CS 1110
Lecture 25: Models, Views, Controllers, and Games

Announcements

A5
…is out! Get started right away—you need time to ask questions.

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

Office/consulting hours
…will be changing for study week. See the webpage for details.

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.)
A Standard Game Application

Animates the application, like a movie
A Standard Game Application

Animates the application, like a movie

Update

Draw

Check for user input
Process user input
Update the models
A Standard Game Application

Animates the application, like a movie

- **Update**
  - Check for user input
  - Process user input
  - Update the models

- **Draw**
  - Update display/view
  - *No change to models*
A Standard Game Application

while-loop

Event Loop

Controller

Check for user input
Process user input
Update the models

Update

while-loop

View

Update display/view
No change to models

Draw

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop

while-loop
Must We Write this Loop Each Time?

```python
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?

```python
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
```

Would like to “plug in” code

Why do we need to write this each time?
Must We Write this Loop Each Time?

```python
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
```

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

Model
• Defines and manages the data
• Responds to the controller requests

Controller
• Updates model in response to events
• Updates view with model changes

View
• Displays model to the user
• Provides interface for the controller

Division can apply to classes or modules

Calls the methods or functions of...
Model-View-Controller in A5

Controller
Subclass of Game

Attribute view (inherited)

Other attributes (defined by you)

Model
Subclasses of GObject
- GEllipse, GImage, ...  
- Comprises many objects  
- Knows how to draw

View
Class GView
- Handles details of drawing  
- Provided in Game

Purple because it’s also partly a View

Method GObject.draw
Example: Animation

```python
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
```python
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

Parent class that handles details

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

Is it fast enough?

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

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`
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`