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
```

Class Invariant = Loop Invariants

- Look at the `game loop`
  - Loop body is `update()`
  - Loop vars are attributes
- Class invariant is true
  - At `update()`/body start
  - At `update()`/body end
- Invariants are important!
  - To reason about game
  - Help us debug problems

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.***
        Loop body
    def draw(self):
        ***Draws the ellipse***
        Use method `draw()` defined in GObject
```

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

Current Touch
Previous Touch
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

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

See `subcontroller.py`

---

**Model-View-Controller Pattern**

- **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
- **Controller**: Updates model in response to events
  - Updates view with model changes

Calls the methods or functions of

**Model-View-Controller in CS 1110**

- **Controller Subclass of GameApp**
  - `draw` in `GObject`
  - `GView, GInput`
  - Method draws in `GameApp`
- **View**
  - `GView, GInput`
  - Do not subclass!
  - Part of GameApp
- **Model**
  - Subclasses of `GObject`
  - `GEllipse, GImage, …`
  - Often more than one

Classes in `game2d`