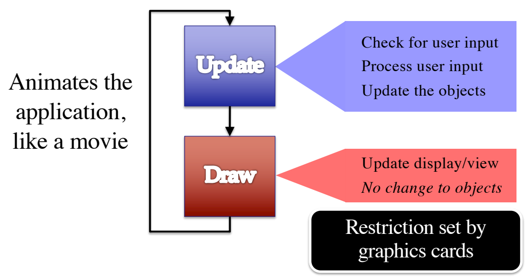
A Standard GUI Application



1

Must We Write this Loop Each Time?

```
while program_is_running:
    # Get information from mouse/keyboard
    # Handled b
    # Your code         ere
    application.update()
    #
    #                         es
```

Method call (for loop body)

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

Custom Application class with its own **attributes**

2

Programming Animation

Intra-Frame	Inter-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	• 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

3

Designing a Game Class: Animation

```
class Animation(game2d.GameApp):
    """App to animate an
    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

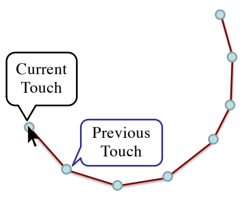
Use method draw() defined in GObject

4

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



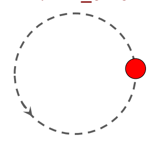
See touch.py

5

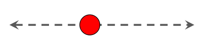
State: Changing What the Loop Does

- **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

6

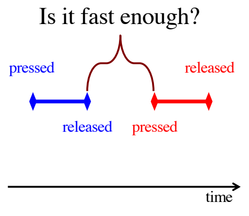
Designing States

- 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?

7

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

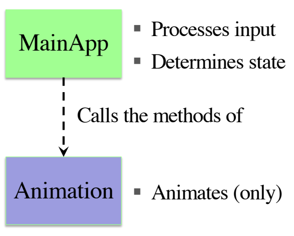


See touch.py

8

Designing Complex Applications

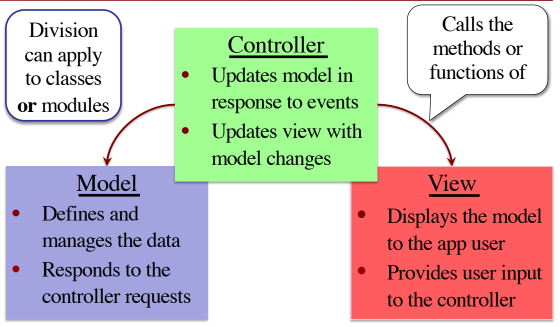
- Applications can become extremely complex
  - Large classes doing a lot
  - Many states & invariants
  - Specification unreadable
- **Idea**: Break application up into several classes
  - Start with a "main" class
  - Other classes have roles
  - Main class delegates work



See subcontroller.py

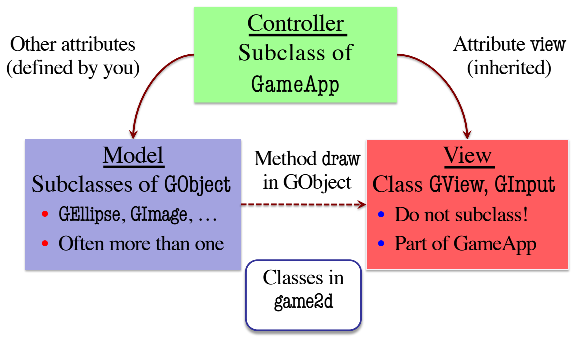
9

Model-View-Controller Pattern


Division can apply to classes **or** modules
Calls the methods or functions of
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

10

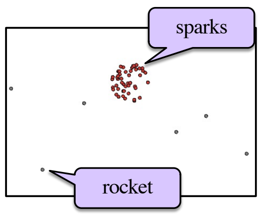
Model-View-Controller in CS 1110



11

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

12