Lecture 22: GUI Applications

CS 1110
Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]
Relevant modules (see schedule page)

- animation.py
- arrows.py
- pyro.py
- state.py
- subcontroller.py
- game2d.zip (unzip)

Try them out!

>>> python arrows.py
A Standard GUI Application

Animates the application, like a movie

Update State
- Check for user input
- Process user input
- Update the objects

Draw
- Update display/view
- No change to objects
while program_is_running:
  
  # Get keyboard info ← Handled by OS/GUI libraries

  # Your code goes here
  application.update() ← Custom application class

  # Draw stuff to screen ← Handled by OS/GUI libraries
Example: Animation

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

Parent class that does hard stuff

Loop initialization
Do NOT use __init__

Loop body

Use method draw() defined in GObject
Does update() respond to the user?

A) Yes, any key changes the animation
B) Yes, certain keys select certain animations
C) No
D) I don't know
Does update() respond to the user?

A) Yes, any key changes the animation
B) Yes, certain keys select certain animations
C) No
D) I don't know

See state.py
Changing the Loop Activity

- **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`
Need rules for when we switch states

- Attribute `key_count` in GInput
  - How many keys are pressed?
  - 0, 1, 2, …
  - `curr_keys = self.input.key_count`
- Is this a new key press?
  - Need current `curr_keys`, and key count from last time:
    - `lastkeys`
    
    `change = curr_keys > 0 and self.lastkeys == 0`
- When we're done, `curr_keys` becomes the new `lastkeys`
  
  `self.lastkeys = curr_keys`

See `_determineState(self):` in state.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
- Calls the methods of
- 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
draw() methods

Division can apply to classes or modules

Wave & Invaders Classes
- Ship, Alien & Bolt Classes

Calls the methods or functions of...
Model-View-Controller in CS 1110

Controller
Subclass of GameApp

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

Neglected for most of this lecture

Method `draw` in GObject

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

Classes in `game2d`

Attribute `view` (inherited)

Other attributes (defined by you)
Models in Assignment 7

• Often subclass of GObject
  ▪ Has built-in draw method
  ▪ See documentation in A7

• 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