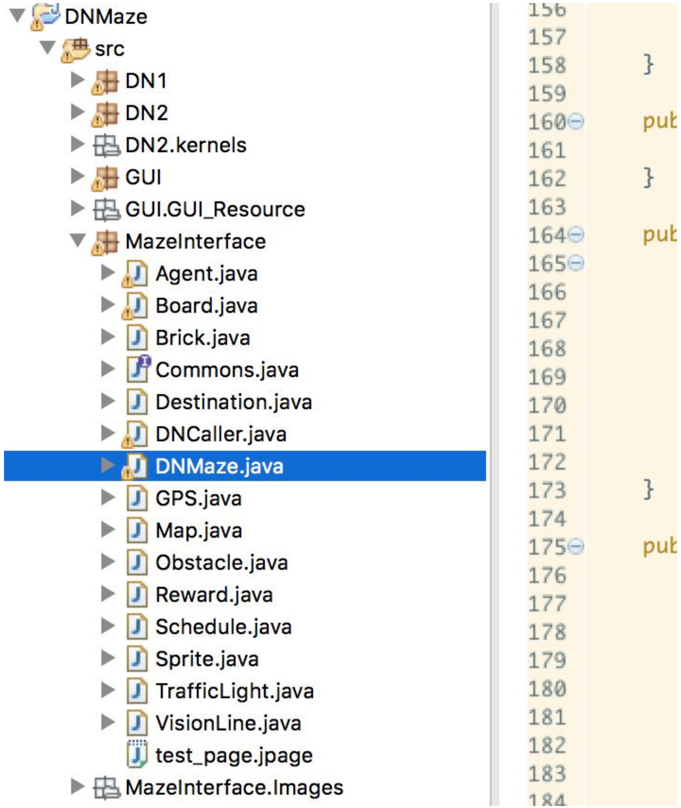
Maze Navigation

Running the maze simulation

Right click DNMaze.java

Run as java application

(make sure use_socket_gui_flag is false in Commons.java)



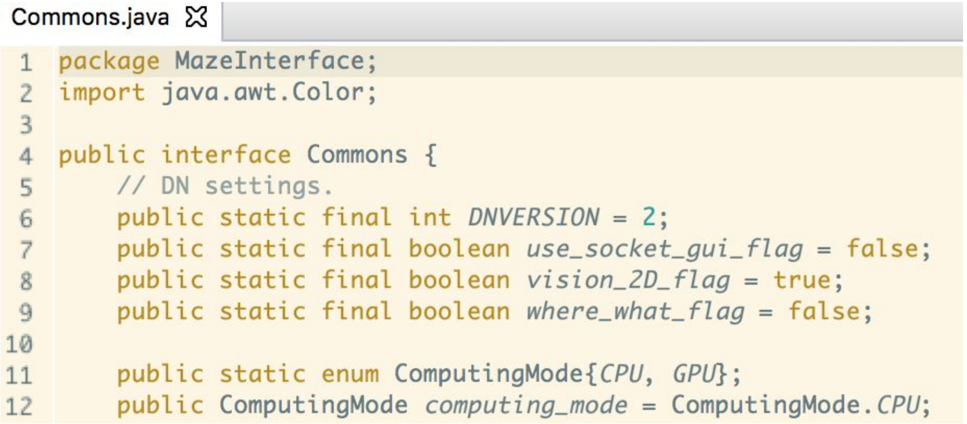
```
▼ DNMaze
  ▼ src
    ▶ DN1
    ▶ DN2
    ▶ DN2.kernels
    ▶ GUI
    ▶ GUI.GUI_Resource
    ▼ MazeInterface
      ▶ Agent.java
      ▶ Board.java
      ▶ Brick.java
      ▶ Commons.java
      ▶ Destination.java
      ▶ DNCaller.java
      ▶ DNMaze.java
      ▶ GPS.java
      ▶ Map.java
      ▶ Obstacle.java
      ▶ Reward.java
      ▶ Schedule.java
      ▶ Sprite.java
      ▶ TrafficLight.java
      ▶ VisionLine.java
        test_page.jpage
    ▶ MazeInterface.Images
```

Some settings: CPU mode or GPU mode

Change computing mode accordingly

Need to set up JOCL for GPU computation (current project set for Mac OS)

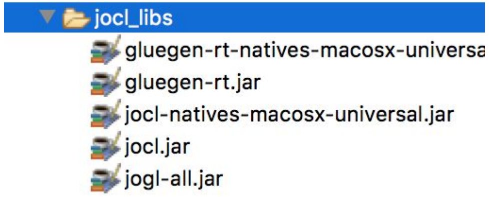
JOCL: http://jogamp.org/jocl/www/


Commons.java

```java
package MazeInterface;
import java.awt.Color;

public interface Commons {
    // DN settings.
    public static final int DNVERSION = 2;
    public static final boolean use_socket_gui_flag = false;
    public static final boolean vision_2D_flag = true;
    public static final boolean where_what_flag = false;

    public static enum ComputingMode{CPU, GPU};
    public ComputingMode computing_mode = ComputingMode.CPU;
```


jocl_libs
- gluegen-rt-natives-macosx-universa
- gluegen-rt.jar
- jocl-natives-macosx-universal.jar
- jocl.jar
- jogl-all.jar

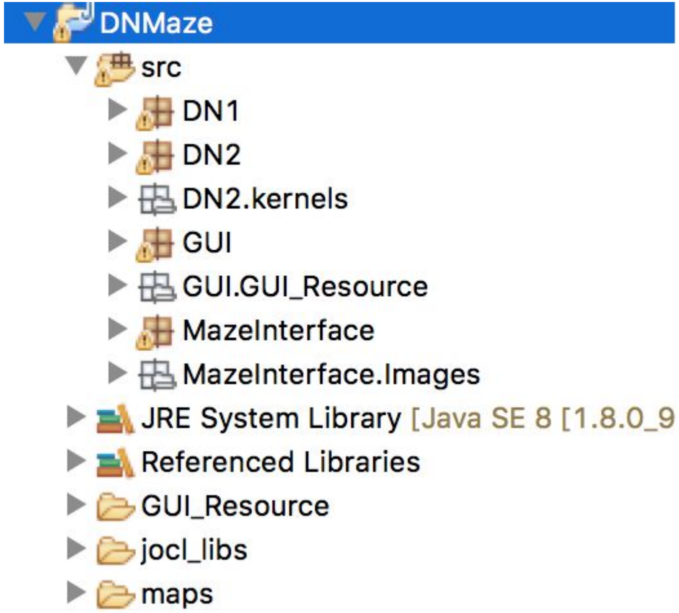
Project Structure

Maze interface has the maze objects

DN2 has the DN library

Kernels contains code for GPU computation

Maps contains the individualized maze maps and teaching schedules



```
▼ DNMaze
  ▼ src
    ▶ DN1
    ▶ DN2
    ▶ DN2.kernels
    ▶ GUI
    ▶ GUI.GUI_Resource
    ▶ MazeInterface
    ▶ MazeInterface.Images
  ▶ JRE System Library [Java SE 8 [1.8.0_9
  ▶ Referenced Libraries
  ▶ GUI_Resource
  ▶ jocl_libs
  ▶ maps
```

How to understand the interface: training

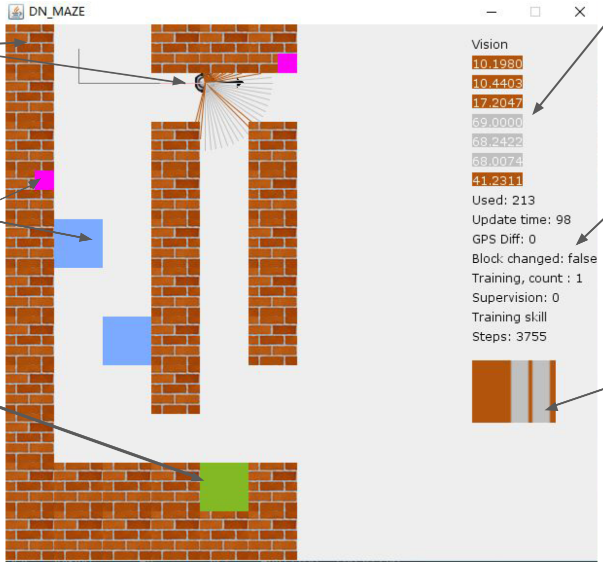
The agent

Current map (wall)

Current map (obstacle)

Current map (goal)

Current map (traffic light)



Vision
10.1980
10.4403
17.2047
69.0000
68.2422
68.0074
41.2311
Used: 213
Update time: 98
GPS Diff: 0
Block changed: false
Training, count : 1
Supervision: 0
Training skill
Steps: 3755

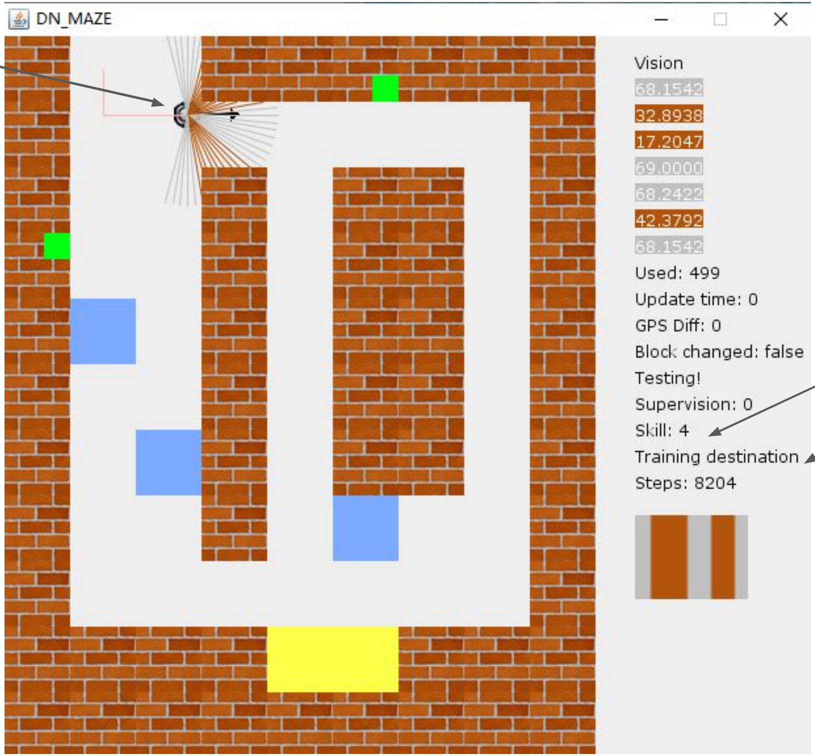
Selected vision line info (type, distance)

Training info (gps, block change, skill, steps, etc.)

2D vision image

How to understand the interface: testing chaining

Emergent action

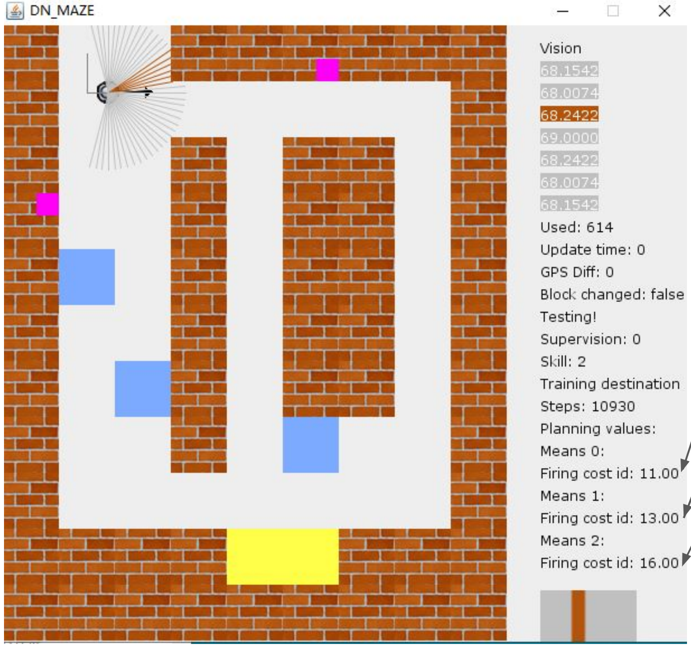


```
Vision
68.1542
32.8938
17.2047
69.0000
68.2422
42.3792
68.1542
Used: 499
Update time: 0
GPS Diff: 0
Block changed: false
Testing!
Supervision: 0
Skill: 4
Training destination
Steps: 8204
```

Emergent skill

Supervising destination
Supervising cost

How to understand the interface: testing planning



```
Vision
68.1542
68.0074
68.2422
69.0000
68.2422
68.0074
68.1542
Used: 614
Update time: 0
GPS Diff: 0
Block changed: false
Testing!
Supervision: 0
Skill: 2
Training destination
Steps: 10930
Planning values:
Means 0:
Firing cost id: 11.00
Means 1:
Firing cost id: 13.00
Means 2:
Firing cost id: 16.00
```

Emergent cost for route 0

Emergent cost for route 1

Emergent cost for route 2

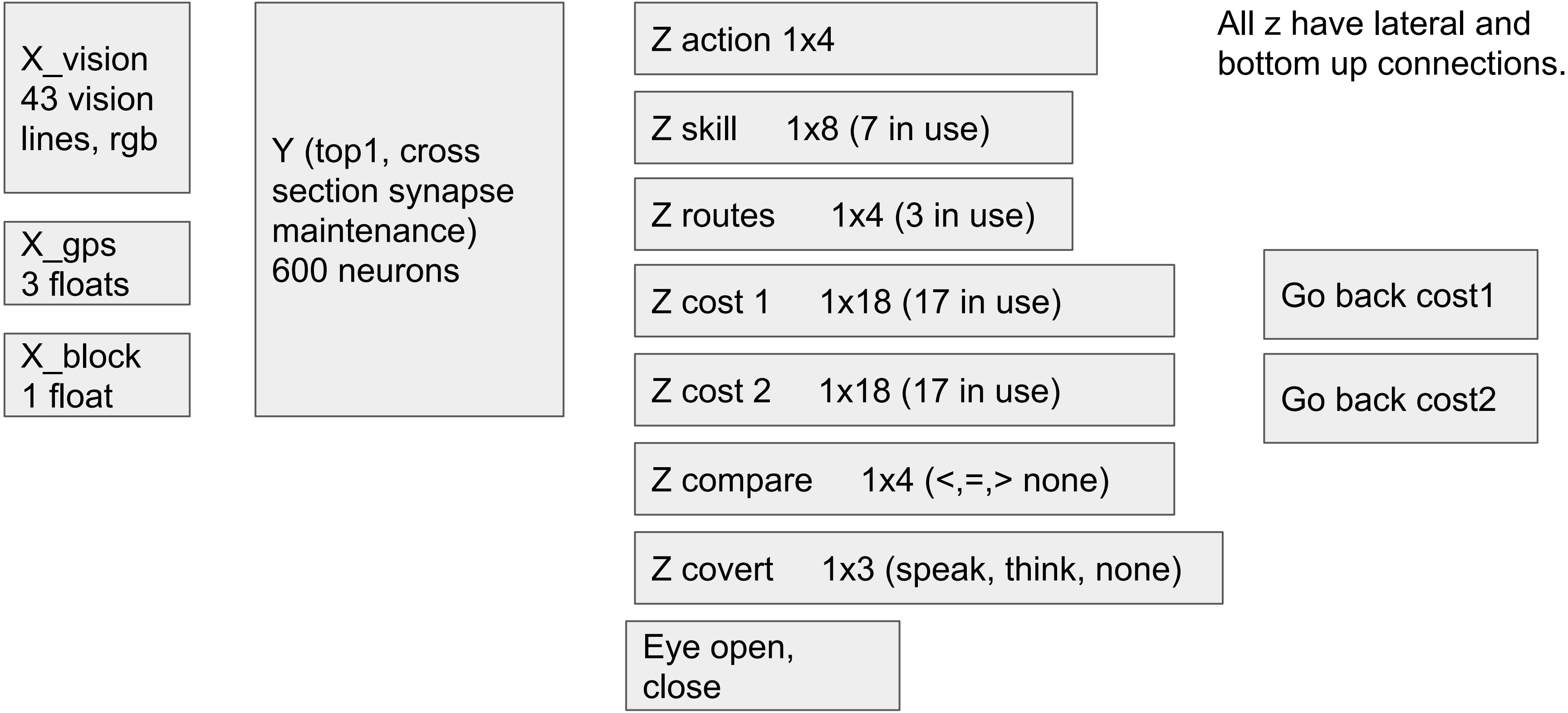
Network needs to choose one means from another by comparing cost of three routes

Settings

Environment related settings file : commons.java

DN-2 related parameters: DNCaller.java

Network



X_vision
43 vision
lines, rgb

X_gps
3 floats

X_block
1 float

Y (top1, cross
section synapse
maintenance)
600 neurons

Z action 1x4

Z skill 1x8 (7 in use)

Z routes 1x4 (3 in use)

Z cost 1 1x18 (17 in use)

Z cost 2 1x18 (17 in use)

Z compare 1x4 (<,=,> none)

Z covert 1x3 (speak, think, none)

Eye open,
close

All z have lateral and bottom up connections.

Go back cost1

Go back cost2

Training

1. (z_action, z_skill, z_means, z_cost1, z_cost2, z_compare, z_covert)
2. Train individual skills
   a. (action_t, skill_i, none, none, none, none, none) --X(t)--> (action_t+1, skill_i, none, none, none, none, none)
   b. After reach destination, go back: (action_t, back, none, none, none, none, none) --X(t)--> (action_t+1, back, none, none, none, none, none, none)
   c. Repeat a, b for all skills
3. Train routes (chaining skills together)
   a. Route_1:
      (action_t, emergent_t, route_1, cost_t, none, none, none) --X(t)-->
      (action_t+1, emergent_t+1, none, cost_t+1, none, none, none) (only supervise route_1 for 5 timesteps)
   b. After reach destination, go back:
      (action_t, back, back, none, none, none, none) --X(t)--> (action_t+1, back, back, none, none, none, none, none)
   c. Route_2:
      (action_t, emergent, route_2, none, cost_t, none, none) --X(t)-->
      (action_t+1, emergent, none, none, cost_t+1, none, none) (only supervise route_2 for 5 timesteps)
   d. Go back

Training

1. Train thinking state to thinking state:
   (none, none, none, none, none, none, thinking) --X_background-->
   (none, none, none, none, none, none, thinking)
2. Train comparison:
   (none, none, none, small_cost, high_cost, none, thinking) -- X_background →
   (none, none, means_1, none, none, comparison_result, thinking) ---
   (none, none, none, high_cost, small_cost, none, thinking) -- X_background →
   (none, none, none, means_2, none, comparison_result, speak)
3. Train planning:
   (close eye, skill_t, route_1, cost_1, none, none, thinking) --X_background-->
   (close eye, skill_t+1, route_1, cost_1, none, none, thinking)
   (action_t, skill_t, route_2, none, cost_2, none, thinking) --X_background-->
   (action_t+1, skill_t+1, route_2, none, cost_2, none, thinking)
4. Test planning:
   (close eye, skill_0, [1, 1], none, none, none, thinking) --X_background-->
   (emergent, emergent, emergent, emergent, emergent, emergent, emergent) -- (thinking)X_background --> (repeat)
   Until speak with route_i
   (emergent, emergent, route_i, emergent, emergent, emergent, emergent) -- X_(0) → (repeat)

Network performance

1. Successfully chained different tasks together based on the current context.
2. Successfully associated lower level skills with high level means.
3. Successfully learned comparison between different costs.
4. Successfully learned transition from thinking state to speaking state.
5. Successfully learned to keep thinking when no comparison result is available.
6. Successfully learned to plan and choose routes with lower cost in simulated environment