Lecture 24

GUI Applications
Announcements for This Lecture

Prelim 2

- Prelim, Thurs at 7:30
  - A–N in Bailey 101
  - O–Z in Uris G01
- Material up to Nov. 8
  - Recursion + Loops + Classes
- Conflict with Prelim?
  - Made decisions Monday
  - Contact Amy if no email

Assignments

- A6 due TONIGHT
  - Last minute extension
  - Also, fill out survey
- A7 due December 7th
  - Focus of today’s lecture
  - 2.5 weeks excluding T-Day
  - 3 weeks including the break
  - Minor extensions possible
- Both are very important
  - Each worth 8% of grade

New Rooms!
Announcements for This Lecture

Labs

• There is a **lab today**
  ▪ Covers lecture material
  ▪ Jump starts you on A7
  ▪ But no manual grade

• **No lab** Thurs/Fri or Tues!
  ▪ Time to study/work on A7
  ▪ Will have open OH Tues

• Nov 29/30 is **LAST LAB**
  ▪ Will have manual grade
  ▪ Finish it by end of week

(Optional) Videos

• **Lesson 27 (all)** for today
• **Lesson 30** for next time
• Will return to **Lesson 28** later
A Standard GUI Application

Animates the application, like a movie
A Standard GUI Application

Animates the application, like a movie

- Check for user input
- Process user input
- Update the objects
A Standard GUI Application

Animates the application, like a movie

Update

Check for user input
Process user input
Update the objects

Draw

Update display/view
No change to objects

Restriction set by graphics cards
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

Why do we need to write this each time?

Would like to “plug in” code
Must We Write this Loop Each Time?

```python
while program_is_running:
    # Get information from mouse/keyboard
    # Handled by OS/GUI libraries
    # Your code goes here
    application.update()
    # Draw stuff on the screen
    # Handled by OS/GUI libraries
```

- Write loop body in an app class.
- OS/GUI handles everything else.

Custom Application class with its own attributes

Method call (for loop body)
Programming Animation

Intra-Frame

- Computation within frame
  - Only need current frame
- **Example:** Collisions
  - Need current position
  - Use to check for overlap
- Can use **local variables**
  - All lost at `update()` end
  - But no longer need them
Programming Animation

Inter-Frame

- Computation across frames
  - Use values from last frame
- **Example:** Movement
  - Need old position/velocity
  - Compute next position
- Requires **attributes**
  - Attributes never deleted
  - Remain after `update()` ends
Variables and the Loop

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

    # Your code goes here
    application.update()

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

Local variables erased. But **attributes** persist.
## Programming Animation

### Intra-Frame
- Computation within frame
  - Only need current frame
- **Example:** Collisions
  - Need current position
  - Use to check for overlap
- Can use **local variables**
  - All lost at `update()` end
  - But no longer need them

### Inter-Frame
- Computation across frames
  - Use values from last frame
- **Example:** Movement
  - Need old position/velocity
  - Compute next position
- Requires **attributes**
  - Attributes never deleted
  - Remain after `update()` ends
Attributes = Loop Variables

Normal Loops

Variables “external” to the loop body

\[
x = 0
\]

\[
i = 2
\]

# x = sum of squares of 2..i-1

\[
\text{while } i \leq 5:
\]

\[
x = x + i^2
\]

\[
i = i + 1
\]

# x = sum of squares of 2..5

Application

Attributes are the “external” variables

\[
\text{while program\_running:}
\]

# Get input

# Your code called here

application.update()

# Draw

11/15/22

GUI Applications
# Constructor

game = GameApp(...)  

...  

game.start() # Loop initialization  

while program_running:  
    # Get input  
    # Your code goes here  

game.update(time_elapsed)  

game.draw()
Designing a Game Class: Animation

```python
class Animation(game2d.GameApp):
    """App to animate an ellipse in a circle."""

    def start(self):
        """Initializes the game loop."""
        ...

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

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

See animation.py
class Animation(game2d.GameApp):
    """App to animate an ellipse."""

def start(self):
    """Initializes the game loop."""
    ...

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

def draw(self):
    """Draws the ellipse"""
    ...

Parent class that does hard stuff

See animation.py
class Animation(game2d.GameApp):
    """App to animate an ellipse."""
    ...

def start(self):
    """Initializes the game loop."""
    ...

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

def draw(self):
    """Draws the ellipse"""
    ...

See animation.py

Parent class that does hard stuff

Loop initialization
Do NOT use __init__

Loop body

Use method draw() defined in GObject
Comparing Attributes: Touch

- Attribute `touch` in `GInput`
  - The mouse press position
  - Or `None` if not pressed
  - Access with `self.input.touch`
- Compare `touch`, `last` position
  - Mouse button `pressed`: last `None`, `touch` not `None`
  - Mouse button `released`: last `not None`, `touch` `None`
  - Mouse `dragged`: last and `touch` `not None`

Line segment = 2 points

See `touch.py`
Input and Invariants

• Attribute input is…
  ▪ A GInput object

• Attribute input.touch is…
  ▪ Either a Point2 or None
  ▪ Location of mouse cursor (if it is pressed)

• Attribute last is…
  ▪ Either a Point2 or None
  ▪ input.touch in prev. frame

Line segment = 2 points

See touch.py

Relationship between two variables.
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

State `ANIMATE_CIRCLE`

State `ANIMATE_HORIZONTAL`

See `state.py`
Designing States

• Each state has its own set of invariants.
  ▪ Drawing? Then touch and last are not None
  ▪ Erasing? Then touch is None, but last is not

• Need rules for when we switch states
  ▪ Could just be “check which invariants are true”
  ▪ Or could be a triggering event (e.g. key press)

• Need to make clear in class invariant
  ▪ What are the invariants for each state?
  ▪ What are the rules to switch to a new state?
Triggers: 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`
Designing Complex Applications

• Applications can become extremely complex
  ▪ Large classes doing a lot
  ▪ Many states & invariants
  ▪ Specification unreadable

• **Idea**: Break application up into several classes
  ▪ Start with a “main” class
  ▪ Other classes have roles
  ▪ Main class delegates work

- Processes input
- Determines state
- Animates (only)

See `subcontroller.py`
How to Break Up: Software Patterns

• **Pattern**: reusable solution to a common problem
  - Template, not a single program
  - Tells you how to design your code
  - Made by someone who ran into problem first

• In many cases, a pattern gives you the interface
  - List of headers for non-hidden methods
  - Specification for non-hidden methods
  - Only thing missing is the implementation

11/15/22
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 the model to the app user
- Provides user input to the controller

Division can apply to classes or modules

Calls the methods or functions of
## MVC in this Course

<table>
<thead>
<tr>
<th><strong>Model</strong></th>
<th><strong>Controller</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A3</strong>: Color classes</td>
<td></td>
</tr>
<tr>
<td>- RGB, CMYK &amp; HSV</td>
<td></td>
</tr>
<tr>
<td><strong>A4</strong>: Turtle, Pen</td>
<td></td>
</tr>
<tr>
<td>- Window is View</td>
<td></td>
</tr>
<tr>
<td><strong>A6</strong>: Image</td>
<td></td>
</tr>
<tr>
<td>- Data is always in model</td>
<td></td>
</tr>
<tr>
<td><strong>A7</strong>: Ship, Asteroid, etc..</td>
<td></td>
</tr>
<tr>
<td>- All shapes/geometry</td>
<td></td>
</tr>
<tr>
<td><strong>A3</strong>: a3app.py</td>
<td></td>
</tr>
<tr>
<td>- Hidden classes</td>
<td></td>
</tr>
<tr>
<td><strong>A4</strong>: Functions in a4.py</td>
<td></td>
</tr>
<tr>
<td>- No need for classes</td>
<td></td>
</tr>
<tr>
<td><strong>A6</strong>: Filter, Encode</td>
<td></td>
</tr>
<tr>
<td>- Drives program forward</td>
<td></td>
</tr>
<tr>
<td><strong>A7</strong>: Planetoids, Wave</td>
<td></td>
</tr>
<tr>
<td>- Main part of assignment!</td>
<td></td>
</tr>
</tbody>
</table>

11/15/22 GUI Applications 27
## MVC in this Course

### Model
- **A3**: Color classes
  - RGB, CMYK & HSV
- **A4**: Turtle, Pen
  - Window is View
- **A6**: Dataset, Cluster
  - Data is always in model
- **A7**: Ship, Asteroid, etc.
  - All shapes/geometry

### Controller
- **A3**: a3app.py
  - Hidden classes
- **A4**: Functions in a4.py
  - No need for classes
- **A6**: Filter, Encode
  - Drives program forward
- **A7**: Planetoids, Wave
  - Main part of assignment!

---

Why classes sometimes and functions others?
Model-View-Controller in CS 1110

Model
Subclasses of GObject
• GEllipse, GImage, …
• Often more than one

Controller
Subclass of GameApp

View
Class GView, GInput
• Do not subclass!
• Part of GameApp

Other attributes
(defined by you)

Attribute view
(inherited)

Method draw
in GObject

Classes in game2d

GUI Applications 29
Model-View-Controller in CS 1110

Other attributes (defined by you)

Controller
Subclass of GameApp

Attribute view (inherited)

Model
Subclass of GameApp

Neglected for most of this lecture

Method draw in GObject

View
Class GView, GInput
- Do not subclass!
- Part of GameApp

Classes in game2d

GUI Applications 30

11/15/22
Models in Assignment 7

- Often subclass of GObject
  - Has built-in draw method
- Includes groups of models
  - Example: rockets in pyro.py
  - Each rocket is a model
  - But so is the entire list!
  - update() will change both
- A7: Several model classes
  - Ship to animate the player
  - Alien to represent an alien

See pyro.py