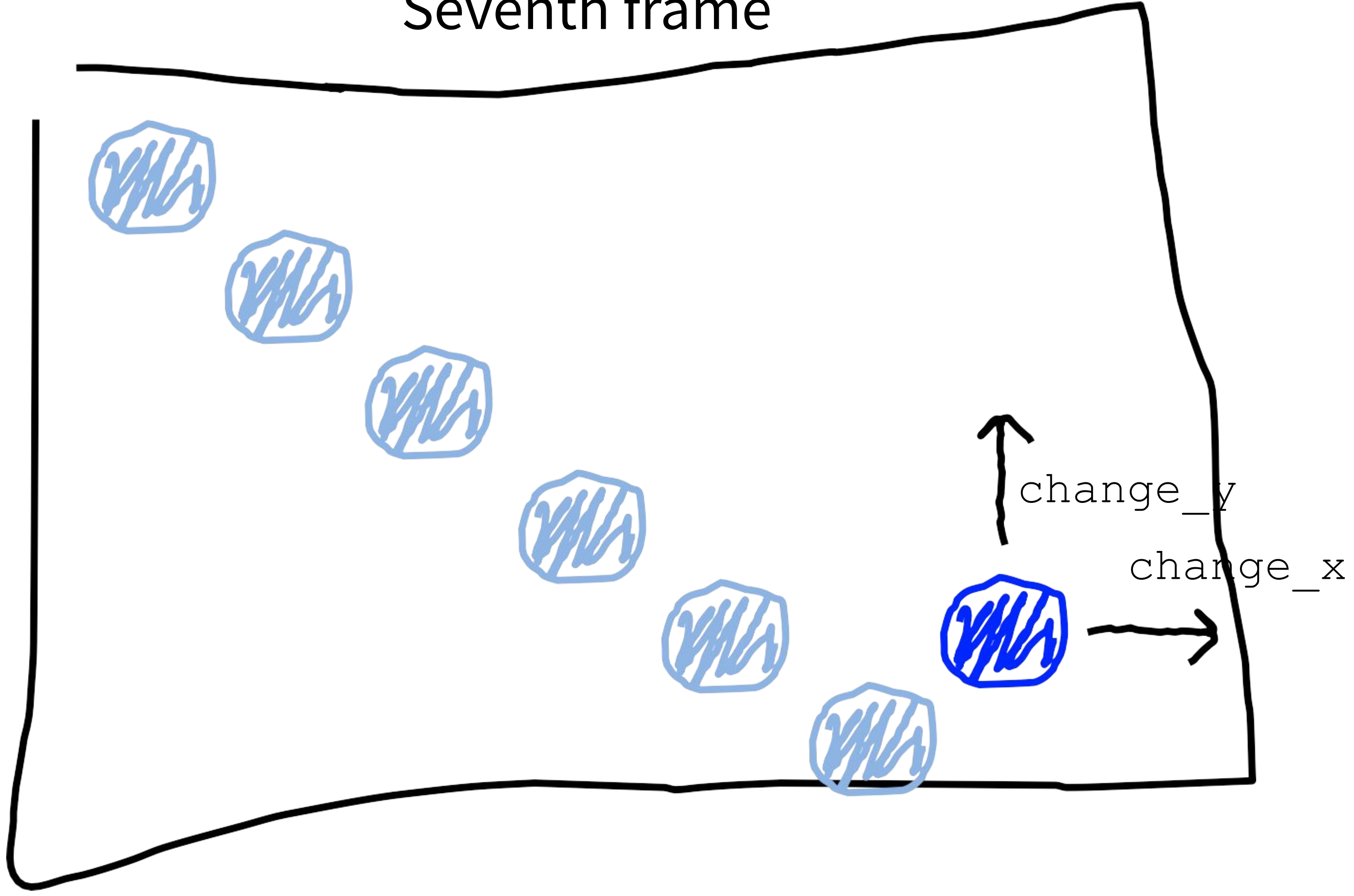


When reflecting vertically:

`change_y = -change_y`

Bouncing Ball

Seventh frame



Bouncing Ball General Idea

```
def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'DVD')
    # create "dvd"
    ball = canvas.create_oval(...)
    # start with this initial velocity
    change_x = 1
    change_y = 1
    while True:
        canvas.move(ball, change_x, change_y)
        if # we hit the top or bottom:
            change_y = -1 * change_y
        elif #we hit the left or right:
            change_x = -1 * change_x
        canvas.update()
        time.sleep(DELAY)
```

Review dvd_screen_saver_soln on your own!

Add in “winner winner” code if dvd hits the corner

Today

- ~~Recap Animation~~
    - ~~Animation loop~~
    - ~~Bouncing ball mechanics~~

- **Graphics odds and ends**
    - **Demo a few more graphics functions**
    - **cleanup_circles**

- Python main - how to process input from command line

Stanford | ENGINEERING
Computer Science

More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()
```

More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)
```

More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)

# move shape by a given change_x and change_y
canvas.move(shape, change_x, change_y)
```

More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)

# move shape by a given change_x and change_y
canvas.move(shape, change_x, change_y)

# get the coordinates of a shape
top_y = canvas.get_top_y(shape)
left_x = canvas.get_left_x(shape)
coord_list = canvas.coords(shape)
```

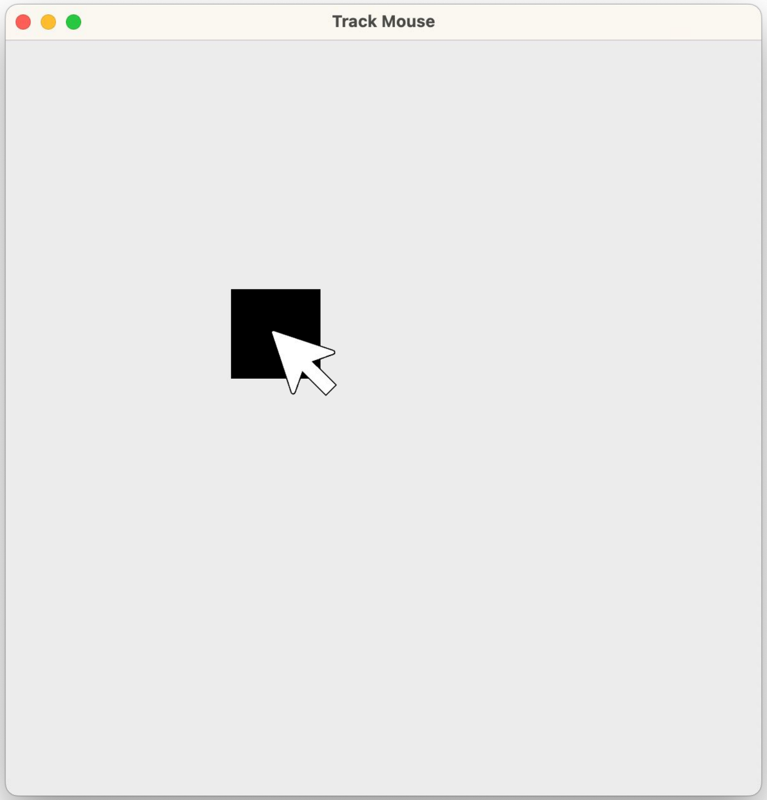
More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)

# move shape by a given change_x and change_y
canvas.move(shape, change_x, change_y)

# get the coordinates of a shape
top_y = canvas.get_top_y(shape)
left_x = canvas.get_left_x(shape)
coord_list = canvas.coords(shape)

# return a list of elements in a rectangle area
results = canvas.find_overlapping(x1, y1, x2, y2)
```

More Graphics Functions Reference

```
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()

# move shape to some new coordinates
canvas.moveto(shape, new_x, new_y)

# move shape by a given change_x and change_y
canvas.move(shape, change_x, change_y)

# get the coordinates of a shape
top_y = canvas.get_top_y(shape)
left_x = canvas.get_left_x(shape)
coord_list = canvas.coords(shape)

# return a list of elements in a rectangle area
results = canvas.find_overlapping(x1, y1, x2, y2)

# wait for a click
canvas.wait_for_click()
```

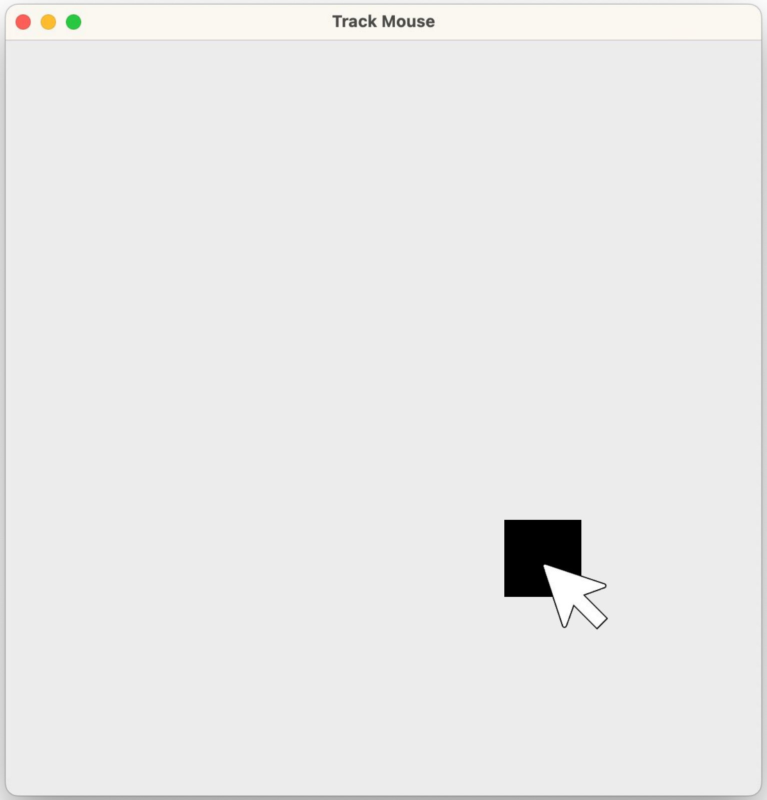
Tracking



Frankie Cerkvenik, CS106A, 2023

Stanford | ENGINEERING
Computer Science

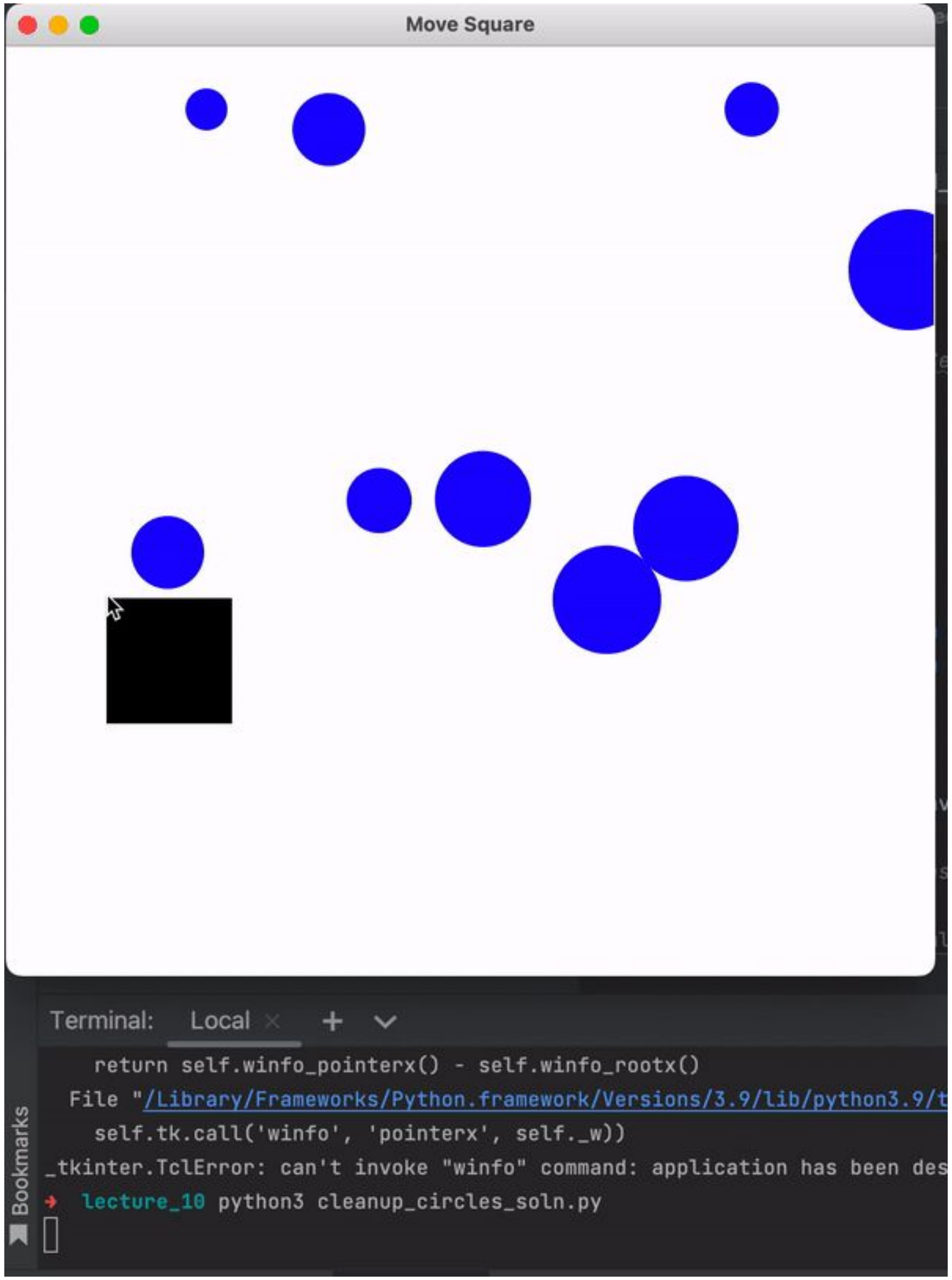
Tracking



Stanford | ENGINEERING
Computer Science

Cleanup circles

- Write a program that:
  - Fills a Canvas with 10 circles with random size and position
  - Allows the user to move a square around with their mouse
  - When the square touches a circle, the circle is removed
  - When all circles are gone, prints “Winner” to the terminal
  - Decompose making the circles and removing a circle



```
    return self.winfo_pointerx() - self.winfo_rootx()
  File "/Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/t
    self.tk.call('winfo', 'pointerx', self._w))
_tkinter.TclError: can't invoke "winfo" command: application has been des
➜  lecture_10 python3 cleanup_circles_soln.py
```

Goal

Stanford ENGINEERING
Computer Science

cleanup_circles.py

To Pycharm!

Today

- ~~Recap Animation~~
    - ~~Animation loop~~
    - ~~Bouncing ball mechanics~~

- ~~Graphics odds and ends~~
    - ~~Demo a few more graphics functions~~
    - ~~cleanup_circles~~

- **Python main - how to process input from command line**

Python main

- The **`main`** function is where a program starts

- When I run **`python3 example.py`**, it will run the code in the **`main`** function - including any function calls

example.py

```
def func():
  print("first line of func!")

def main():
  print("first line of pgm!")
  my_helper()
```

Terminal:

```
$ python3 example.py
```

Stanford | ENGINEERING
Computer Science

Python main

- The **`main`** function is where a program starts

- When I run **`python3 example.py`**, it will run the code in the **`main`** function - including any function calls

example.py

```
def func():
  print("first line of func!")

def main():
  print("first line of pgm!")
  my_helper()
```

Terminal:

```
$ python3 example.py
first line of pgm!
first line of func!
```

Arguments

- When I run **`python3 example.py Frankie`**, the string “Frankie” becomes an “argument”
- Arguments don’t do much until we tell main to use them

example.py

```
def func():
  print("first line of func!")

def main():
  print("first line of pgm!")
  my_helper()
```

Terminal:

```
$ python3 example.py Frankie
first line of pgm!
first line of func!
```

Arguments

- We can access a **list** of arguments like so

example.py

```
import sys

def main():
  args = sys.argv[1:]

  print(args)
```

Terminal:

```
$ python3 example.py Frankie 106A DVDs
[Frankie, 106A, DVDs]
```

Aside: List slicing

- sys.argv includes the file name too, which we “slice” off

example.py

```
import sys

def main():
  all_args = sys.argv
  #all_args:[example.py, Frankie, 106A, DVDs]
  args = all_args[1:]
  #args:[Frankie, 106A, DVDs]
```

Terminal:

```
$ python3 example.py Frankie 106A DVDs
```

Aside: List slicing

- In general, **`list[start:end]`** will take the “slice” of the list starting at index start and ending before index end
- If we omit **`start`** or **`end`**, it will treat it as **`0`** or **`len(list)`**, respectively

```
def main():
  list = [5, 6, 7, 8]
  a = list[1:3] # [6, 7]
  b = list[:3]  # [5, 6, 7]
  c = list[0:]  # [5, 6, 7, 8]
```

Stanford | ENGINEERING
Computer Science

Arguments

- By default, all arguments are interpreted as strings

example.py

```
import sys

def main():
  args = sys.argv[1:]

  print(args[0] + 1)
```

Terminal:

```
$ python3 example.py 10
???
```

Stanford | ENGINEERING
Computer Science

Arguments

- By default, all arguments are interpreted as strings

example.py

```
import sys

def main():
  args = sys.argv[1:]

  print(args[0] + 1)
```

Terminal:

```
$ python3 example.py 10
```

Error: Can’t add int and string

Stanford | ENGINEERING
Computer Science

Arguments

- By default, all arguments are interpreted as strings
- To interpret arguments as another type, transform them with **`type(args[i])`**

example.py

```
import sys

def main():
  args = sys.argv[1:]

  print(int(args[0]) + 1)
```

Terminal:

```
$ python3 example.py 10
11
```

Stanford ENGINEERING
Computer Science

Arguments

- By default, a space specifies a new argument

```
import sys

def main():
  args = sys.argv[1:]

  print(args)
  print(len(args))
```

Terminal:

```
$ python3 example.py Frankie loves 106A
[Frankie, loves, 106A]
```

Arguments

- By default, a space specifies a new argument
- Make a multiword argument by using quotes in the command line

```
import sys

def main():
  args = sys.argv[1:]

  print(args)
  print(len(args))
```

Terminal:

```
$ python3 example.py “Frankie loves 106A”
[Frankie loves 106A]
1
```

cleanup_circles_with_input

- Update cleanup_circles so that:
  - The user can specify a **number** of circles they would like to clean up with the first argument
  - The user can specify the color of the circles with the second argument
  - The user can specify text they would like to display on the canvas when they win with the third argument
  - Print an error message if any of these arguments are missing

Cleanup circles

- Write a program that:
  - Fills a Canvas with 10 circles with random size and position
  - Allows the user to move a square around with their mouse
  - When the square touches a circle, the circle is removed
  - When all circles are gone, prints “Winner” to the terminal
  - Decompose making the circles and removing a circle

If time: mess around with dvd_screen_saver

Recap

- The animation loop (particularly bouncing ball loop) will be very helpful for Breakout
- We can make an object move with the mouse by using the **`get_mouse_x/y`** functions and the **`move_to`** function
- We now fully understand the python main function and can interpret command line arguments to our programs!
- Also list slices :)