CSc 110, Autumn 2016

Lecture 7: Graphics

Adapted from slides by Marty Stepp and Stuart Reges
Graphical objects

We will draw graphics in Python using a new kind of object:

- **DrawingPanel**: A window on the screen.
  - Not part of Python; provided by the instructor. See class web site.
Named colors

Chart credit Smith.edu
Custom colors

• You can construct custom colors using hex.
  • # followed by six numbers 0 – 9 and letters A – F
    • A is 10, B is 11 and so on
    • #000000 is black
    • #FFFFFF is white
    • Colors get darker as the number gets lower
    • The first two digits are the amount of red, the next two green, the last two blue

```python
panel = DrawingPanel(80, 50, background="#3367D3")
```
Drawing shapes

`panel.canvas.create_line(x1, y1, x2, y2, fill="color")`

line between points \((x1, y1), (x2, y2)\) in color

`panel.canvas.create_oval(x1, y1, x2, y2, outline="color")`

outline largest oval that fits in a box with top-left at \((x1, y1)\) and lower right at \((x2, y2)\) outlined in color

`panel.canvas.create_rectangle(x1, y1, x2, y2, outline="color")`

outline of rectangle with top-left at \((x1, y1)\) and bottom right at \((x2, y2)\) outlined in color

`panel.canvas.create_text(x, y, text="string")`

text centered vertically and horizontally around \((x, y)\)
Coordinate system

• Each \((x, y)\) position is a *pixel* ("picture element").

• \((0, 0)\) is at the window's top-left corner.
  • \(x\) increases rightward and the \(y\) increases downward.

• The rectangle from \((0, 0)\) to \((200, 100)\) looks like this:
Filled in shapes

• To draw a shape with a fill set its `fill` instead of `outline`.

```python
from drawingpanel import *  # so I can use Graphics
def main():
    p = DrawingPanel(150, 70)
    # inner red fill
    p.canvas.create_rectangle(20, 10, 120, 60, fill="red")

• This will automatically fill the shape but give it a black border. To remove the border add `width=0`.
    p.canvas.create_rectangle(20, 10, 120, 60, fill="red", width=0)
```
Superimposing shapes

• When two shapes occupy the same pixels, the last one drawn is seen.

```python
from drawingpanel import *

def main():
    p = DrawingPanel(200, 100, background="light gray")
    p.canvas.create_rectangle(10, 30, 110, 80, fill="black")
    p.canvas.create_oval(20, 70, 40, 90, fill="red", width=0)
    p.canvas.create_oval(80, 70, 100, 90, fill="red", width=0)
    p.canvas.create_rectangle(80, 40, 110, 60, fill="cyan", width=0)
```
Drawing with loops

• The $x1, y1, x2, y2$ expression can contain the loop counter, $i$.

```python
da panel = DrawingPanel(400, 300, background="yellow")
for i in range(1, 11):
    panel.canvas.create_oval (100 + 20 * i, 5 + 20 * i, 
                             150 + 20 * i, 55 + 20 * i 
                             fill="red", width=0)
```

```python
panel = DrawingPanel(250, 220)
for i in range(1, 11):
    panel.canvas.create_oval (30, 5, 30 + 20 * i, 
                             5 + 20 * i, fill="magenta")
```
Loops that begin at 0

- Beginning a loop at 0 and using < can make coordinates easier to compute.

- Example:
  - Draw ten stacked rectangles starting at (20, 20), height 10, width starting at 100 and decreasing by 10 each time:

```python
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.canvas.create_rectangle (20, 20 + 10 * i, 120 - 10 * i, 30 + 10 * i)
```
Drawing w/ loops questions

• Code from previous slide:

```python
panel = DrawingPanel(160, 160)

for i in range(0, 10):
    panel.canvas.create_rectangle(20, 20 + 10 * i,
                                120 - 10 * i, 30 + 10 * i)
```

• Write variations of the above program that draw the figures at right as output.
Drawing w/ loops answers

• Solution #1:
  
  ```python
  panel = DrawingPanel(160, 160)
  for i in range(0, 10):
      panel.canvas.create_rectangle (20 + 10 * i, 20 + 10 * i, 120, 30 + 10 * i)
  ```

• Solution #2:
  
  ```python
  panel = DrawingPanel(160, 160)
  
  for i in range(0, 10):
      panel.canvas.create_rectangle (110 - 10 * i, 20 + 10 * i, 120, 30 + 10 * i)
  ```
Parameterized figures

• Modify the car-drawing function so that it can draw many cars, such as in the following image.
  • Top-left corners: (10, 30), (150, 10)
  • Hint: We must modify our `draw_car` function to accept x/y coordinates as parameters.
def main():
    panel = DrawingPanel(260, 100, background="light gray")
draw_car(panel, 10, 30)
draw_car(panel, 150, 10)

def draw_car(p, x, y):
    p.canvas.create_rectangle(x, y, 100 + x, 50 + y, fill="black")

    p.canvas.create_oval(x + 10, y + 40, x + 30, y + 60, fill="red", width=0)
    p.canvas.create_oval(x + 70, y + 40, x + 90, y + 60, fill="red", width=0)

    p.canvas.create_rectangle(x + 70, y + 10, x + 100, y + 30, fill="cyan",
                               width=0)
Modify `draw_car` to allow the car to be drawn at any size.
- Existing car: size 100. Second car: (150, 10), size 50.

Once you have this working, use a `for` loop with your function to draw a line of cars, like the picture at right.
- Start at (10, 130), each size 40, separated by 50px.
def main():
    panel = DrawingPanel(260, 100, background="light gray")
    draw_car(panel, 10, 30, 100)
    draw_car(panel, 150, 10, 50)
    for i in range(0, 5):
        draw_car(panel, 10 + i * 50, 130, 40);

def draw_car(p, x, y, size):
    p.canvas.create_rectangle(x, y, x + size, y + size / 2, fill="black")

    p.canvas.create_oval(x + size / 10, y + size / 10 * 4, x + size / 10 * 3, y +
                          size / 10 * 6, fill="red", width=0)

    p.canvas.create_oval(x + size / 10 * 7, y + size / 10 * 4, x + size / 10 * 9,
                          y + size / 10 * 6, fill="red", width=0)

    p.canvas.create_rectangle(x + size / 10 * 7, y + size / 10, x + size,
                               y + size / 10 * 3, fill="cyan", width=0)
Animation with sleep

• DrawingPanel's sleep function pauses your program for a given number of milliseconds.

• You can use sleep to create simple animations.

```python
panel = DrawingPanel(250, 200)
for i in range(1, NUM_CIRCLES + 1):
    panel.canvas.create_oval(15 * i, 15 * i, 30 + 15 * i, 30 + 15 * i)
    panel.sleep(500)
```

• Try adding sleep commands to loops in past exercises in this chapter and watch the panel draw itself piece by piece.