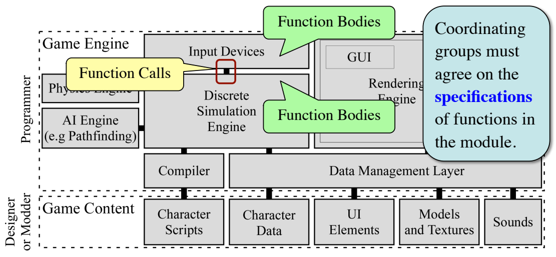
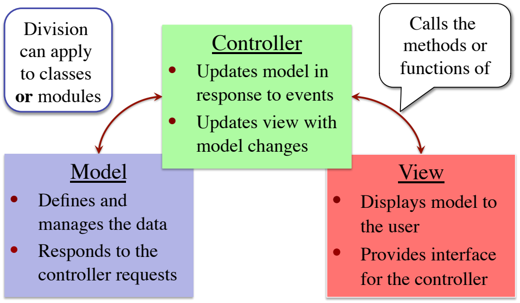
Challenge: Breaking Up Software



Coordinating groups must agree on the **specifications** of functions in the module.

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 model to the user
- Provides interface for the controller

MVC in this Course

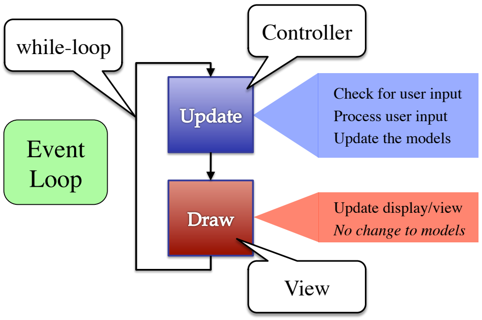
Model

- **A3**: Color classes
  - RGB, CMYK & HSV
- **A6**: Database, Cluster
  - Data is always in model
- **A7**: Ball, Brick, etc..
  - All shapes/geometry

Controller

- **A3**: a3app.py
  - Hidden classes
- **A6**: ClusterGroup
  - Also visualizer
- **A7**: Breakout
  - Controller class for you!

A Standard GUI Application



Check for user input
Process user input
Update the models

Update display/view
No change to models

Must We Write this Loop Each Time?

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

    # Your code g
    controller.update()

                      n the screen
    # Handled by OS/GUI libraries
```

Method call (for loop body)

Controller object

- Write loop body in a controller.
- OS/GUI handles everything else.

Loop Invariants Revisited

Normal Loops

Properties of “external” vars

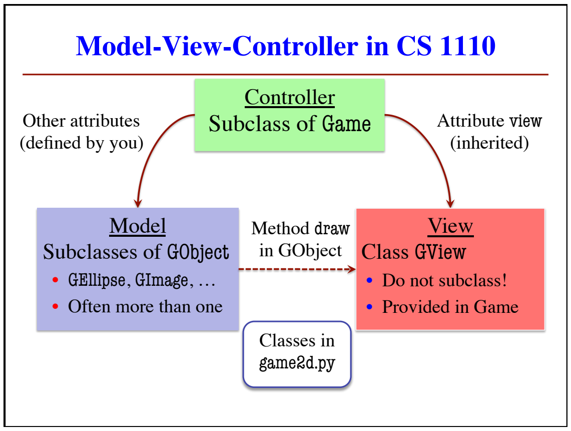
```
x = 0
i = 2
# x = sum of squares of 2..i
while i <= 5:
    x = x + i*i
    i = i +1
# x = sum of squares of 2..5
```

Controller

What are the “external” vars?

```
while program_running:
    # Get input
    # Your code called here
    controller.update()
    # Draw
```

controller is an object.
It will have **attributes**!



Model-View-Controller in CS 1110

Controller
Subclass of Game

Other attributes
(defined by you)

Attribute view
(inherited)

Model
Subclasses of GObject

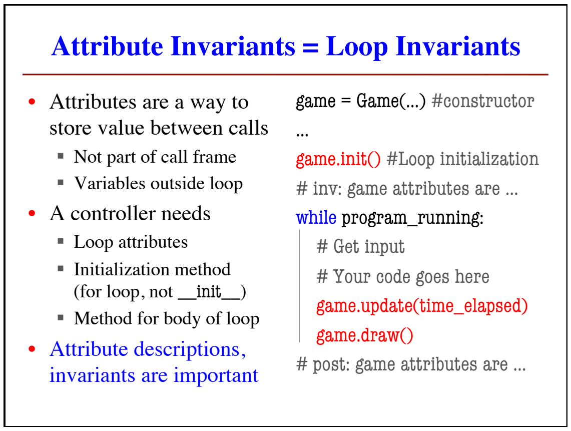
- GEllipse, GImage, …
- Often more than one

Method draw
in GObject

View
Class GView

- Do not subclass!
- Provided in Game

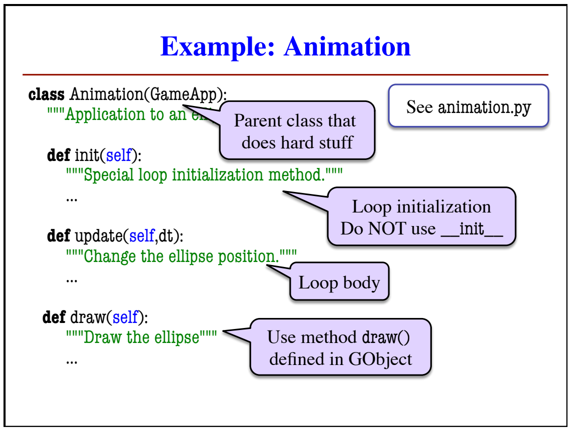
Classes in
game2d.py



Attribute Invariants = Loop Invariants

- Attributes are a way to store value between calls
  - Not part of call frame
  - Variables outside loop
- A controller needs
  - Loop attributes
  - Initialization method (for loop, not __init__)
  - Method for body of loop
- Attribute descriptions, invariants are important

```
game = Game(...) #constructor
...
game.init() #Loop initialization
# inv: game attributes are ...
while program_running:
    # Get input
    # Your code goes here
    game.update(time_elapsed)
    game.draw()
# post: game attributes are ...
```



Example: Animation

```
class Animation(GameApp):
    """Application to an e

    def init(self):
        """Special loop initialization method."""
        ...

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

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

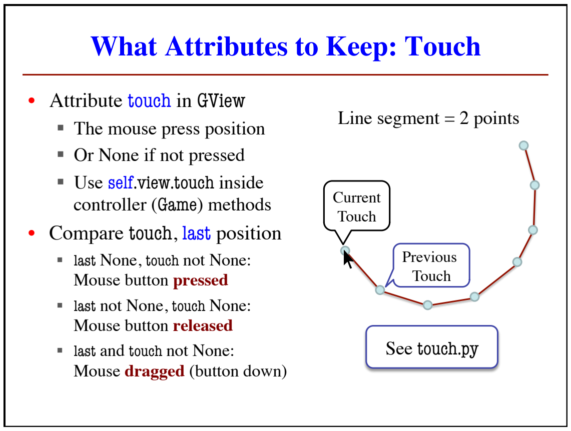
Parent class that does hard stuff

See animation.py

Loop initialization
Do NOT use __init__

Loop body

Use method draw()
defined in GObject



What Attributes to Keep: Touch

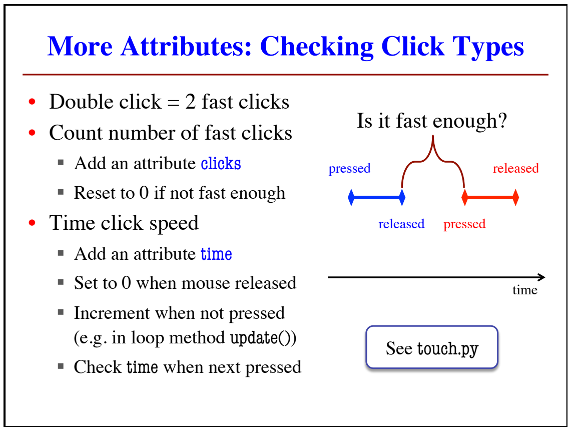
- Attribute touch in GView
  - The mouse press position
  - Or None if not pressed
  - Use self.view.touch inside controller (Game) methods
- Compare touch, last position
  - last None, touch not None: Mouse button **pressed**
  - last not None, touch None: Mouse button **released**
  - last and touch not None: Mouse **dragged** (button down)

Line segment = 2 points

Current
Touch

Previous
Touch

See touch.py

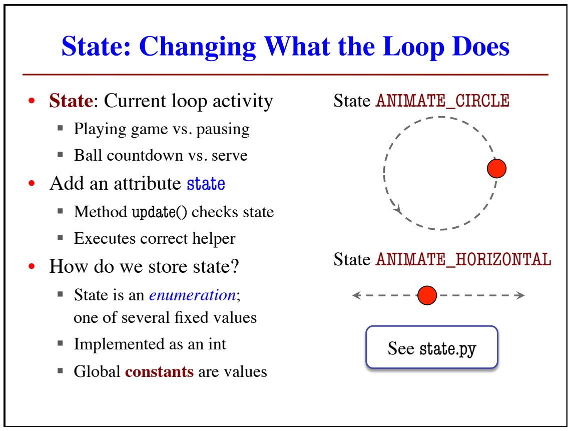


More Attributes: Checking Click Types

- 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



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