Section 8:
Model-View-Controller

Slides by Alex Mariakakis

with material from Krysta Yousoufian
and Kellen Donohue

Agenda

- MVC
- MVC example 1: traffic light
- MVC example 2: registration
- HW8 info

MVC

- The classic design pattern
- Used for data-driven user applications
- Such apps juggle several tasks:
  - Loading and storing the data – getting it in/out of storage on request
  - Constructing the user interface – what the user sees
  - Interpreting user actions – deciding whether to modify the UI or data
- These tasks are largely independent of each other
- Model, view, and controller each get one task

Model

talks to data source to retrieve and store data

Which database tables is the requested data stored in?

What SQL query will get me the data I need?

View

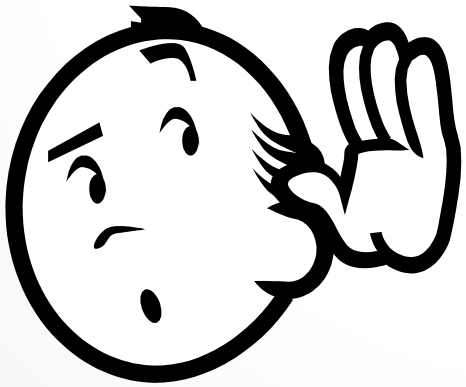
asks model for data and presents it in a user-friendly format

Would this text look better blue or red? In the bottom corner or front and center?

Should these items go in a dropdown list or radio buttons?

Controller

listens for the user to change data or state in the UI, notifying the model or view accordingly



The user just clicked the "hide details" button. I better tell the view.

The user just changed the event details. I better let the model know to update the data.

MVC: Summary

Model

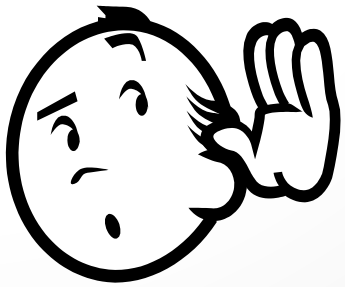
talks to data source to retrieve and store data

View

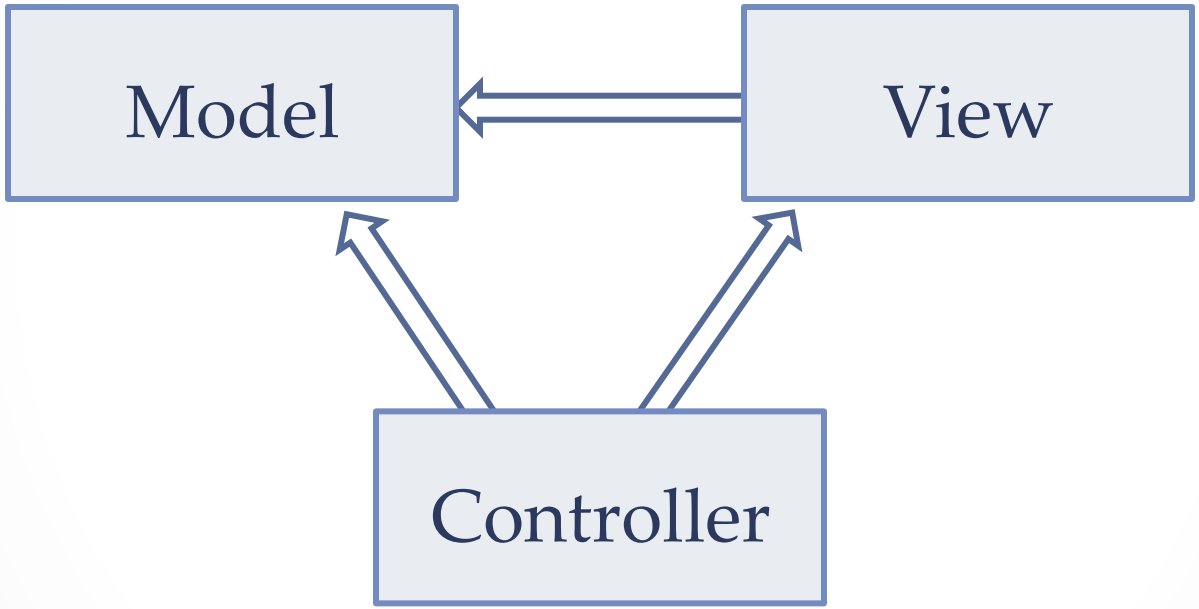
asks model for data and presents it in a user-friendly format

Controller

listens for the user to change data or state in the UI, notifying the model or view accordingly



Communication Flow



What do you think are the benefits of MVC?

Benefits of MVC

- Organization of code
  - Maintainable, easy to find what you need
- Ease of development
  - Build and test components independently
- Flexibility
  - Swap out views for different presentations of the same data (ex: calendar daily, weekly, or monthly view)
  - Swap out models to change data storage without affecting user

MVC Example – Traffic Signal

- Regulate valid traffic movements
  - Don't let cars run into each other
- Detect cars waiting to enter intersection
- Traffic lights to direct car traffic
- Manual override for particular lights
  - Automatic green for fire trucks
- Detect pedestrians waiting to cross
- Pedestrian signals to direct pedestrians
- External timer which triggers changes at set interval

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements			
Detect cars waiting to enter intersection			
Traffic lights to direct car traffic			
Manual override for particular lights			
Detect pedestrians waiting to cross			
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			
Traffic lights to direct car traffic			
Manual override for particular lights			
Detect pedestrians waiting to cross			
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic			
Manual override for particular lights			
Detect pedestrians waiting to cross			
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic		X	
Manual override for particular lights			
Detect pedestrians waiting to cross			
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic		X	
Manual override for particular lights			X
Detect pedestrians waiting to cross			
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic		X	
Manual override for particular lights			X
Detect pedestrians waiting to cross			X
Pedestrian signals to direct pedestrians			
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic		X	
Manual override for particular lights			X
Detect pedestrians waiting to cross			X
Pedestrian signals to direct pedestrians		X	
External timer which triggers changes at set interval			

Traffic Signal – MVC

Component	Model	View	Controller
Regulate valid traffic movements	X		
Detect cars waiting to enter intersection			X
Traffic lights to direct car traffic		X	
Manual override for particular lights			X
Detect pedestrians waiting to cross			X
Pedestrian signals to direct pedestrians		X	
External timer which triggers changes at set interval			X

Traffic Signal – Model

- Stores current state of traffic flow
  - Knows current direction of traffic
  - Capable of skipping a light cycle
- Stores whether there are cars and/or pedestrians waiting
- Implements Observable
- Example
  - TrafficModel – see above

Traffic Signal – Views

- Conveys information to cars and pedestrians in a specific direction
- Implements Observer
- Examples
  - CarLight – traffic light
  - PedestrianLight – pedestrian light

Traffic Signal – Controller

- Aware of model's current direction
- Triggers methods to notify model that state should change
- Examples
  - PedestrianButton – notifies TrafficModel that there is a pedestrian waiting
  - CarDetector – notifies TrafficModel that there is a car waiting
  - LightSwitch – enables or disables the light
  - Timer – regulates time in some way, possibly to skip cycles

MVC Example – Registration

- Registration system with web interface
- Advisors create classes, set space, time, restrictions
- Professors can see who's signed up for their class
- Students can see available classes and sign up for classes
- Administrators can place holds on student registration
- Professors can be notified when a student drops
- Students can be notified when a spot is available in a class they want

Registration

- Would you use push or pull?
- What would change for interaction with an API or mobile app?
- If advisors can see what students are registered for and change their registration, what changes?

HW8 Overview

- Apply your generic graph & Dijkstra's to campus map data
- Given a list of buildings and walking paths
- Produce routes from one building to another on the walking paths
  - Distance in feet, compass directions
- Command-line interface now
  - GUI in HW9

HW8 Data Format

- List of buildings (abbreviation, name, loc in pixels)

```
BAG Bagley Hall (East Entrance) 1914.5103,1708.8816
BGR By George 1671.5499,1258.4333
```

- List of paths (endpoint 1, endpoint 2, dist in feet)

```
1903.7201,1952.4322
        1906.1864,1939.0633: 26.583482327919597
        1897.9472,1960.0194: 20.597253035175832
        1915.7143,1956.5: 26.68364745009741
2337.0143,806.8278
        2346.3446,817.55768: 29.685363221542797
        2321.6193,788.16714: 49.5110360968527
        2316.4876,813.59229: 44.65826043418031
```

- (0,0) is in the upper left

MVC in HW8

- **Model** stores graph, performs Dijkstra's
- **View** shows results to users in text format
- **Controller** takes user commands and uses view to show results
- View and Controller will change in HW9, but Model will stay the same