A Standard GUI Application

- Animates the application, like a movie
- 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
application.update()
# Draw stuff on the screen
# Handled by OS/GUI libraries

Custom Application class with its own attributes

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

Designing a Game Class: 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"""
    ...

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

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
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

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

Model-View-Controller Pattern

MainApp • Processes input
  • Determines state
  • Calls the methods of
Animation • Animates (only)

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

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

View • Displays the model to the app user
  • Provides user input to the controller

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