MyWay

Phase 2: Design Documentation
Donglai Wei, Shidan Xu, Neil Gurram

Programming

1. Basic coding
   - functionality: We implemented two main functionalities
     - Finding a way based on the query
     - Manipulating (get/add/update/del) objects in the database
2. Modularity
   - We used a ruby-on-rail style of code management:
     - /api:
       - /api/app: MVC components (/models, /views, /controllers)
       - /api/config: configuration files for app parameters
     - /test: Qunit tests
     - /public: assets accessible from the html (/js, /stylesheets, /txt, /img, /csv)
     - /preprocess: off-line codes to prepare the data for way finding
3. Verification:
   - Unit tests: We provide reproducible Quint tests in /test folder

Design

1. Design model (Data design):
   - Applied design moves to the old Data Model from proj 3.1:
     - removing object (Figure 1): We removed the “Traveller” object, as we decided to focus on the “way-finding” functionality and we removed the social-network functionality (which included review a way, save a way)

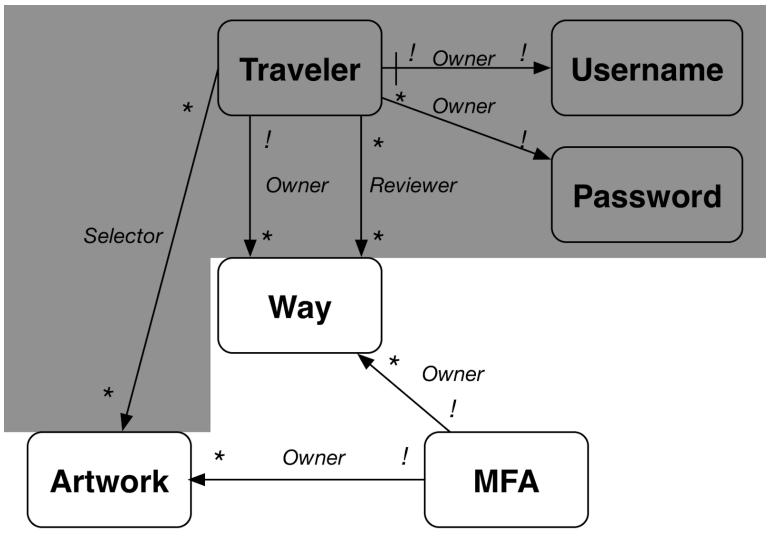

Figure 1. design move: removing objects.

- renaming objects: We renamed our “MFA” object into “Map” object, as we wanted to simplify the problem of finding ways on one floor of “MFA” object.
- adding objects: We added the “gallery” and “road” object.
    - “gallery”: Previously, we designed “artwork” objects to have their own physical locations. To simplify the “Traveling Salesman Problem” to find a way (to be implemented later), we only cared about the location of the gallery that the artworks are in. Also, with the new “gallery” object, we can easily add/update/del all the “artwork” objects that are in the same physical gallery
    - “road”: Not all “gallery” objects are connected to each other. In order to find a feasible way, we need the “road” object to record the connection between pairs of galleries.

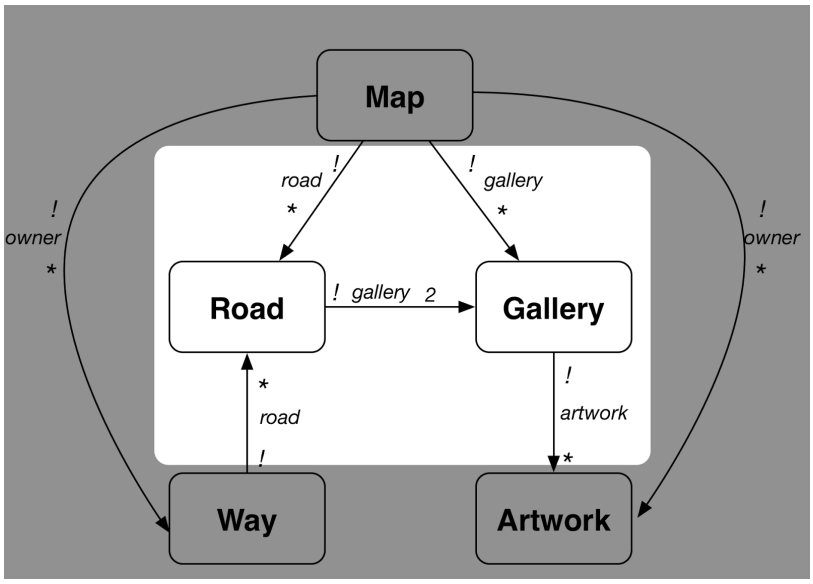

Figure 2. design move: adding objects.

- Contours of the Data Model:
    - Gallery: a gallery contains a list of artwork ids
    - Road: a road contains two gallery ids
    - Way: a way contains a list of road ids
    - Map: a map contains a list of road ids and a list of gallery ids

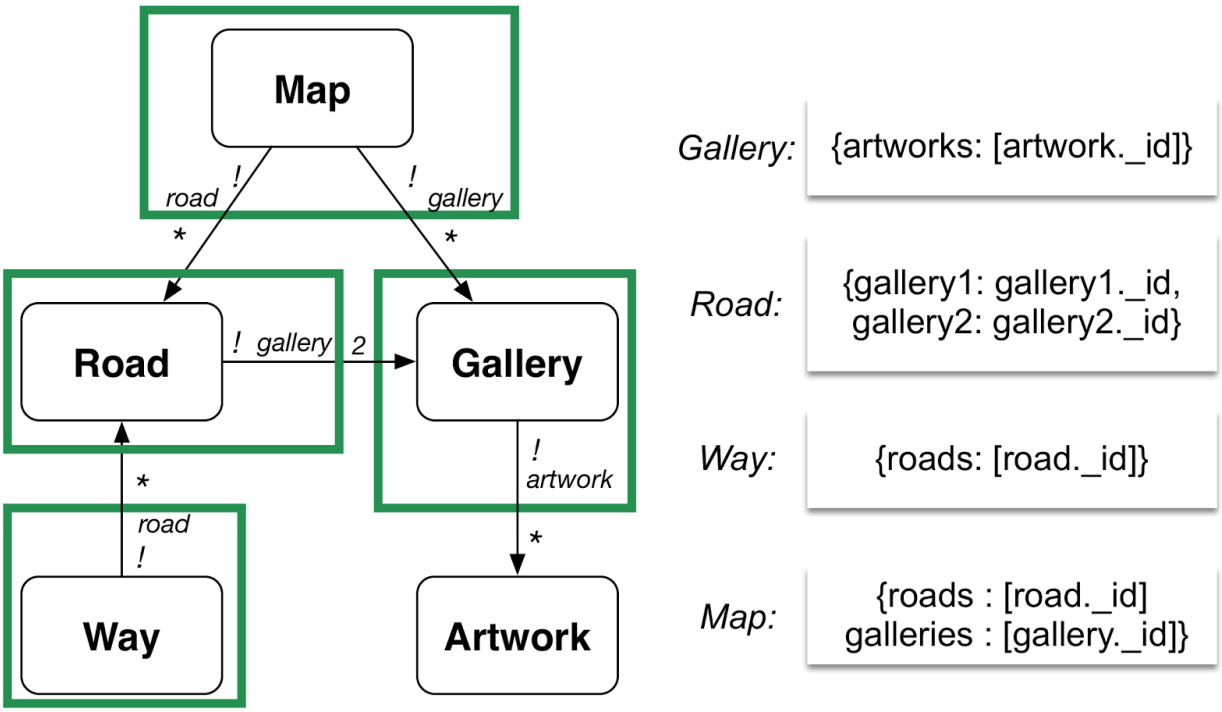

Figure 3. contours on the final data model and the implementation.

2. Design challenges:

- Challenge 1: How to design objects to enable route planning?
    - our choice: We needed to represent the graph of artworks, so that we can run the use “Traveling Salesman Problem” solver to output the shortest path to cover the artworks in the query. Thus, we add “gallery” (node in the graph) and “road” (edge in the graph) object into our data model.
- Challenge 2: How to represent the location information of the artworks?
    - choice 1: every “artwork” object has the location information
    - choice 2: every ”artwork” object is contained in a “gallery” object that has the location information
    - our choice: We choose option 1 to simplify the “Traveling Salesman Problem” to find a way (to be implemented later)
- Challenge 3: How to design the a “map/way/road” object?
    - choice 1: light-weight (normalized), only contains references to the contained objects
    - choice 2: heavy-weight (embedded), contains the associated objects themselves
    - our choice: We choose option 1, as we want a normalized data model where we can easily update individual objects without much side effects.
- Challenge 4: Whether to include the “Traveler” object?
    - choice 1: yes. implement save/review “way” functionality for travellers

- choice 2: no.
- our choice: We choose option 2, as we want to focus on the purpose of the app, which is to provide an interface for users to find a way based on the artworks that they are interested in.

Application Programmer's Interface:

1. Find a way based on the url query artworks
   1.1. Currently implemented as a dummy method, will return just a way regardless of input.
   1.2. This method can also be found in the more detailed API below in Way.

- **Way**
  - **POST**:
    - **/mfa/way/search/artworks_id/**
      - Request body:
        - ids: the list of artwork ids
      - Response:
        - success: true if successfully found a way connecting all artworks minimizing total distance
        - content: json file of the way
        - err: error message

2. Manipulate (get/add/update/del) objects in the database. There are 2 show methods, 2 delete methods, 1 update method, and 1 delete method for Artwork, Gallery, Way, Map. When there are two methods, one refers to the specific (show a particular artwork) and the other refers to the general (show all artworks). Road has all but the update method, as updating the edge involves a remove and an add, which feels more natural to us to be called separately).

- **Artwork**:
  - **GET**:
    - **/mfa/artwork/id=...**
      - Request parameters:
        - id: a string representation of the desired artwork's _id
      - Response:
        - success: true if this artwork is in database
        - content: json file of the requested artwork
        - err: error message
    - **/mfa/artwork/**
      - Request parameters:
        - null
      - Response:
        - success: displays all artworks in the database
        - content: json file of all artworks

            - err: error message
        - **/mfa/artwork/delete**
            - Request body:
                - null
            - Response:
                - success: true if successfully deleted all artworks
                - content: null
                - err: error message

- **POST**:
    - **/mfa/artwork/new/**
        - Request body:
            - id: the string representation of the ID of the artwork
            - access_number: the access number of the artwork
            - gallery: the room number of the gallery the artwork is in
        - Response:
            - success: true if new artwork created
            - content: null
            - err: error message
    - **/mfa/artwork/update/update**
        - Request body:
            - content: content of the fields to be changed (other fields can be null, id is required)
                - id: the _id of the artwork in question
                - name: the new name for the artwork
                - access_number: the new access number
        - Response:
            - success: true if successfully updated the artwork
            - content: json file of the modified artwork
            - err: error message
    - **/mfa/artwork/delete/id**
        - Request body:
            - id: the _id of the artwork to be deleted
        - Response:
            - success: true if successfully deleted artwork with given id in database
            - content: null
            - err: error message

- **Gallery**:
    - **GET**
        - **/mfa/gallery/**
            - Request parameters:

        - null
      - Response:
        - success: true if galleries in the database
        - content: json file of all galleries
        - err: error message
    - **/mfa/gallery/delete**
      - Request parameters:
        - null
      - Response:
        - success: true if successfully deleted all galleries
        - content: null
        - err: error message

  - **/mfa/gallery/id=...**
    - Request parameters:
      - id: a string representation of the desired gallery's _id
    - Response:
      - success: true if this gallery is in database
      - content: json file of the requested gallery
      - err: error message

- **POST**
  - **/mfa/gallery/new**
    - Request body:
      - name: the room number of the gallery
      - x: the x coordinate of the gallery chamber
      - y: the y coordinate of the gallery chamber
      - artworks: a list of artwork in the gallery, separated by ','
    - Response:
      - success: true if new gallery created
      - content: a JSON file for the gallery created
      - err: error message
  - **/mfa/gallery/delete/id**
    - Request body:
      - id: the _id of the gallery to be deleted
    - Response:
      - success: true if successfully deleted gallery with given id in database
      - content: null
      - err: error message
  - **/mfa/gallery/update/**
    - Request body:

            - content: content of the fields to be changed (other fields can be null, id is required)
                - id: the _id of the gallery in question
                - x: the new x position of the gallery
                - y: the new y position of the gallery
                - artworks: the new list of artworks in the gallery, overwrites old one
        - Response:
            - success: true if successfully updated the gallery
            - content: json file of the modified gallery
            - err: error message

- **Road**:
    - **GET**:
        - **/mfa/road/id=...**
            - Request parameters:
                - id: a string representation of the desired gallery's _id
            - Response:
                - success: true if this gallery is in database
                - content: json file of the requested gallery
                - err: error message
        - **/mfa/road/**
            - Request parameters:
                - null
            - Response:
                - success: true if roads in the database
                - content: json file of roads
                - err: error message
        - **/mfa/road/delete**
            - Request body:
                - null
            - Response:
                - success: true if successfully deleted all roads
                - content: null
                - err: error message
    - **POST**:
        - **/mfa/road/delete/id**
            - Request body:
                - id: the _id of the road to be deleted
            - Response:
                - success: true if successfully deleted road with given id in database

            - content: null
            - err: error message
        - **/mfa/road/new**
            - Request body:
                - gallery1: the id of the first gallery chamber
                - gallery2: the id of the second gallery chamber
            - Response:
                - success: true if new road created
                - content: a JSON file for the road created
                - err: error message
- **Way:**
    - **GET:**
        - **/mfa/way/id=...**
            - Request parameters:
                - id: a string representation of the desired way's _id
            - Response:
                - success: true if this way is in database
                - content: json file of the requested artwork
                - err: error message
        - **/mfa/way/delete**
            - Request body:
                - null
            - Response:
                - success: true if successfully deleted all ways
                - content: null
                - err: error message
        - **/mfa/way/**
            - Request parameters:
                - null
            - Response:
                - success: true if ways in the database
                - content: json file of artworks
                - err: error message
    - **POST**:
        - **/mfa/way/search/artworks_id/**
            - Request body:
                - ids: the list of artwork ids
            - Response:
                - success: true if successfully found a way connecting all artworks minimizing total distance
                - content: json file of the way
                - err: error message
        - **/mfa/way/update/**

- Request body:
  - content: content of the fields to be changed (other fields can be null, id is required)
  - name: the name of the way
  - description: Some explanation of the way
  - roads: a list of edges, in order, of the way
- Response:
  - success: true if successfully updated the artwork
  - content: json file of the modified artwork
  - err: error message

/mfa/way/delete/id

- Request body:
  - id: the _id of the way to be deleted
- Response:
  - success: true if successfully deleted way with given id in database
  - content: null
  - err: error message

/mfa/way/new

- Request body:
  - name: the name of the way
  - description: Some explanation of the way
  - roads: a list of edges, in order, of the way
- Response:
  - success: true if new way created
  - content: null
  - err: error message

- **Map**:
  - **POST**
    - **/mfa/map/new**
      - Request body:
        - image: the image for the map
        - name: the name of the map
        - description: some description of the map
        - roads: list of roads in the map
        - galleries: list of galleries in the map
      - Response:
        - success: true if new map created
        - content: a JSON file for the map created
        - err: error message
    - **/mfa/gallery/delete/id**
      - Request body:
        - id: the _id of the gallery to be deleted

        - Response:
            - success: true if successfully deleted gallery with given id in database
            - content: null
            - err: error message
    - **/mfa/map/update/**
        - Request body:
            - content: content of the fields to be changed (other fields can be null, id is required)
                - id: the _id of the map in question
                - image: the new image for the map, overwrites old one
                - name: the name of the map
                - description: some description of the map
                - roads: list of roads in the map
                - galleries: list of galleries in the map
        - Response:
            - success: true if successfully updated the map
            - content: json file of the modified map
            - err: error message