CS 1110

Lecture 25: Models, Views, Controllers, and Games

Announcements

A5

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

Office/consulting hours

…will be changing for study week. See the webpage for details.

Final exam makeups

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

No lab next week

There is no lab assignment for the last week. Use the time to ask questions about A5.

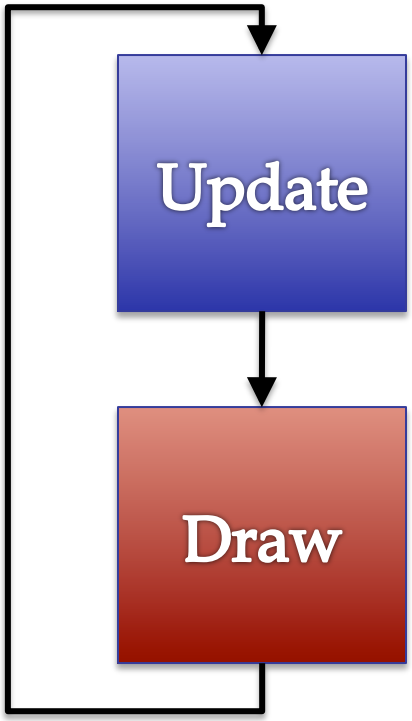
Prelim 2

Exam papers may be picked up in Gates 216 as before. They will not be handed back in lab. Solutions are in CMS. Regrades follow the same procedure as Prelim 1: study solution; write on separate sheet; attach to exam; hand to us **in class by Tuesday May 6**. (See March 13 lecture slides.)

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

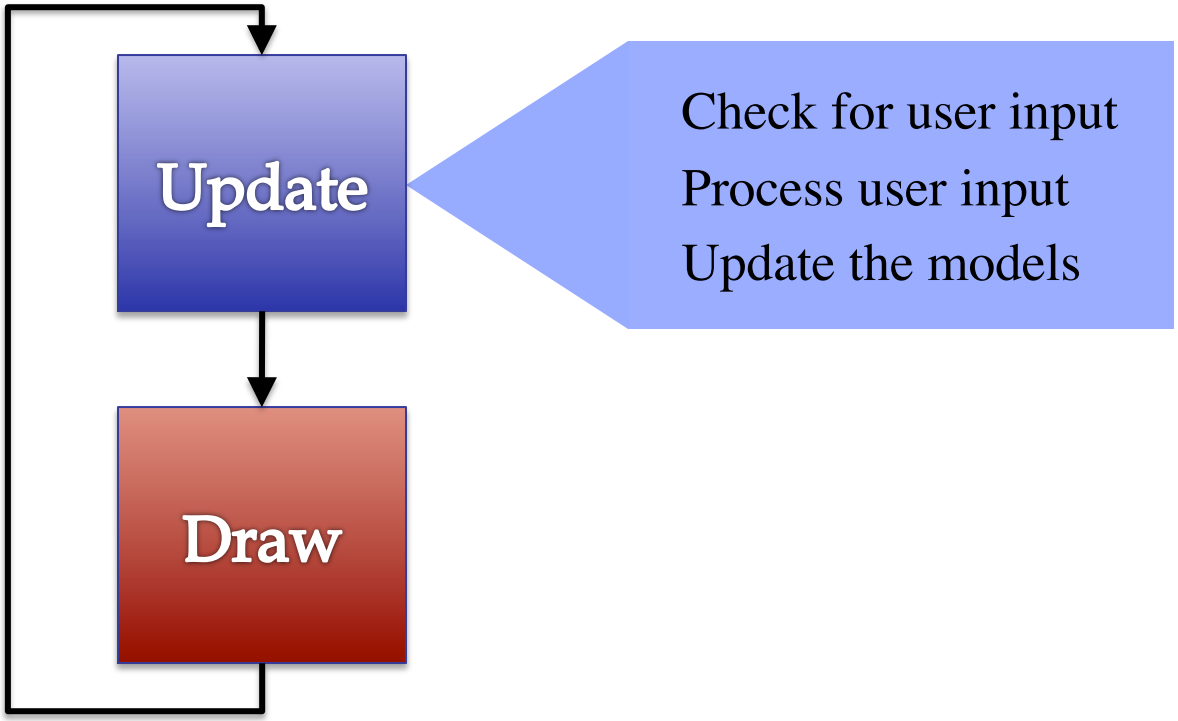
A Standard Game Application

Animates the application, like a movie



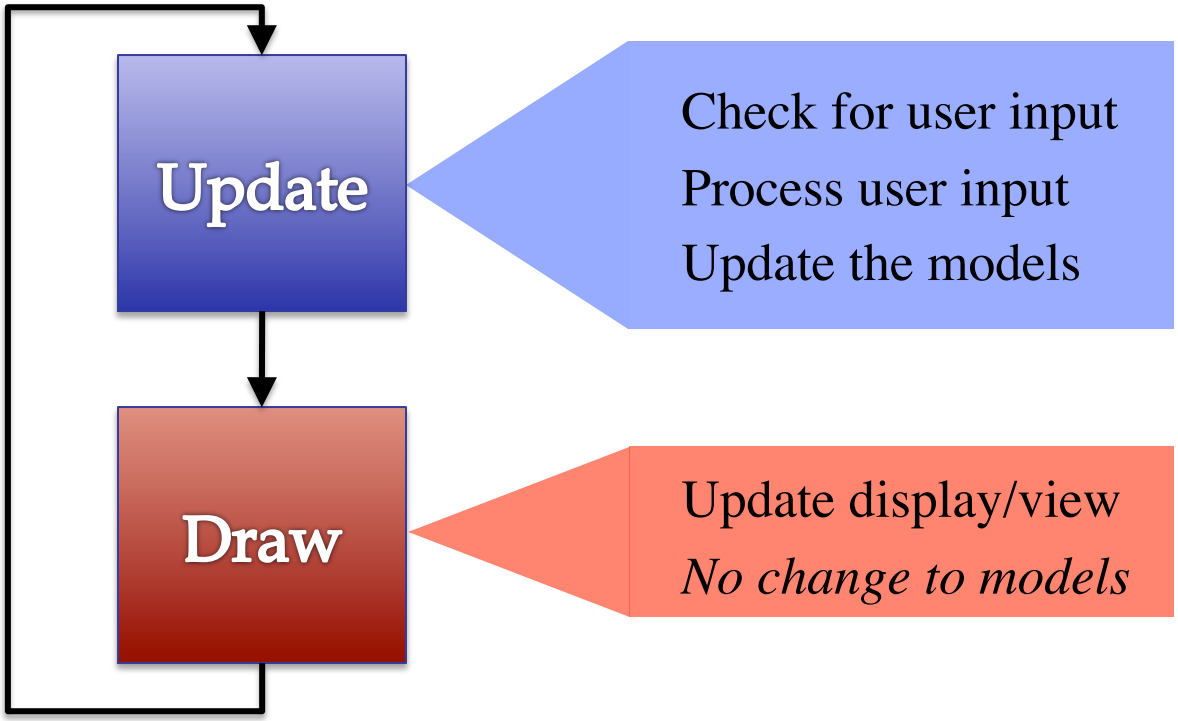
A Standard Game Application

Animates the application, like a movie

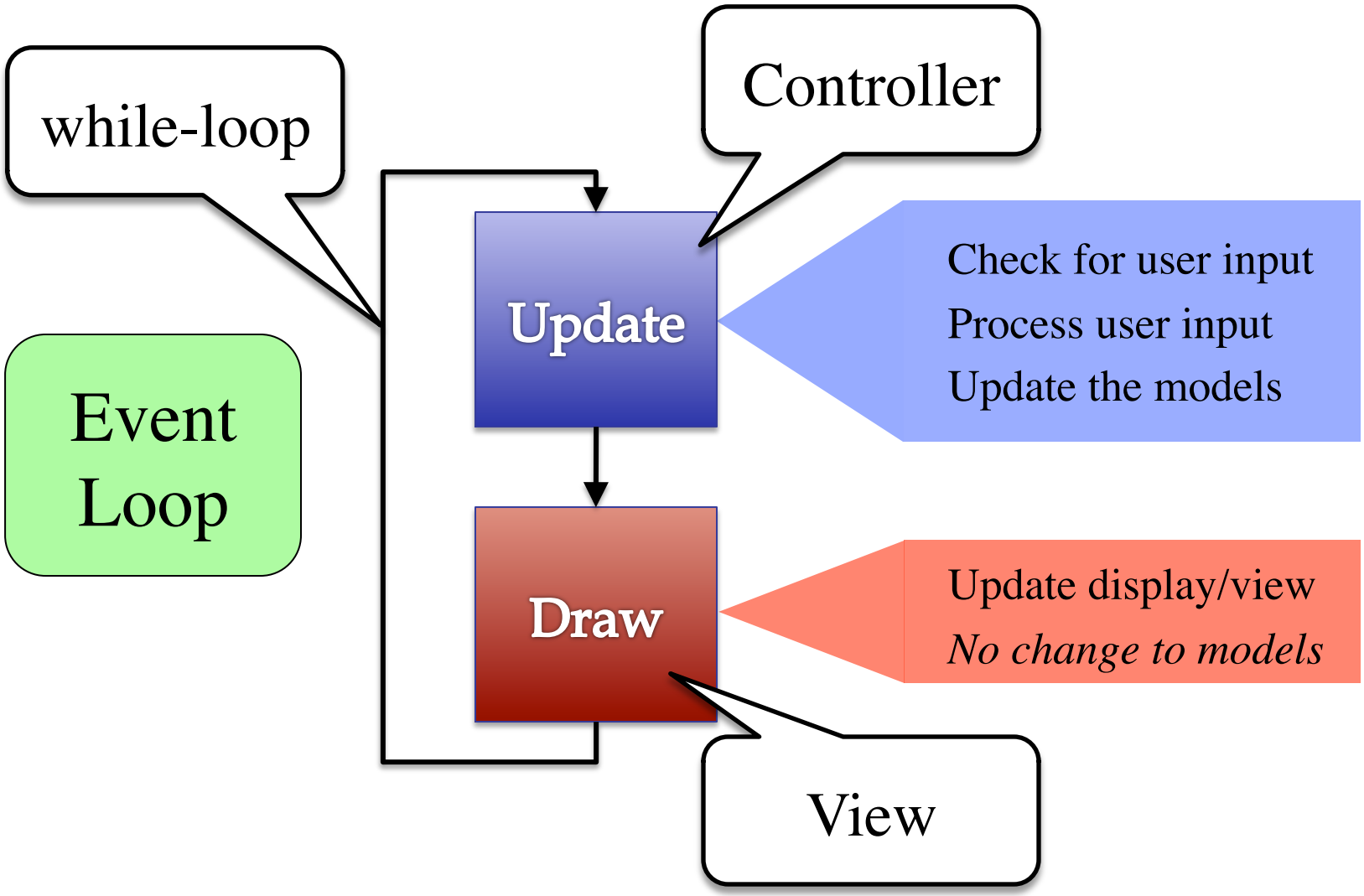


A Standard Game Application

Animates the application, like a movie



A Standard Game Application



Must We Write this Loop Each Time?

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

    # Code to respond to user input
    # Code to draw stuff in the window

    # Get window onto 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 libraries

    # Code to respond to user input
    # Code to draw stuff in the window

    # Get window onto the screen
    # Handled by OS/GUI libraries
```

Would like to "plug in" code

Why do we need to write this each time?

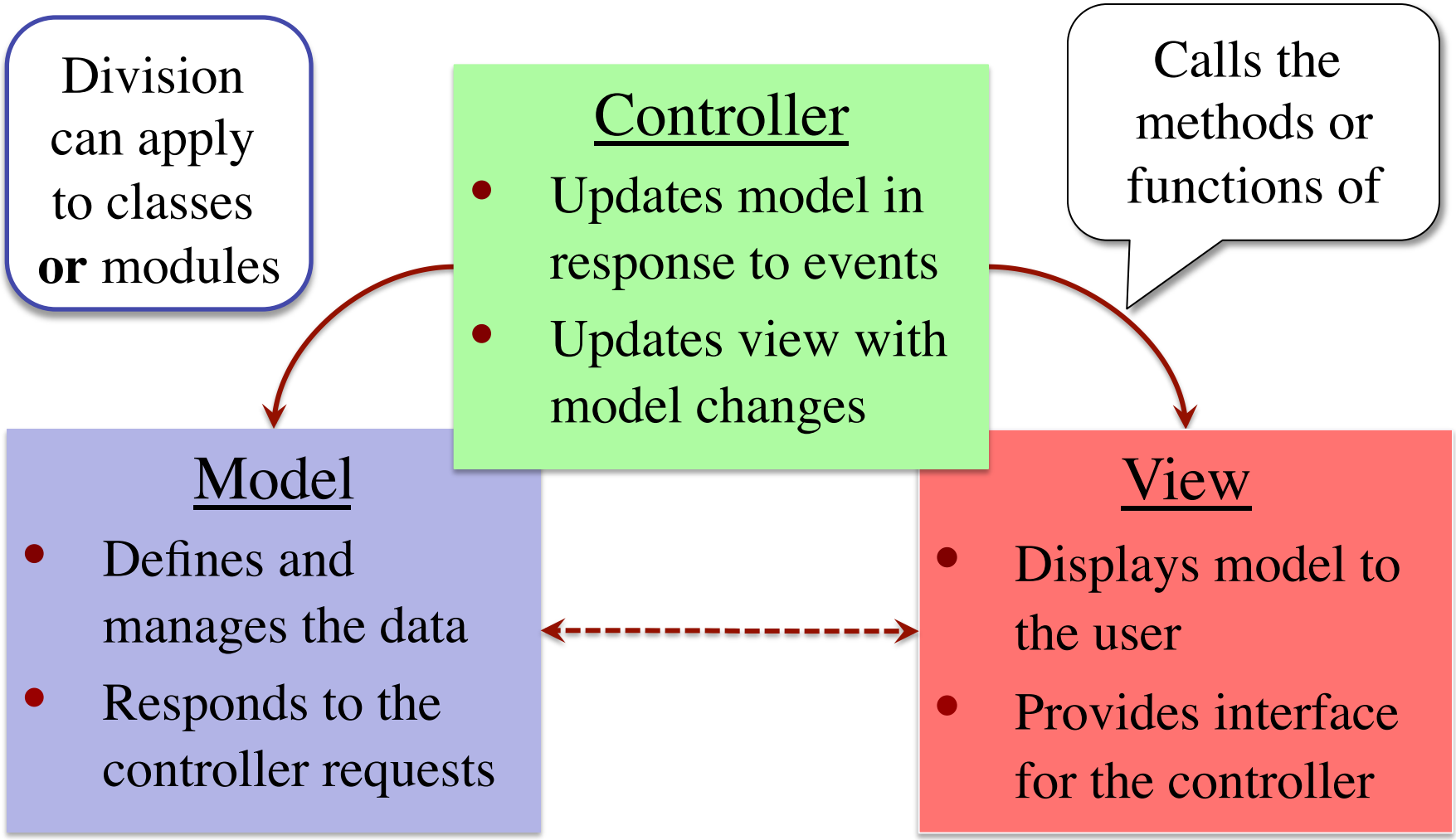
Must We Write this Loop Each Time?

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

    controller.update()
    controller.draw()

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

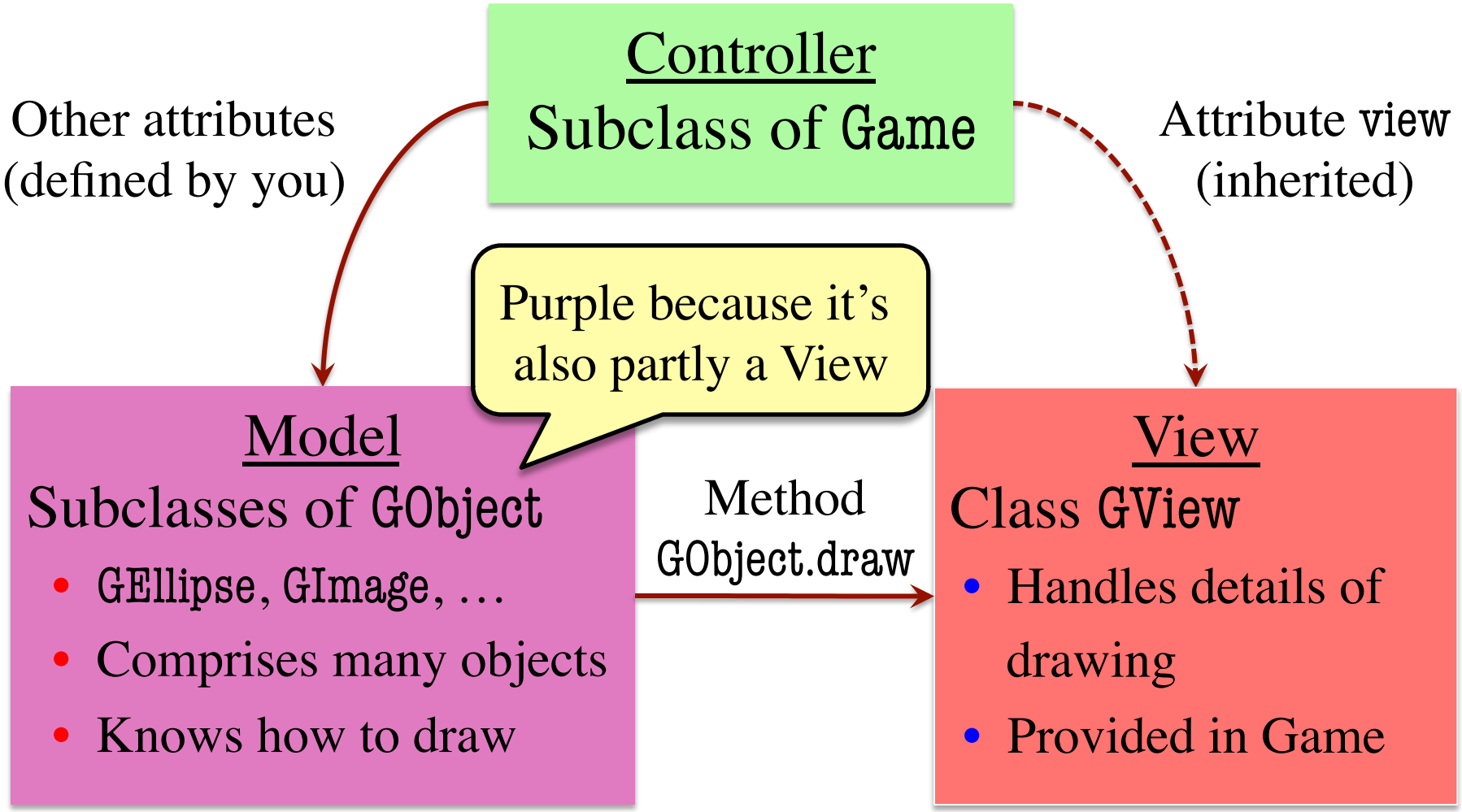
Method calls for loop body

- No: make this loop part of the library!
- Then we only have to provide the controller class.

Model-View-Controller Pattern



Model-View-Controller in A5



Example: Animation

```
class Animation(Game):
    """Application to an ellipse in a circle."""

    def init(self):
        """Special loop initialization method."""
        ...

    def update(self, dt):
        """Change the ellipse position."""
        ...

    def draw(self):
        """Draw the ellipse"""
        ...
```

See animation.py

Loop initialization
Do NOT use __init__

Loop body

Use method draw()
defined in GObject

Example: Animation

```
class Animation(Game):
    """Application to an el
    def init(self):
        """Special loop initialization method."""
        ...
    def update(self, dt):
        """Change the ellipse position."""
        ...
    def draw(self):
        """Draw the ellipse"""
        ...
```

Parent class that handles details

See animation.py

Loop initialization
Do NOT use __init__

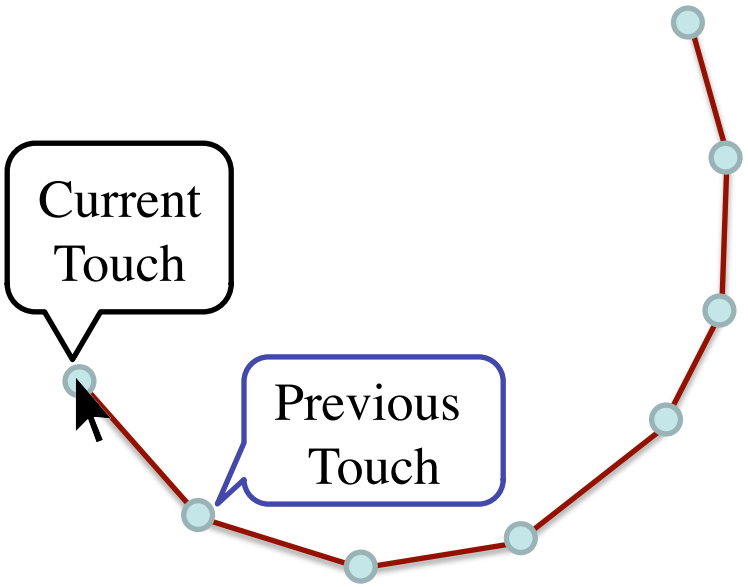
Loop body

Use method draw() defined in GObject

What Attributes to Keep: Touch

- Attribute touch in GView
  - The mouse press position
  - Or None if not pressed
  - Use self.view.touch inside controller (Game) methods
- Compare touch, last position
  - last None, touch not None: Mouse button **pressed**
  - last not None, touch None: Mouse button **released**
  - last and touch not None: Mouse **dragged** (button down)

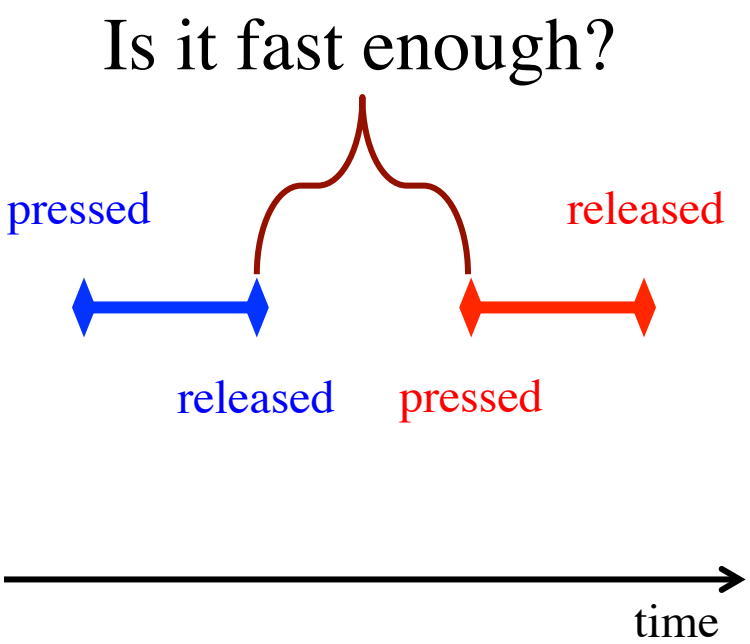
Line segment = 2 points



See touch.py

More Attributes: Checking Click Types

- Double click = 2 fast clicks
- Count number of fast clicks
  - Add an attribute `clicks`
  - Reset to 0 if not fast enough
- Time click speed
  - Add an attribute `time`
  - Set to 0 when mouse released
  - Increment when not pressed (e.g. in loop method `update()`)
  - Check `time` when next pressed

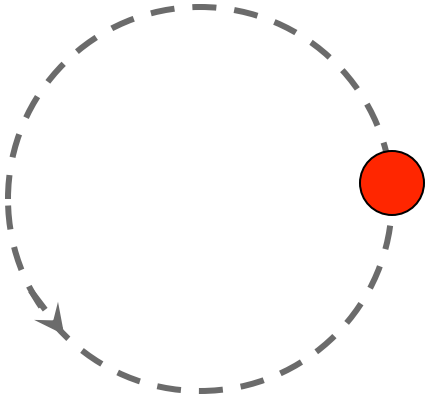


See `touch.py`

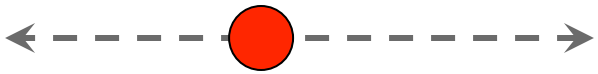
State: Changing What the Loop Does

- **State**: Current loop activity
  - Playing game vs. pausing
  - Ball countdown vs. serve
- Add an attribute `state`
  - Method `update()` checks state
  - Executes correct helper
- How do we store state?
  - State is an *enumeration*; one of several fixed values
  - Implemented as an int
  - Global **constants** are values

State `ANIMATE_CIRCLE`



State `ANIMATE_HORIZONTAL`

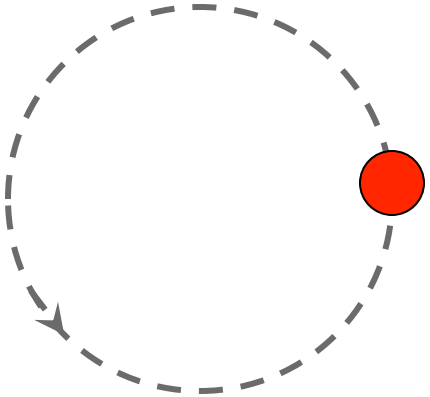


See state.py

State: Changing What the Loop Does

- **State**: Current loop activity
  - Playing game vs. pausing
  - Ball countdown vs. serve
- Add an attribute `state`
  - Method `update()` checks state
  - Executes correct helper
- How do we store state?
  - State is an *enumeration*; one of several fixed values
  - Implemented as an int
  - Global **constants** are values

State `ANIMATE_CIRCLE`



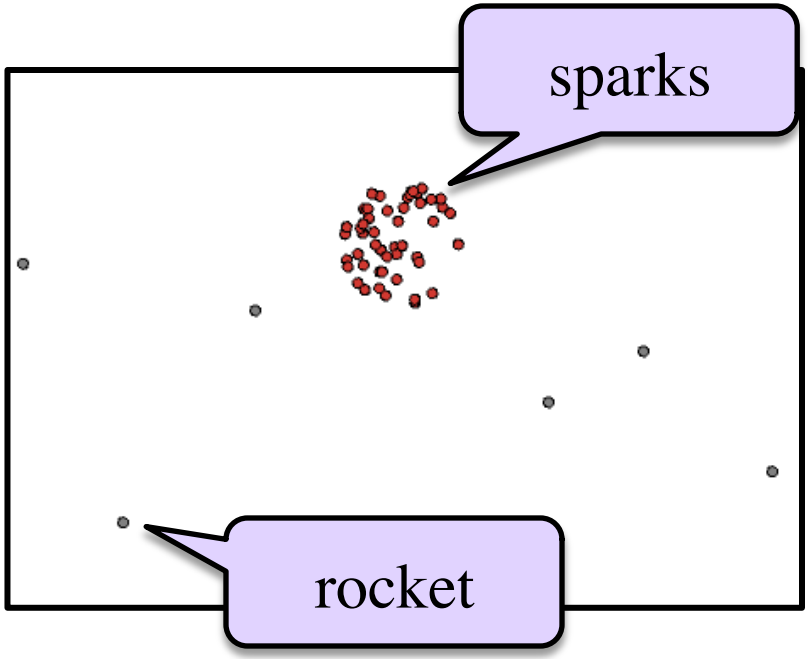
State `ANIMATE_HORIZONTAL`

Importance of class invariants

e.py

Types of Models for Assignment 6

- Often subclass of `GObject`
  - Has built-in draw method
  - See documentation in A6
- Includes groups of models
  - **Example**: rockets in `pyro.py`
  - Each rocket is a model
  - But so is the entire list!
  - `update()` will change both
- **A5**: `Model` class
  - Container, like A4 `Dataset`
  - Holds bricks, ball, paddle



See `pyro.py`