MVC

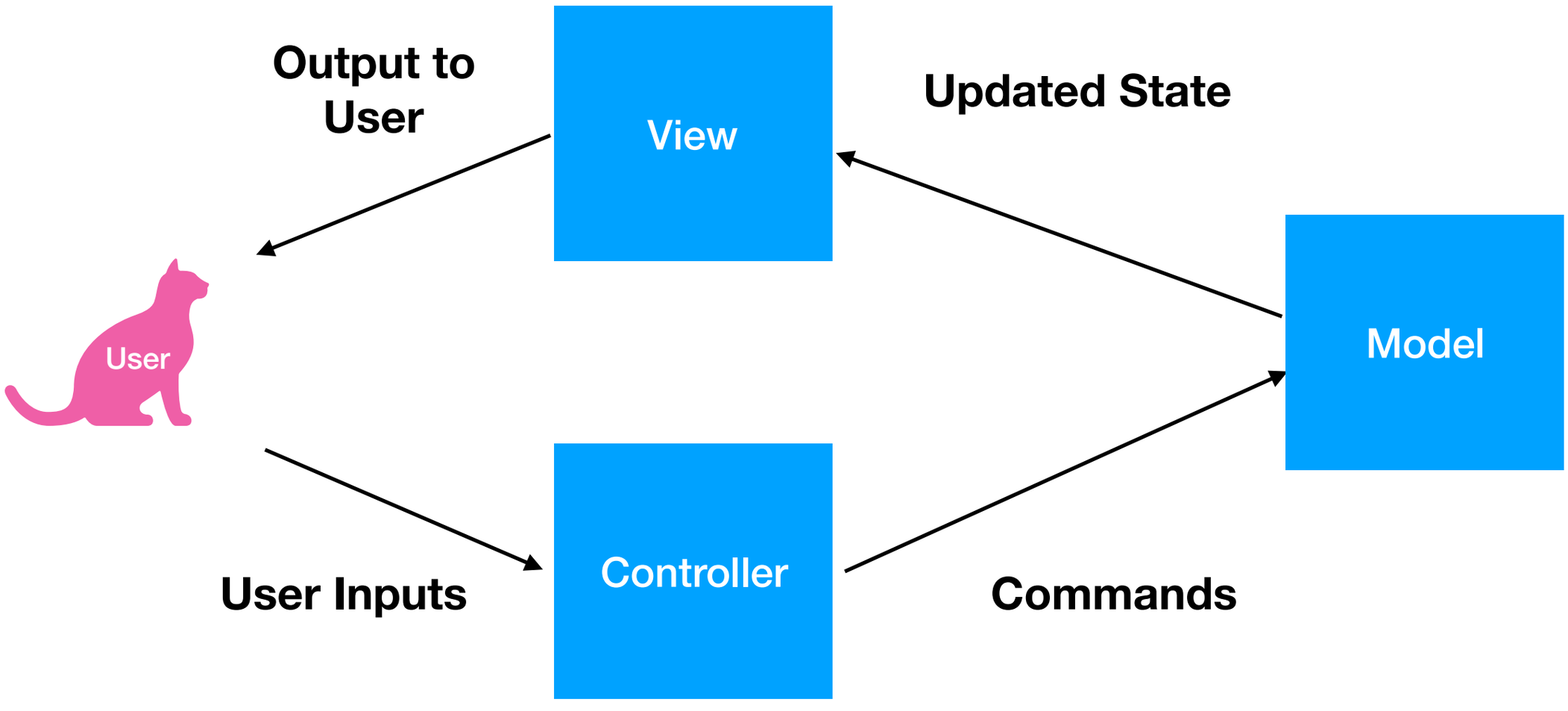
MVC

- Software architecture pattern
  - A way to organize our code
- Separate code into a Model, View, and Controller
- All three parts work independently and communicate with each other through APIs

MVC

- Model (Data and Logic)
  - Controls the app and it's data
  - The core of the app
- View (Display)
  - Visualizes the app
  - No logic
- Control (User Inputs)
  - Handles user inputs
  - Sends commands to the model based on inputs

MVC


Output to User
View
Updated State
User
Model
Controller
User Inputs
Commands

MVC - Model

- The core of the app
- What you've build for all 3 homework assignments
    - Think of physics handling how objects move in a game world
- Controls the logic and functionality of the app
- Maintains the data
    - Controls any databases/files
- Has no knowledge of the user of the app
- Accessed through an API

MVC - View

- Displays the state of the app to the user
- **Output only**
- No logic
    - The view cannot change the state of the app
    - Since the view is output only and does not alter the app it can change and be replaced without affecting the app itself
    - Can have the same app with a CLI (command line interface) and a GUI (graphical user interface)
    - Can have the same app with a web front-end and a desktop front-end!

MVC - Controller

- Handles user inputs
- Think EventHandlers
- Processes user inputs and converts them into commands for the Model
- Communicates with Model via it's API (methods and variables)
- Can validate and block invalid inputs

MVC - Advantages

- Focus on 1 piece of the code at a time
  - Reduce spaghetti code
  - Divide work to team members
- Views can be easily replaced
- Keeps code organized
- Easier to add new features
  - Model can add features as long as API remains unchanged

API

- We've seen web APIs which have various endpoints
    - In general, APIs are a set of functions/methods that can be called
- A model has an API consisting of it's functions/methods that can be called by the controller
    - Controller only cares about these endpoints
    - Controller does not care about how these functions are implemented
    - Model can update and add features, just don't change the API
- Ex. Physics was lacking bouncing off walls. Add this without touching the view or controller

MVC on the Web

- Model runs on the server
- View runs in the browser (HTML/CSS)
- Controller can run on both
  - JavaScript in the browser converts user inputs into AJAX requests
  - Server validates the data and sends the commands to the Model

Project

- For Lab Demo 1 You had a choice of either model, view, or controller
- Moving forward it is recommended that you stick with this
  - Ideally, group of 3 with one member for each
  - Agree on your APIs
  - Work somewhat independently
  - Avoid standstills since each member knows what to implement
- For demo 2 desktop app, keep the three parts separate
  - Make demo 3 much easier when you have to separate them for the web app

Lecture Question

Free Again. Submit anything

- Finish Clicker 1 if you haven't already
- Prepare for Project Demo 1

* This question will be open until midnight