CS 1110

Lecture 26: Subclasses in Event-driven Programs

A7

…is out! Get started right away—you need time to ask questions.

Academic integrity

Please be careful: do not share your code or look at other groups' code.

Prelim 2 handback

Exams on front table, in piles by lab section.

Announcements

Final exam makeups

Requests for makeups (including cases of 3 exams in 24 hrs) are due **tonight in CMS**.

No lab next week

There is no new lab assignment for the last week. Use the time to ask questions about A7 or to finish lab 13.

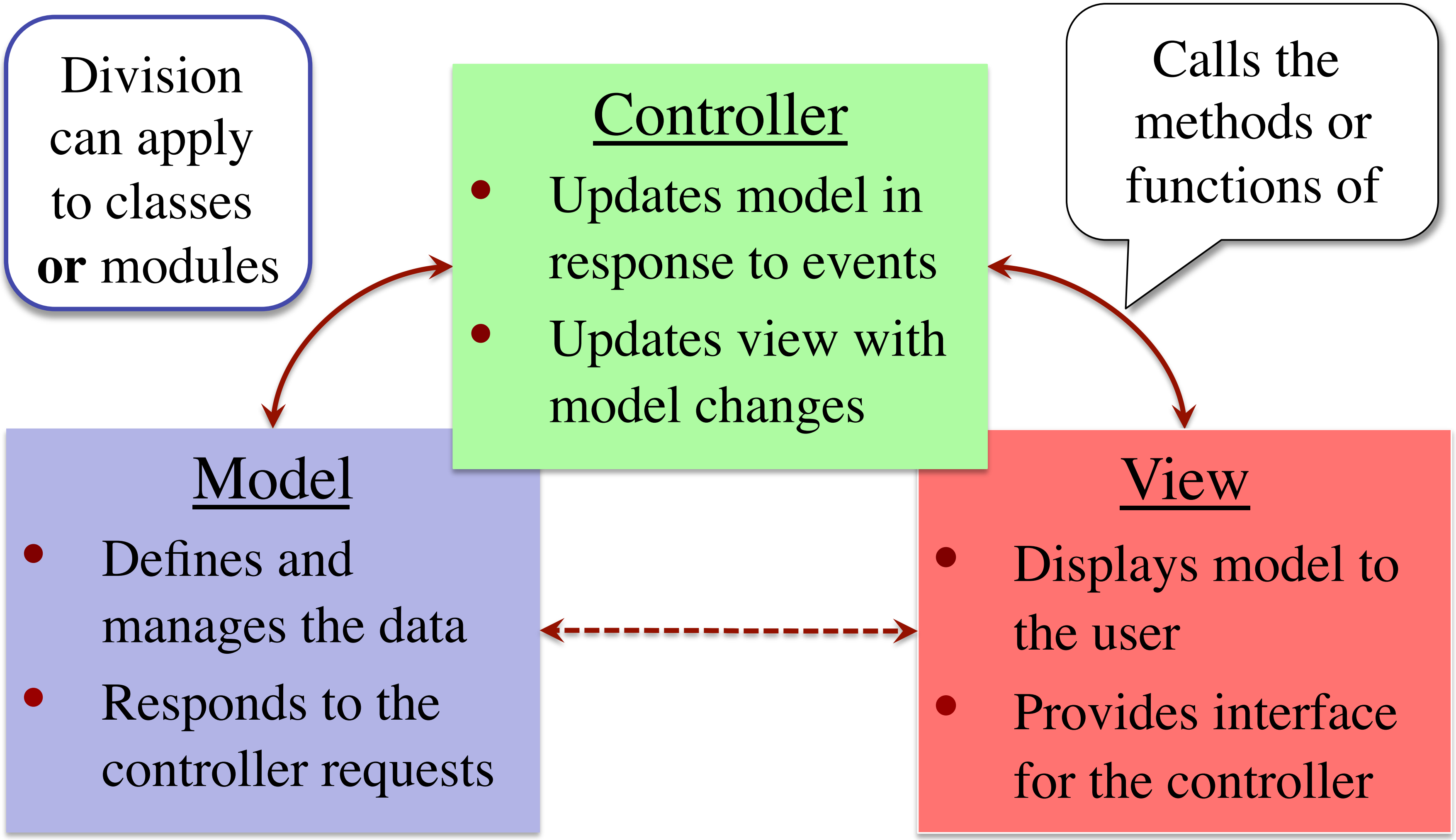
ACSU-W KICKOFF EVENT
Cornell Women in Computing
SHE++
the documentary
directed by two of Stanford's Good Girls Gone Geek, Ayna Agarwal and Ellora Israni

You are invited to celebrate the creation of ACSU-W (Association of Computer Science Undergraduates for Women) with fellow students and Computer Science faculty on Monday April 29 from 7 - 8:30 PM in Upson Lounge. Watch the exclusive screening of a 15 minute documentary by ShePlusPlus and follow with your own empowering messages in our photo and video booth. Dinner will be served.

https://www.facebook.com/CornellWomenInComputing

Slides by D. Gries, L. Lee, S. Marschner, W. White

Model-View-Controller Pattern



Division can apply to classes **or** modules

Controller

- Updates model in response to events
- Updates view with model changes

Calls the methods or functions of

Model

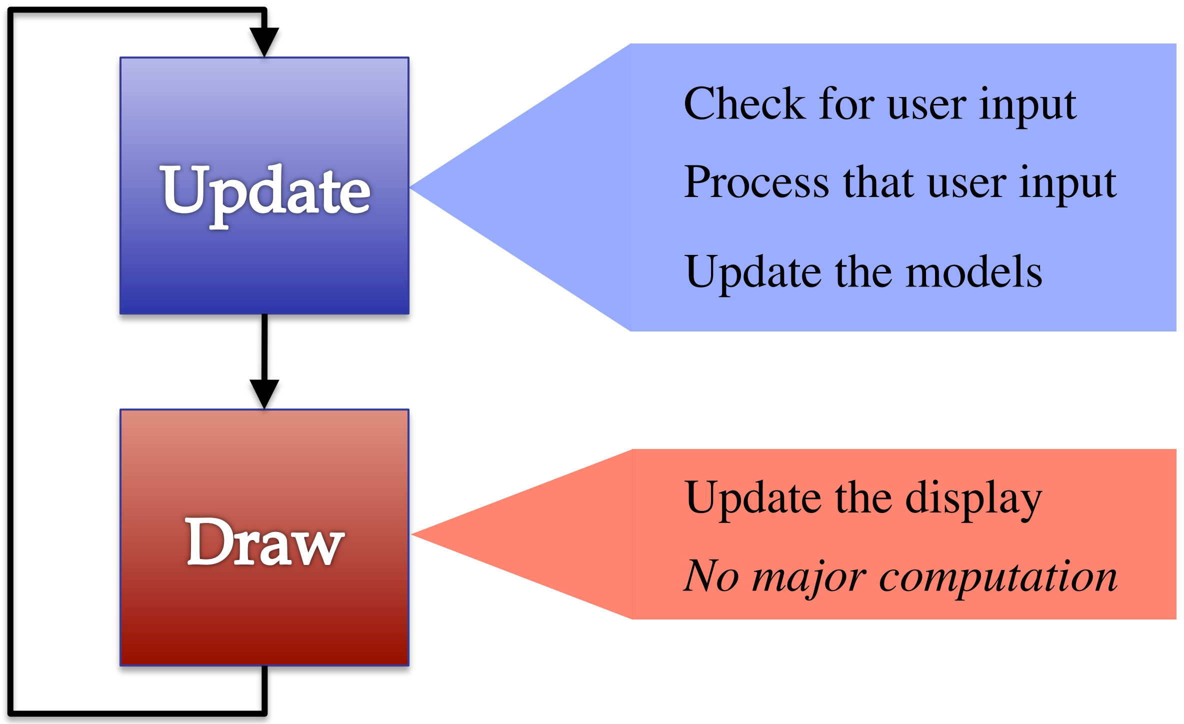
- Defines and manages the data
- Responds to the controller requests

View

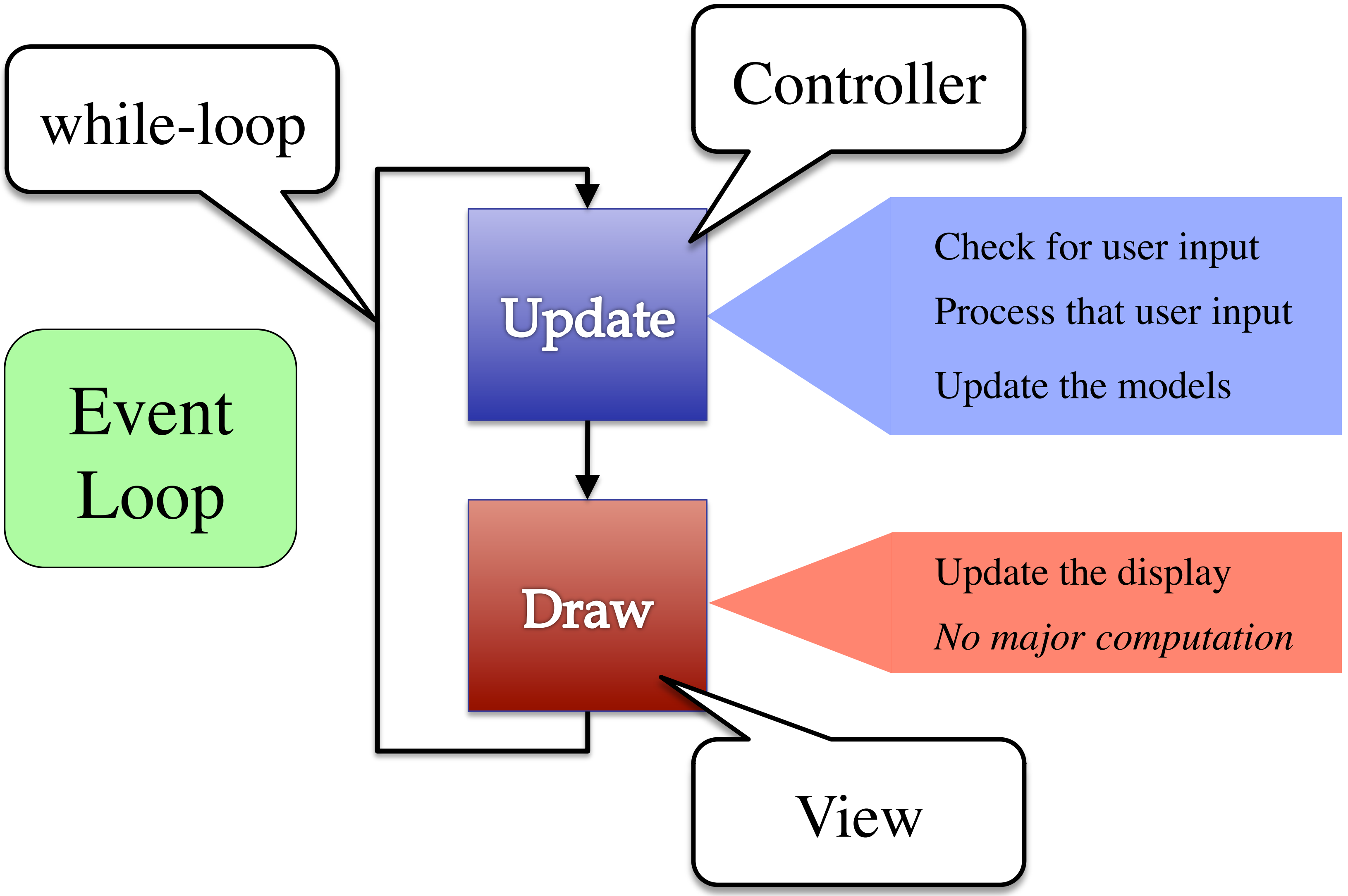
- Displays model to the user
- Provides interface for the controller

A Standard GUI Application

Animates the application, like a movie



A Standard GUI Application



Check for user input

Process that user input

Update the models

Update the display

No major computation

Must We Write this Loop Each Time?

```
while program_is_running:
    # Get information from mouse/keyboard
    # Handled by OS/GUI libraries

    # Your code goes here

    # Draw stuff on the screen
    # Handled by OS/GUI libraries
```

Must We Write this Loop Each Time?

```
while program_is_running:
    # Get information from mouse/keyboard
    # Handled by OS/GUI libraries


    # Your code goes here


    # Draw stuff on the screen
    # Handled by OS/GUI libraries
```

Would like to "plug in" code

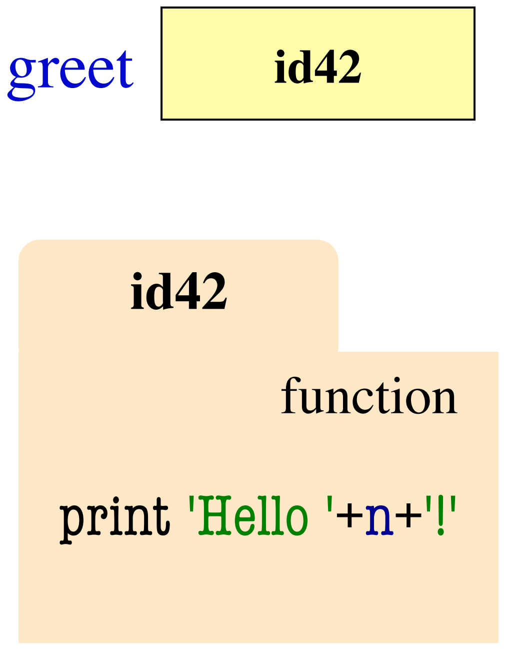
Why do we need to write this each time?

Functions Are Objects

- Calling a function
  - Provide arguments in ()
  - Executes the body
- Passing a function
  - Assign another variable
  - Use the name without ()
- Example:

```
>>> x = greet
>>> x('Walker')
Hello Walker!
```

```
def greet(n):
    print 'Hello '+n+'!'
```



Callback Functions

- **Given**: predefined code that calls some function
  - But function not defined
  - You want to replace it with your function
- You redefine that function
  - By overriding it in a subclass (do this in A7)
  - Or by storing a reference to your function somewhere ("registering" your callback)
  - Works the same either way

```
while program_running:
    # Get input
    # Your code goes here
    callback()
    # Draw
```

See callback.py

Example: Animation

- Callback: animate(…)
  - Called 60x a second
  - Moves back and forth
- Animate is a method
  - Associated with an object
  - Object has changing state
- **Examples** of state
  - Ellipse position
  - Current velocity
  - Current animation step

```
def animate(self,dt):
    """Animate the ellipse back & forth"""
    if self._steps == 0:
        # Initialize
        …
    elif self._steps > ANIMATION_STEPS/2:
        # Move away
        x = self._ellipse.pos[0]
        y = self._ellipse.pos[1]
        self._ellipse.pos = (x+self._vx,y+self._vy)
        self._steps = self._steps - 1
    else:  # Move back
        x = self._ellipse.pos[0]
        y = self._ellipse.pos[1]
        self._ellipse.pos = (x-self._vx,y-self._vy)
        self._steps = self._steps - 1
```

Example: Animation

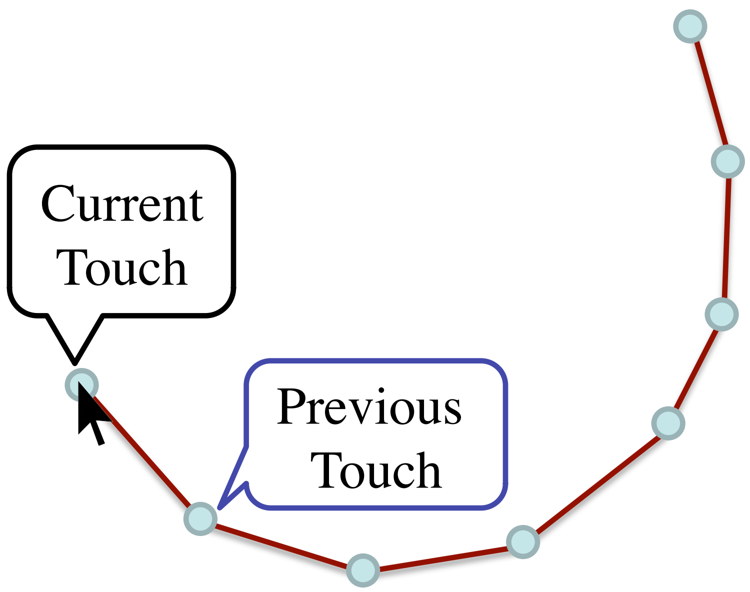
- Callback: animate(…)
  - Called 60x a second
  - Moves back and forth
- Animate is a method
  - Associated with an object
  - Object has changing state
- **Examples** of state
  - Ellipse position
  - Current velocity
  - Current animation step

```
def animate(self,dt):
    """Animate the ellipse back & forth"""
    if self._steps == 0:
        # Initialize
        …
    elif self._steps > ANIMATION_STEPS/2:
        # Move away
        x = self._ellipse.pos[0]
        y = self._ellipse.pos[1]
        self._ellipse.pos = (x+self._vx,y+self._vy)
        self._steps = self._steps - 1
    else:  # Move back
        x = self._ellipse.pos[0]
        y = self._ellipse.pos[1]
        self._ellipse.pos = (x-self._vx,y-self._vy)
        self._steps = self._steps - 1
```

See animate.py

State Across Multiple Callbacks

- Sometimes have more than one callback function
- Example: touch events
  - `on_touch_down`: User presses mouse (or a finger); does not release
  - `on_touch_up`: Releases mouse (or finger)
  - `on_touch_move`: Moves mouse (or finger)
- State needed to track change in touch over time



See `touch.py`

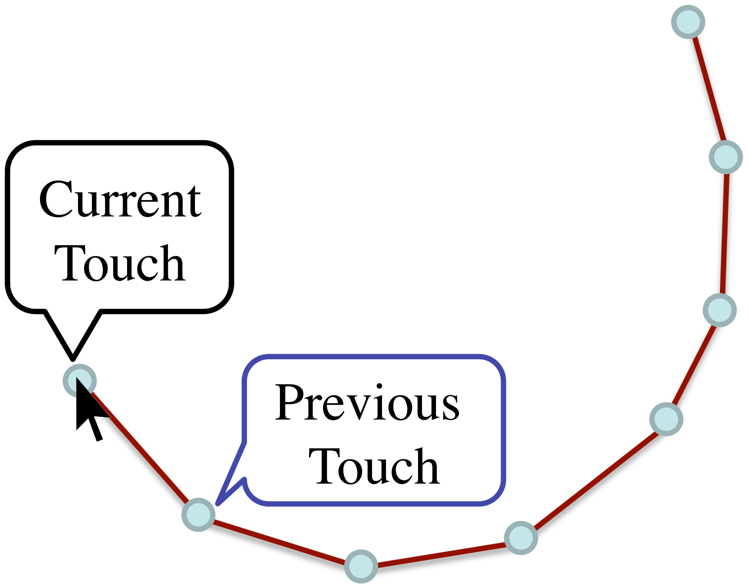
State Across Multiple Callbacks

```
# None or previous touch
_anchor = None

def on_touch_down(self,touch):
    # Track touch state
    self._anchor = (touch.x,touch.y)

 def on_touch_up(self,touch):
    # Nothing to track
    self._anchor = None

 def on_touch_move(self,touch):
    if not self._anchor is None:
        self.drawLine(self._anchor[0], self._anchor[1],
                      touch.x,touch.y,LINE_COLOR)
        self._anchor = (touch.x,touch.y)
```



See touch.py