Animation

Woah woah woah
Housekeeping

- Breakout is released today, due **nxt Sunday (Jul 23) at midnight**. Grace period extends to Monday.

- Midterm is Wednesday July 26th at 5pm, try not to use grace period! More info will be released this weekend.

- Assn 2 (Images) is due tonight (grace period until tomorrow at midnight).
Today

- Recap the graphics library
  - Use the Graphics reference!
  - Put a shape on a canvas

- Introduce Animation
  - You have permission to use a while true loop
  - The sleep function exists!
  - Remember the DVD logo bouncing around the screen? We are making that
**Turing Award Winner**

- Turing Award is like the Nobel Prize in CS
- Professor Pat Hanrahan here at Stanford (CS107E, CS348)
- Founding employee at Pixar
- Wrote RenderMan, won 3 Academy Awards
Sneak peek at you in 75 minutes
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas
CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()
from graphics import Canvas

CANVAS_HEIGHT = 600
CANVAS_WIDTH = 600
SQUARE_SIZE = 20

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'Move Square')
    start_y = CANVAS_HEIGHT / 2 - SQUARE_SIZE / 2
    end_y = start_y + SQUARE_SIZE
    rect = canvas.create_rectangle(0, start_y, SQUARE_SIZE, end_y, fill='black')
    canvas.mainloop()  # need this to display!
You’re now all graphics programmers!

Woot!
Today

- Recap the graphics library
  - Use the Graphics reference!
  - Put a shape on a canvas

- Introduce Animation
  - You have permission to use a while true loop
  - The sleep function exists!
  - Remember the DVD logo bouncing around the screen? We are making that
Move to Center

- Goal: edit the previous program to make the square move towards the center and stop

- Not quite Toy Story, but it’s a start!
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

```python
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
```

Frankie Cerkvenik, CS106A, 2023
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
DELAY = 1 / 120

def main():
    # setup
    canvas = Canvas(CANVAS_WIDTH,CANVAS_HEIGHT)
    rect = canvas.create_rectangle(0,0,100,100)

    while # the square is not past the center:
        # update world - move square right 1 pixel

            canvas.update() # call update each loop
            # pause
            time.sleep(DELAY)
    # keep canvas open after moving
    canvas.mainloop()
MOVE TO CENTER - USE GRAPHICS REFERENCE!

```
DELAY = 1 / 120

def main():
    # setup
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT)
    rect = canvas.create_rectangle(0, 0, 100, 100)

    while # the square is not past the center:
        # update world - move square right 1 pixel
        canvas.move(rect, 1, 0)
        canvas.update()
        # pause
        time.sleep(DELAY)
        # keep canvas open after moving
        canvas.mainloop()
```
Move to center - lets make a helper!

DELAY = 1 / 120

def main():
    # setup
    canvas = Canvas(CANVAS_WIDTH,CANVAS_HEIGHT)
    rect = canvas.create_rectangle(...)

    while not is_past_center(canvas, rect):
        # update world - move square right 1 pixel
        canvas.move(rect, 1, 0)
        canvas.update()
        # pause
        time.sleep(DELAY)
        # keep canvas open after moving
        canvas.mainloop()
Move to center - Pycharm
We are ready
For some nostalgia
Goal
Milestone 1
Bouncing Ball

First frame

Key variable: how much does the ball position change each heartbeat?
Bouncing Ball

First frame

The `move` function takes in a change in x and a change in y.
Bouncing Ball

Second frame

\[ \text{change}_x \]
\[ \text{change}_y \]
Bouncing Ball
Third frame

change_x
change_y
Bouncing Ball
What happens when we hit a wall?
Bouncing Ball

We have this velocity

\[ \text{change}_x \]

\[ \text{change}_y \]
Bouncing Ball

Our new velocity

When reflecting vertically:

\[ change_y = -change_y \]
Bouncing Ball

Seventh frame

change_y
change_x
Bouncing Ball

Eighth frame

change_y
change_x
Bouncing Ball

Ninth frame

change_y
change_x
Bouncing Ball

We want this!
Bouncing Ball

This was our old velocity

change_y
change_x
Bouncing Ball

This is our new velocity

When reflecting horizontally:
\[ \text{change}_x = -\text{change}_x \]
Bouncing Ball

Tenth heartbeat

change_x

change_y
Bouncing Ball General Idea

def main():
    canvas = Canvas(CANVAS_WIDTH, CANVAS_HEIGHT, 'DVD')
    # create "dvd"
    dvd = canvas.create_oval(...)  # start with this initial velocity
    change_x = 1
    change_y = 1
    while True:
        canvas.move(dvd, change_x, change_y)
        if # we hit the top or bottom:
            change_y = -change_y
        elif # we hit the left or right:
            change_x = -change_x
        canvas.update()
        time.sleep(DELAY)
Off to Pycharm!
Milestone 2: Add randomness

- Currently, the ball always follows the same path and bounces in the same way
- To add randomness:
  - Start with `change_x = 1` and `change_y = 2`
  - When we bounce vertically, change the `sign` of the `change_y`, but also randomly change its magnitude to be either 1 or 2
  - To keep the ball a consistent speed, we will need to change `change_x` to be the same sign as it was, but magnitude 1 if `change_y` has magnitude 2 and 2 otherwise
  - Vice versa for horizontal bounces
import random

# get a random number between 1 and 10, inclusive
num = random.randint(0, 10)

def sign(num):
    """
    returns the sign of parameter num
    """
    return num / abs(num)
Milestone 2:

- Decompose a function to find the sign of a number (previous slide)

- Decompose a function to find the value of \texttt{change\_x} given the new value of \texttt{change\_y} and the old value of \texttt{change\_x}, and vice versa

```python
def find_complement(new_change, old_change):
    """
    new_change: The new value of change\_x or _y
    old_change: The old value of the other one
    return: The new value of the other one
    """
```
To Pycharm!
Milestone 3: Make the ball a DVD

- There is a list, DVD_IMGS, of files containing dvd logos of different colors
- Start the logo as the first item in the list use create
  - Use `canvas.create_image_with_size(...)`
- When the logo bounces:
  - Delete the old logo
  - Make a new logo of a different color (next file in the list) exactly where the old one was
  - You will need to keep track of an “index” variable that tells you which logo file in the list to use
- Decompose a function that deletes and makes the new dvd logo
To Pycharm!
Tracking

[Image of a mouse tracking software interface]
# get the x location of the mouse
mouse_x = canvas.get_mouse_x()
# 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)
# 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 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)
# 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)
# 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()
Check out cleanup_circles.py
For some Breakout hints :0