Lecture 24

GUI Applications
Announcements for This Lecture

Next Week

- There is no lab next week
  - But Tuesday hours are open
  - Open to EVERYONE
  - Go for help on lab or A7
- But lecture is important
  - Return to topic of invariants
  - Setting us up for sorting
- Try to finish lab 11 first
  - Frees remaining time for A7

Assignment 7

- Start working on it now!
  - Timeline is very important
  - Else too much stress at end
- Goal: Move aliens by Tues
  - Expected to be hardest part
  - Use lab next week
- Due last day of class
  - Midnight, December 4th
  - Last day I am allowed to
A Standard GUI Application

Animates the application, like a movie
A Standard GUI Application

Updates the application, like a movie

- **Update**: Check for user input, process user input, update the objects
- **Draw**: Draw 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
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?

```python
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?

11/15/18 GUI Applications
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.
Normal Loops

\[ x = 0 \]
\[ i = 2 \]

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

\textbf{while} i <= 5:
  \[ x = x + i \times i \]
  \[ i = i + 1 \]

\# x = sum of squares of 2..5

Properties of “external” vars

Application

What are the “external” vars?

\textbf{while} program\_running:
  \# Get input
  \# Your code called here
  \texttt{application.update()}  
  \# Draw
Loop Invariants Revisited

Normal Loops

\[ x = 0 \]
\[ i = 2 \]
\[ \text{# } x = \text{sum of squares of } 2..i \]
\[ \textbf{while } i \leq 5: \]
\[ \quad x = x + i \times i \]
\[ \quad i = i + 1 \]
\[ \text{# } x = \text{sum of squares of } 2..5 \]

Properties of “external” vars

Application

What are the “external” vars?

\[ \textbf{while } \text{program\_running}: \]
\[ \quad \text{# Get input} \]
\[ \quad \text{# Your code called here} \]
\[ \quad \text{application.update()} \]
\[ \quad \text{# Draw} \]

Application is an object. It will have attributes!
Attribute Invariants = Loop Invariants

• Attributes are a way to store value between calls
  ▪ Not part of call frame
  ▪ Variables outside loop
• An application needs
  ▪ Loop attributes
  ▪ Initialization method (for loop, not \_\_init\_\_)  
  ▪ Method for body of loop
• Attribute descriptions, invariants are important

# Constructor
```python
game = GameApp(...)  
...
```

# Loop initialization
```python
game.start()  
# inv: game attributes are ...
```

```python
while program_running:
  # Get input
  # Your code goes here
  game.update(time_elapsed)
  game.draw()
  # post: game attributes are ...
```
Example: 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
Example: Animation

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
Example: Animation

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
What Attributes to Keep: Touch

• Attribute `touch` in `GInput`
  - The mouse press position
  - Or `None` if not pressed
  - Use `self.input.touch` inside your subclass definition

• Compare `touch`, `last` position
  - `last None, touch not None`: Mouse button `pressed`
  - `last not None, touch None`: Mouse button `released`
  - `last and touch both not None`: Mouse `dragged` (button down)

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

Relationship between two variables.

See touch.py

Line segment = 2 points
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`
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 specification
  - 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

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

![Diagram]

- 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

Just like this course!
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

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

## Controller
- **A3**: `a3app.py`
  - Hidden classes
- **A4**: Functions in `a4.py`
  - No need for classes
- **A6**: Algorithm
  - Also our custom modules
- **A7**: Invaders, Wave
  - Main part of assignment!
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, Alien, etc.
  - All shapes/geometry

Why **classes** sometimes and **functions** others?

Controller

- **A3**: a3app.py
  - Hidden classes
- **A4**: Functions in a4.py
  - No need for classes
- **A6**: Algorithm
  - Also our custom modules
- **A7**: Invaders, Wave
  - Main part of assignment!
Model-View-Controller in CS 1110

Controller
Subclass of GameApp

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

Method draw
in GObject

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

Classes in game2d

Other attributes (defined by you)

Attribute view (inherited)

11/15/18
GUI Applications
Model-View-Controller in CS 1110

Controller
Subclass of GameApp

Attribute view (inherited)

Other attributes (defined by you)

Method draw in GObject

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

Classes in game2d

Model
Subclass of GameApp

Neglected for most of this lecture
Models in Assignment 7

• 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
• A7: Several model classes
  ▪ Ship to animate the player
  ▪ Alien to represent an alien

See pyro.py