GLASGOW FANTASTIC IO

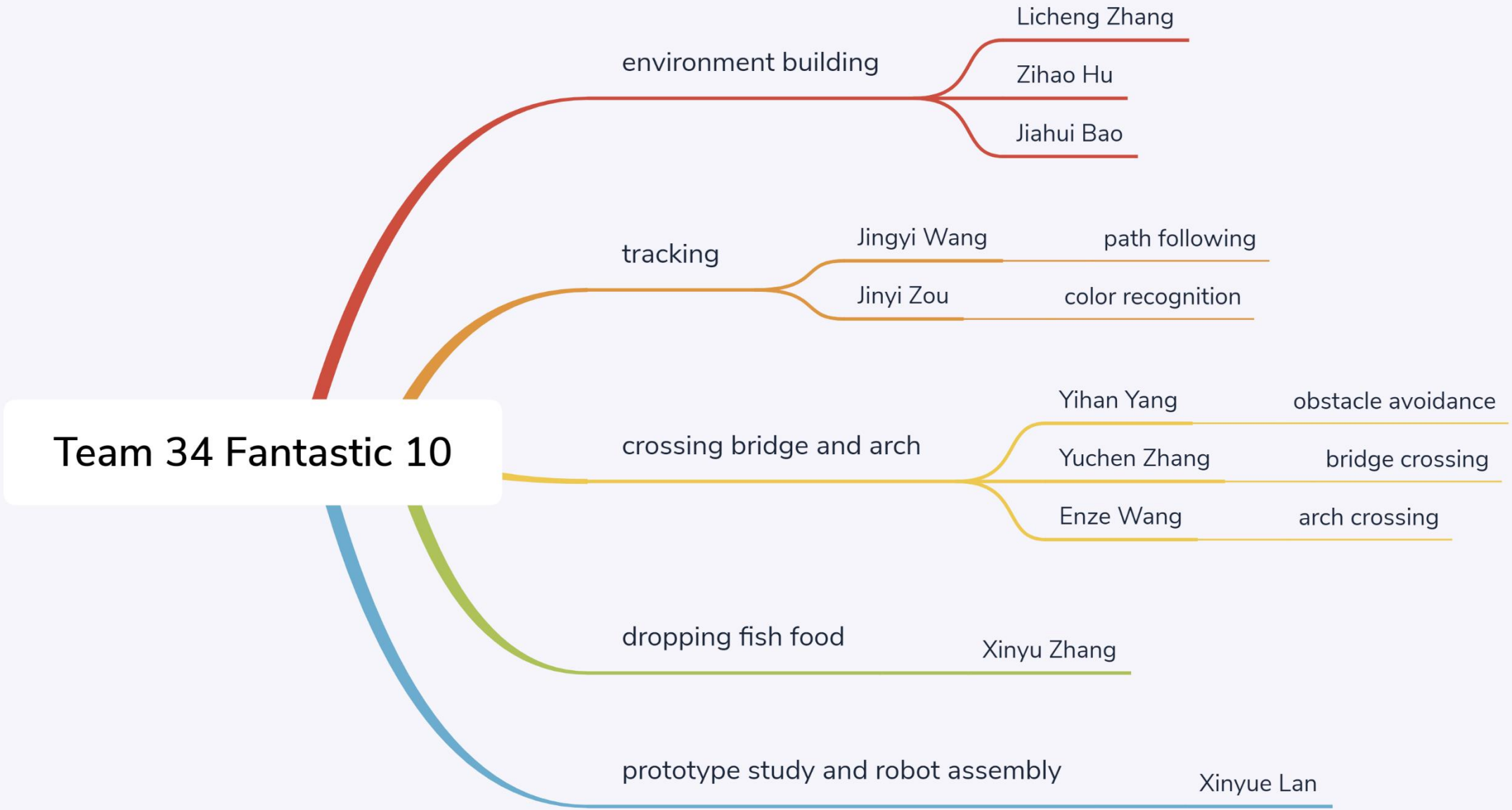
TEAM 34

Fantastic 10

Final Presentation

University of Glasgow

电子科技大学 University of Electronic Science and Technology of China


Team 34 Fantastic 10
environment building
Licheng Zhang
Zihao Hu
Jiahui Bao
tracking
Jingyi Wang
path following
Jinyi Zou
color recognition
crossing bridge and arch
Yihan Yang
obstacle avoidance
Yuchen Zhang
bridge crossing
Enze Wang
arch crossing
dropping fish food
Xinyu Zhang
prototype study and robot assembly
Xinyue Lan

CONTENTS

00 Overview

+

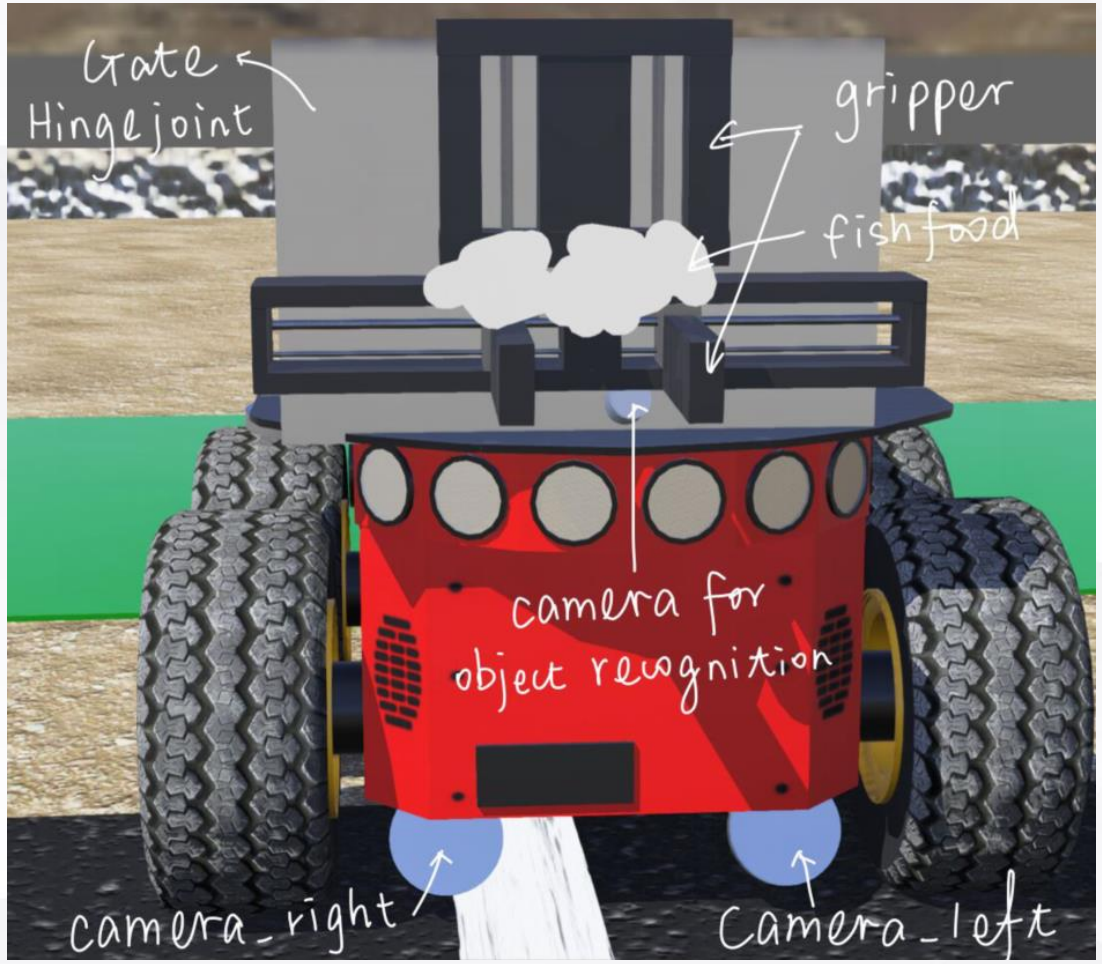
01-10 Personal Presentation

00

Overview

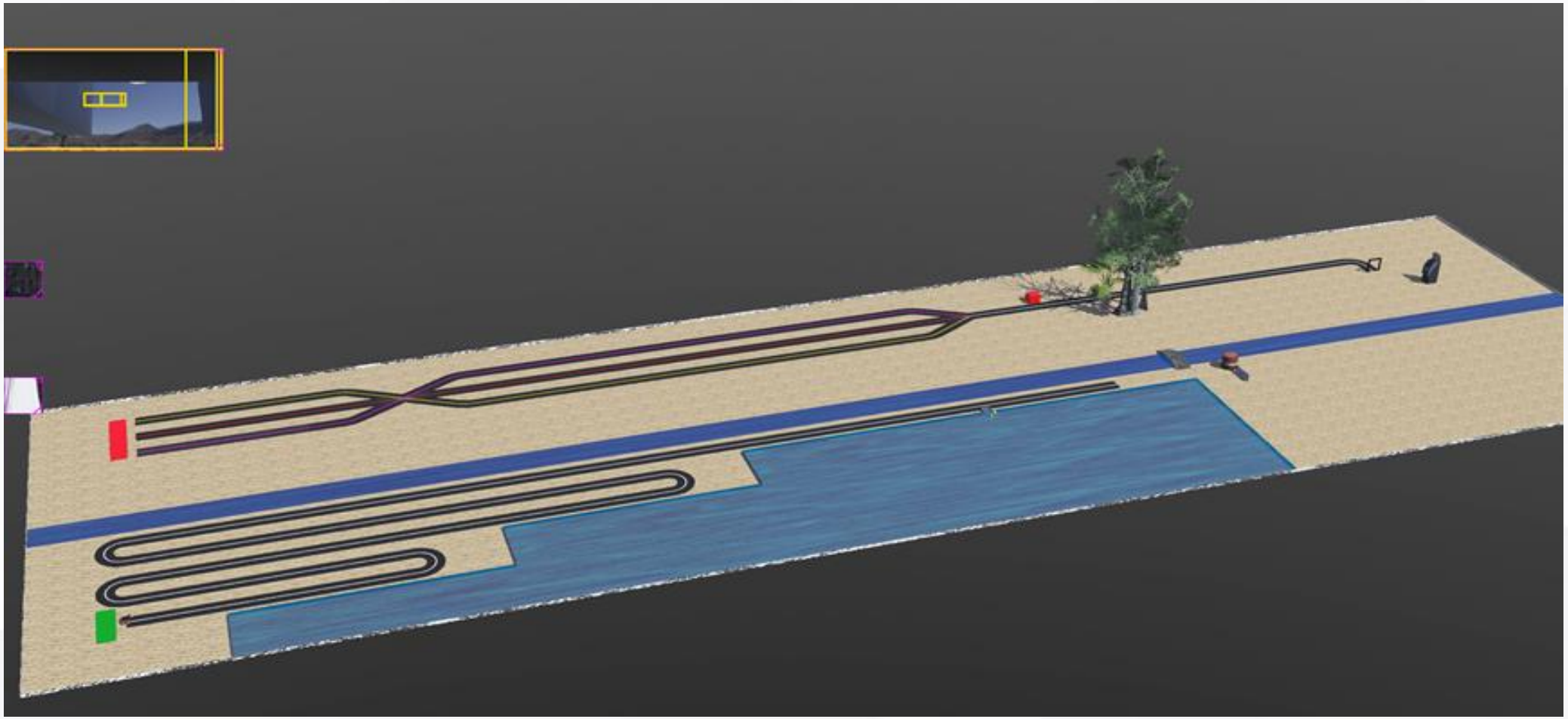
1.Final design of robot
2.Final design of patio

00 Final design of Robot



*A compass is equipped inside the rover

00 Final design of Patio

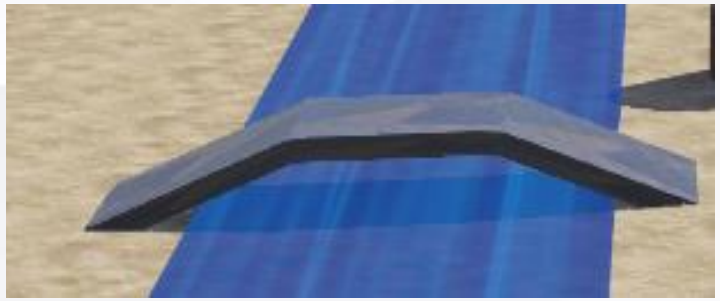



01
Environment
Building
Jiahui Bao

01

Environment Building

Contribution (Mainly in Task3&4)



Bridge

02

River

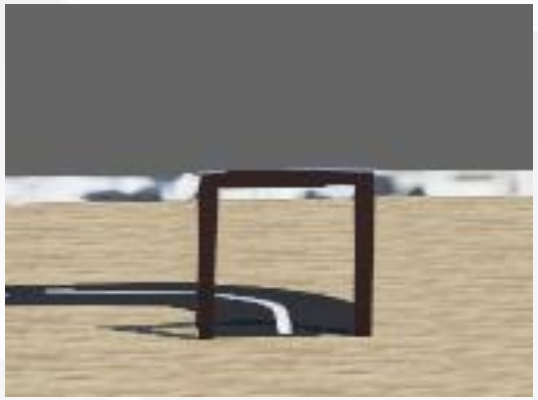
01

03

Trees

04

Arch

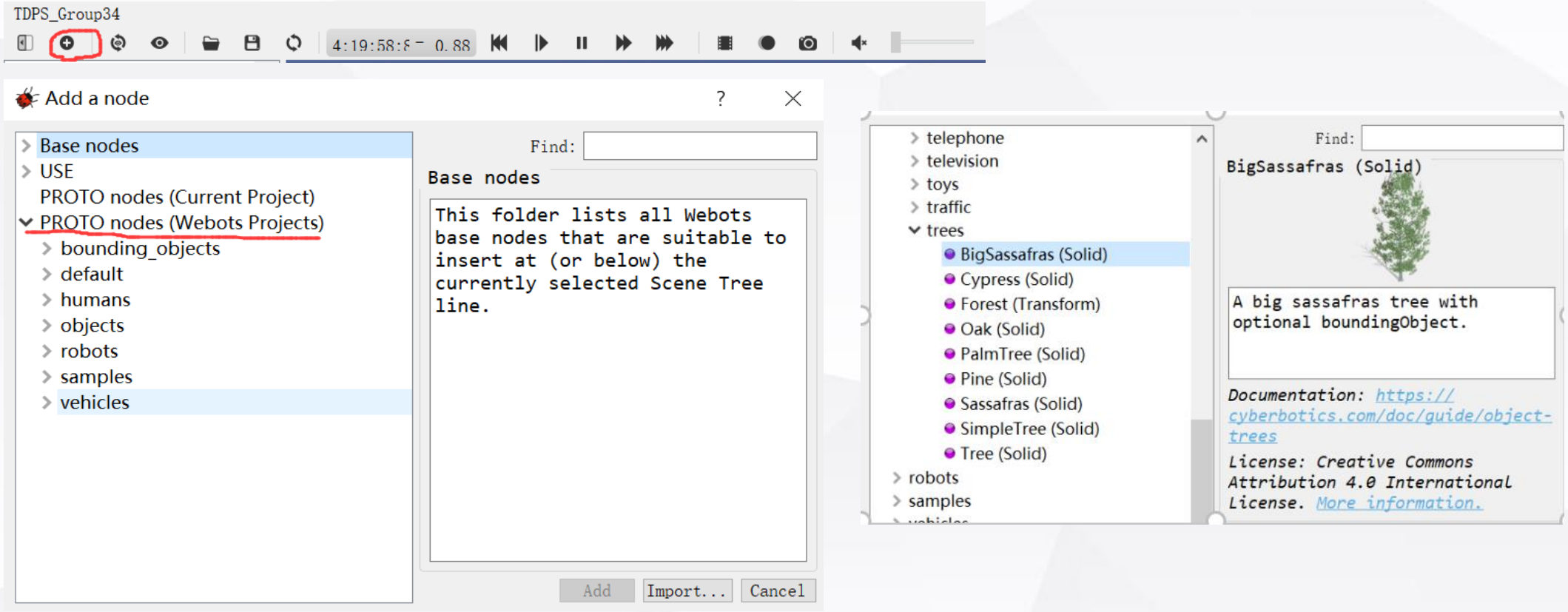


01

Environment Building

Method of building Objects

1. Look up the Webots repository directly :

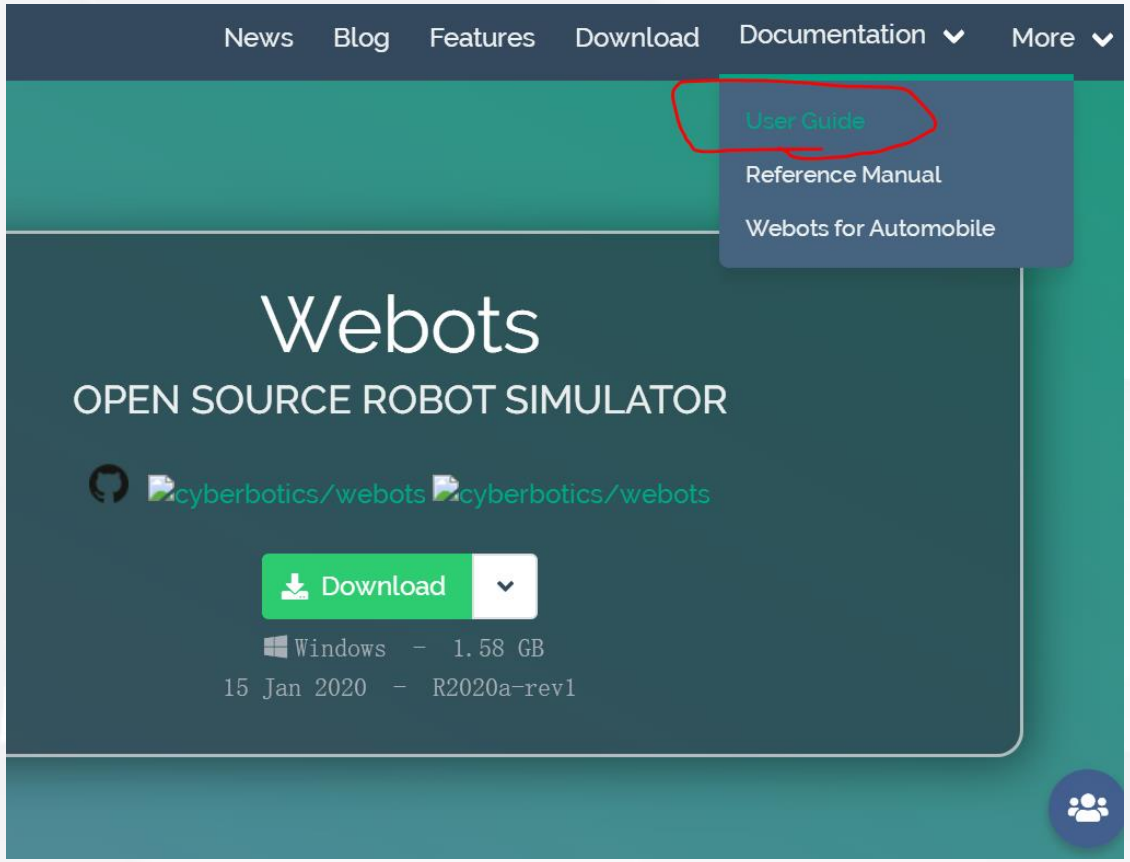


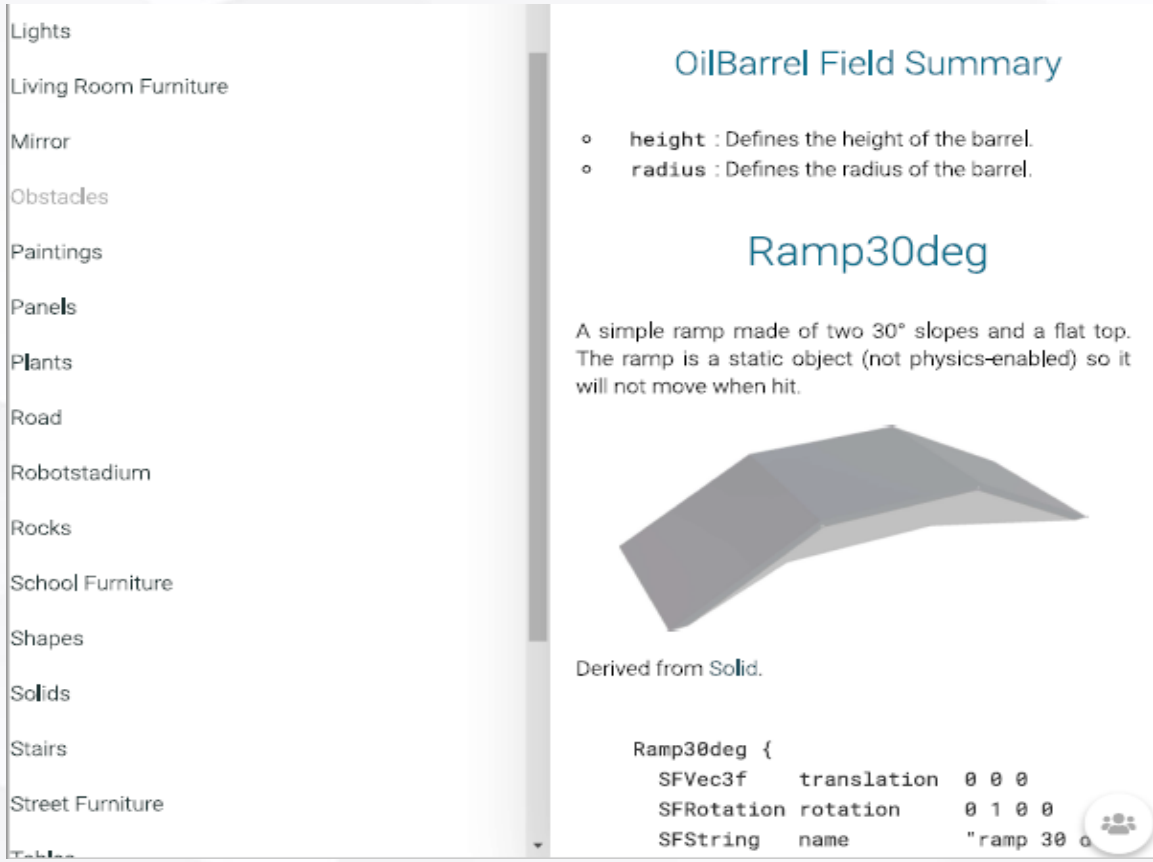
Find and select the objects we want through different categories.

01 Environment Building

Method of building Objects

2. Look up the Webots website: https://www.cyberbotics.com/



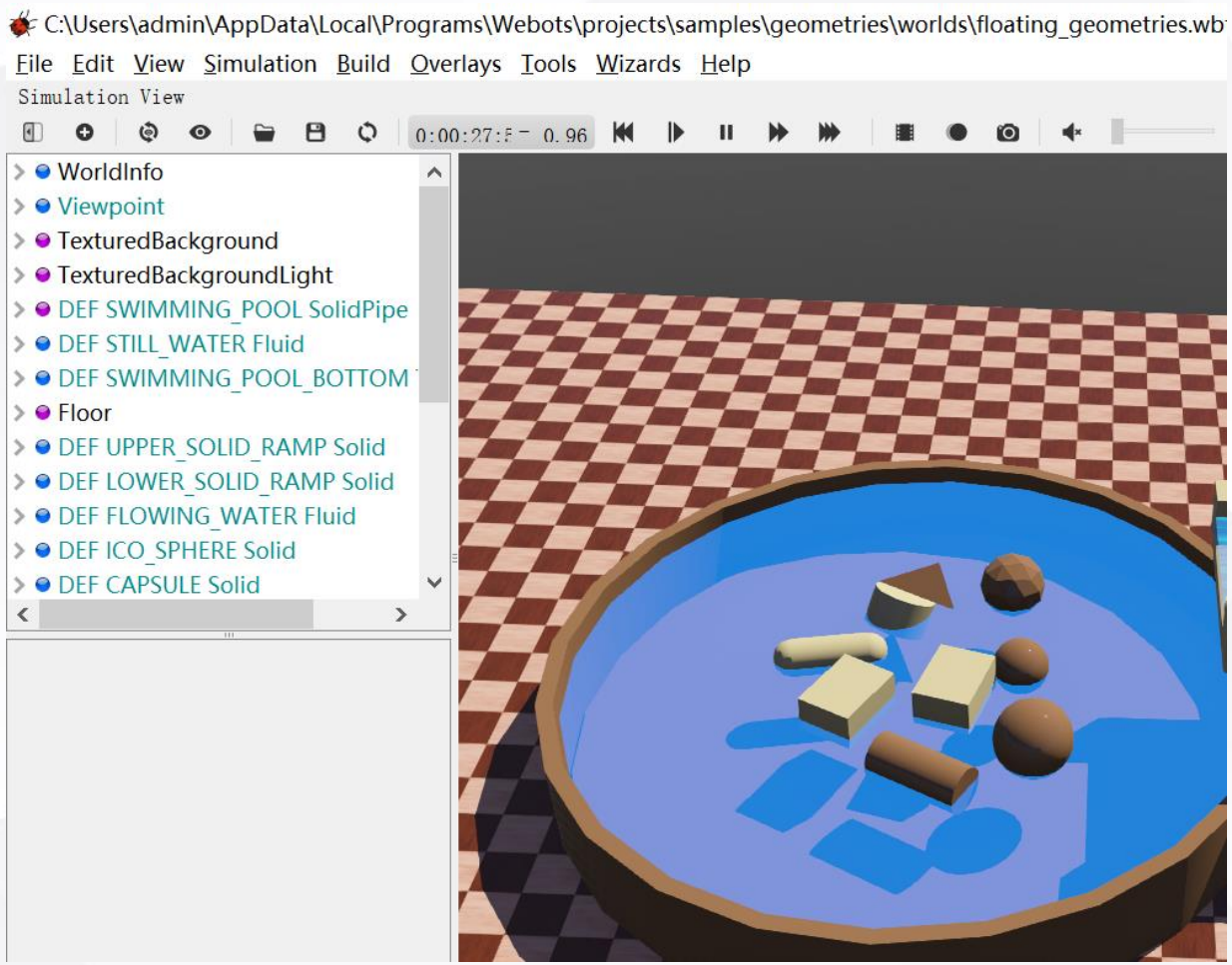


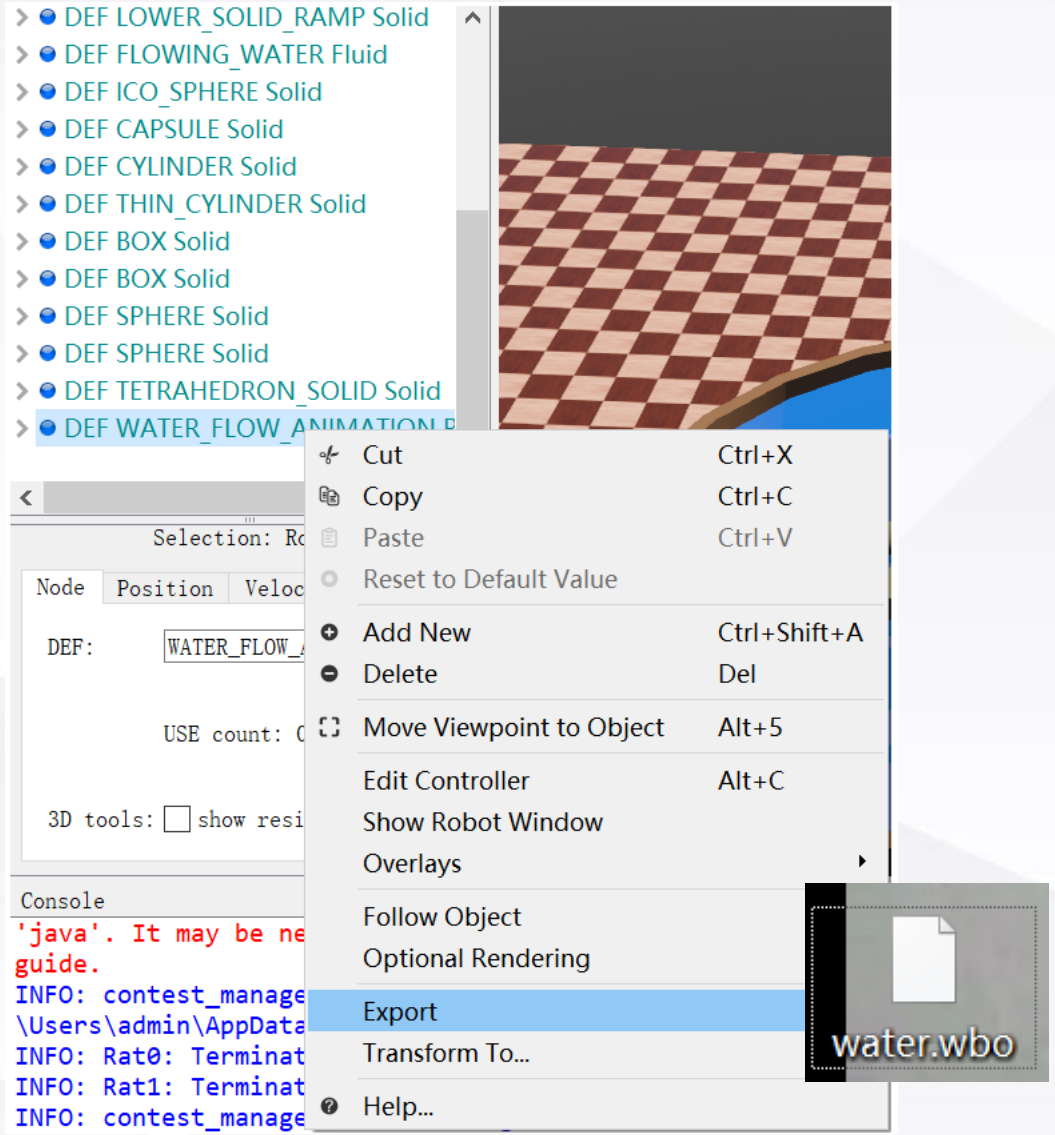
01

Environment Building

Method of building Objects

3. Look up the sample world:

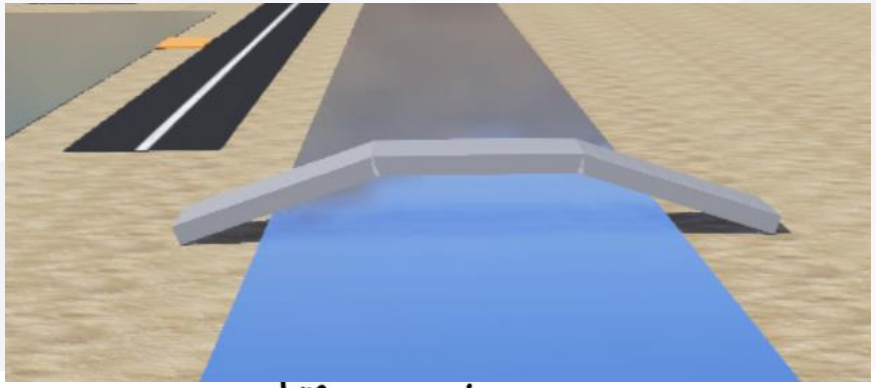


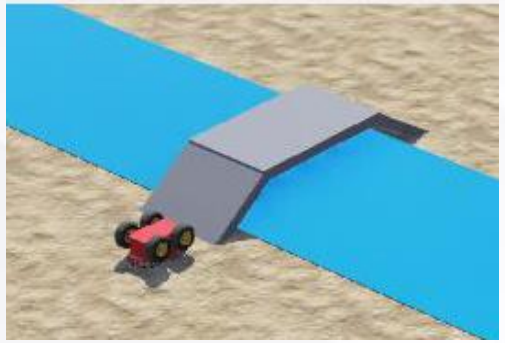


01

Environment Building

Building Bridge





1.The car could not make it on the bridge

↓ Increase initial velocity

2.The car overturned as it went down the bridge

Problems

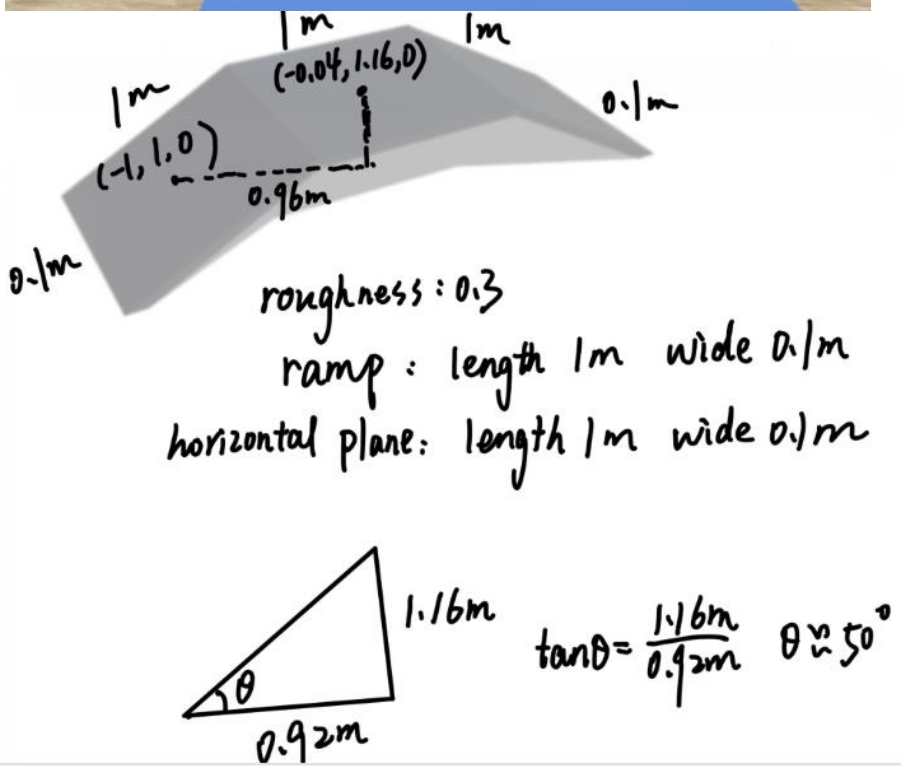
↓

Solutions

1.Increase the friction → Same material

Individual coefficient of friction cannot be simply changed

2.Reduce the tilt of the bridge




02
Environment
Building
Zihao Hu

02

Environment Building

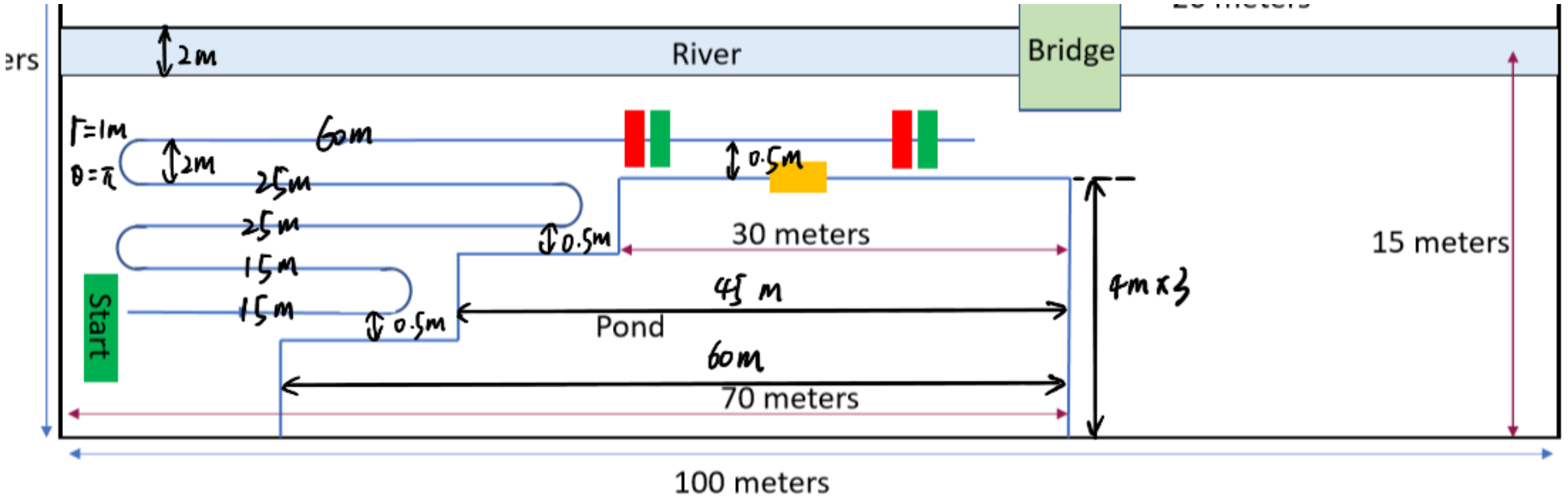
Distribution

Patio building (Route design) & Adjusting

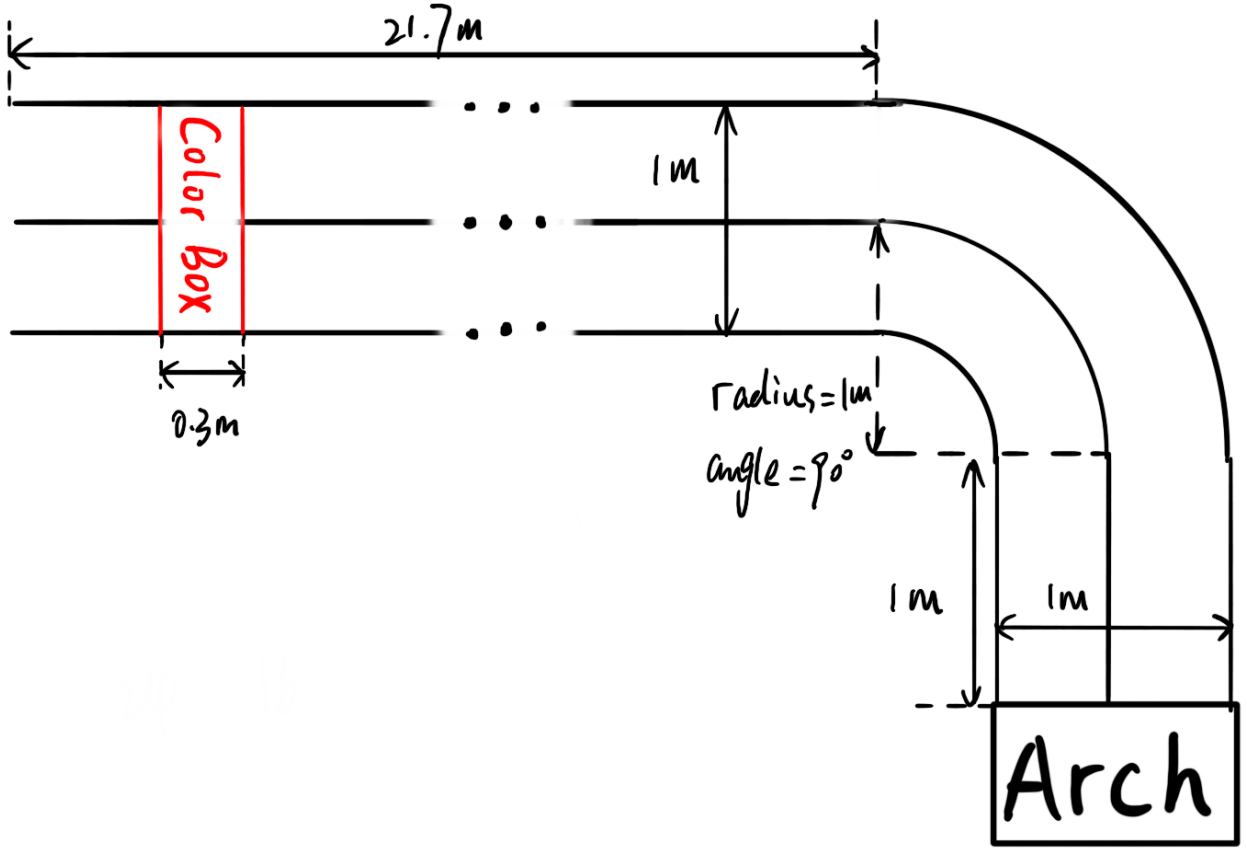
Debugging the assembled Pioneer 3-AT

1. Identify patio requirements & Draw drafts

a. Draft for Patio 1&2



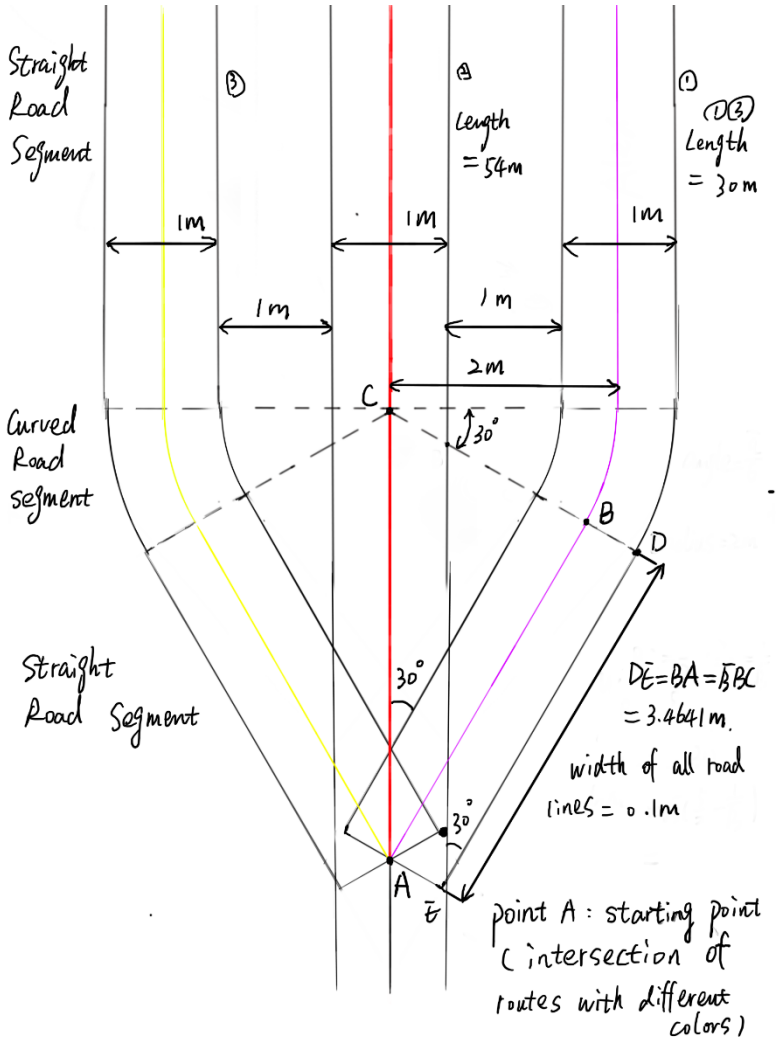
b. Draft for Patio 4

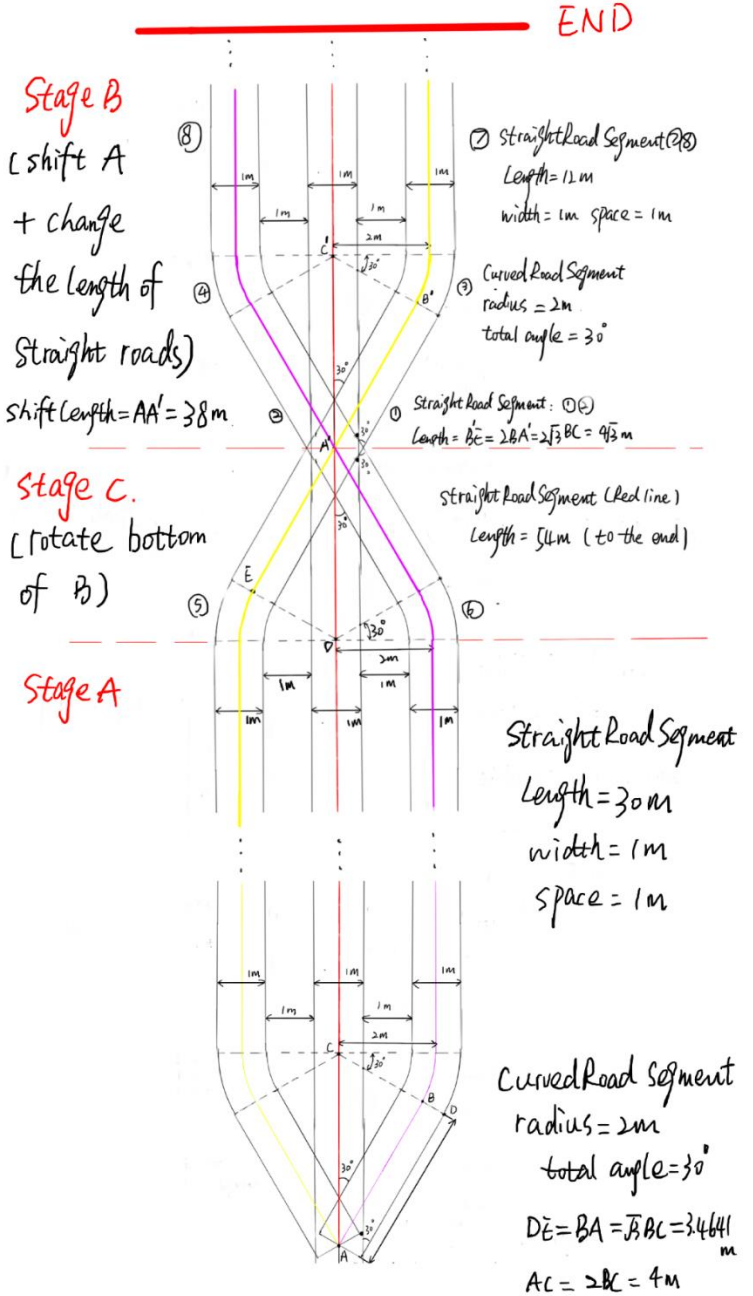


Environment Building

Route Design

c. Drafts for Patio 5



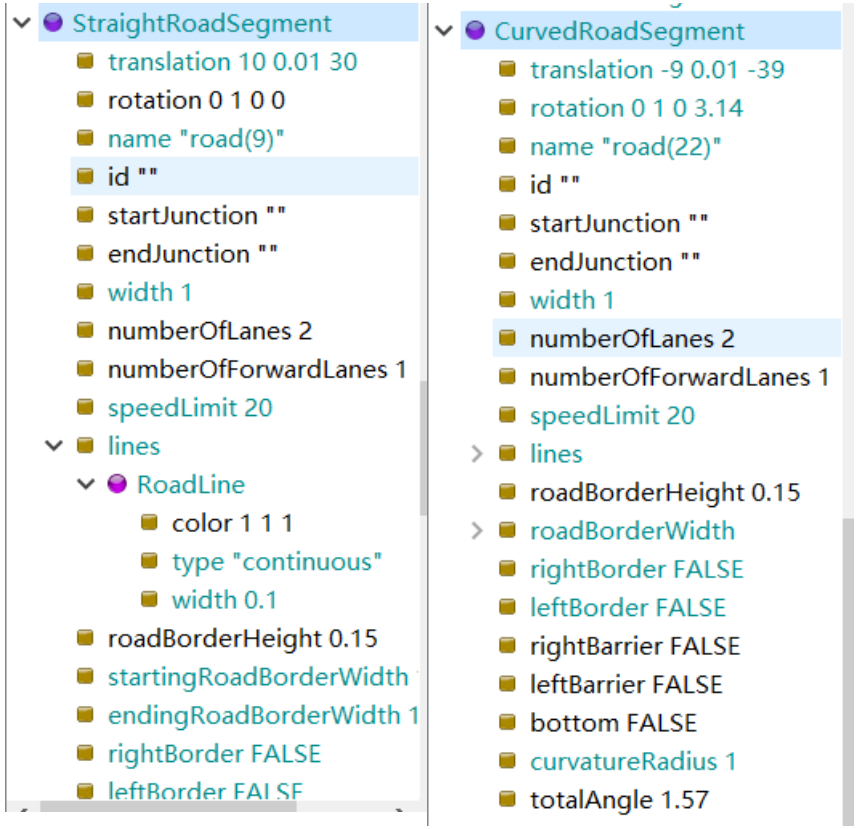


02

Environment Building

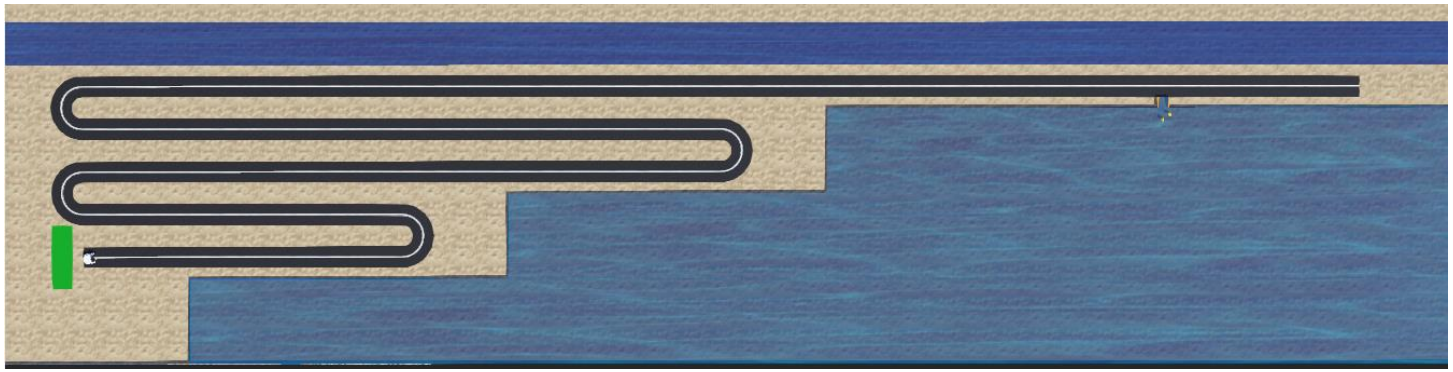
Patio Building & Adjusting

a. Patio 1&2



Scene tree

Approach



Final Patio 1&2

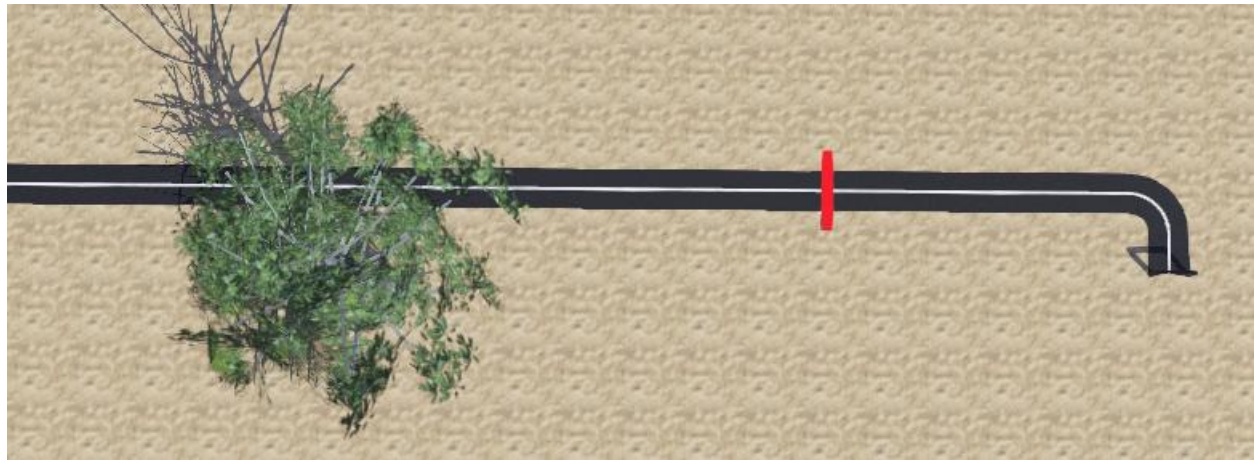
Environment Building

Patio Building & Adjusting

b. Patio 4

- DEF RED_BOX Solid
  - translation -10.9 0.25 -15
  - rotation 0 1 0 0
  - scale 1 1 1
  - children
    - Shape
      - appearance PBRAppearance
        - baseColor 1 0 1
        - baseColorMap NULL
        - transparency 0
        - roughness 0.2
        - roughnessMap NULL
        - metalness 0
        - metalnessMap NULL
        - IBLStrength 1
        - normalMap NULL
        - normalMapFactor 1
        - occlusionMap NULL
        - occlusionMapStrength 1
        - emissiveColor 0 0 0
        - emissiveColorMap NULL
        - emissiveIntensity 1
        - textureTransform NULL
        - name "PBRAppearance"
      - geometry DEF BOX0 Box
        - size 0.75 0.75 0.75
      - castShadows TRUE
      - isPickable TRUE

Scene tree



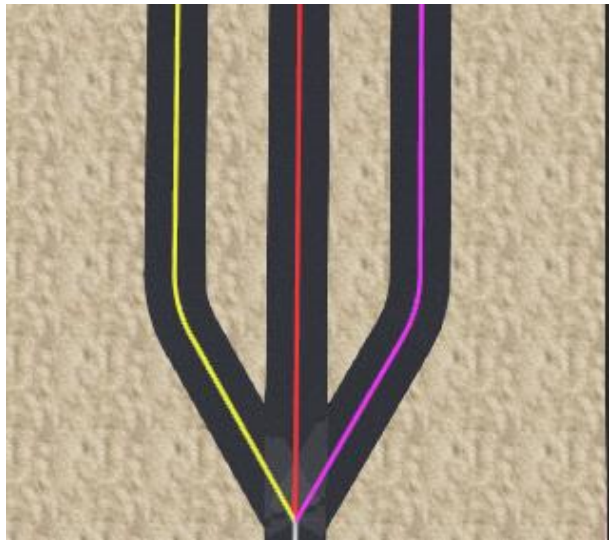
Patio 4

Patio 4 with adjustment

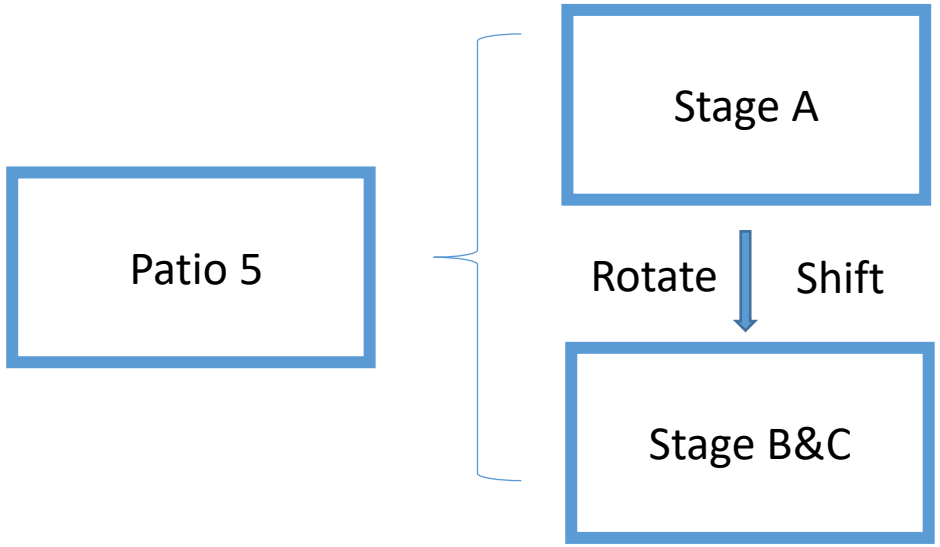
Environment Building

Patio Building & Adjusting

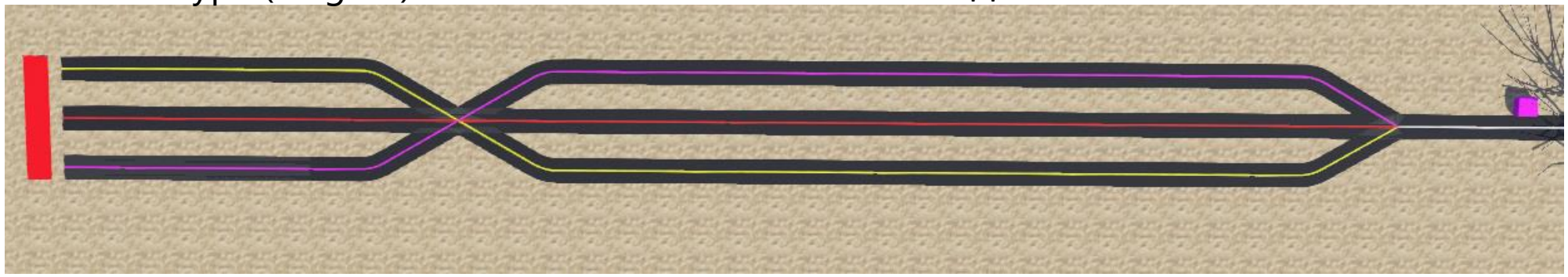
c. Patio 5



Prototype (Stage A)



Approach



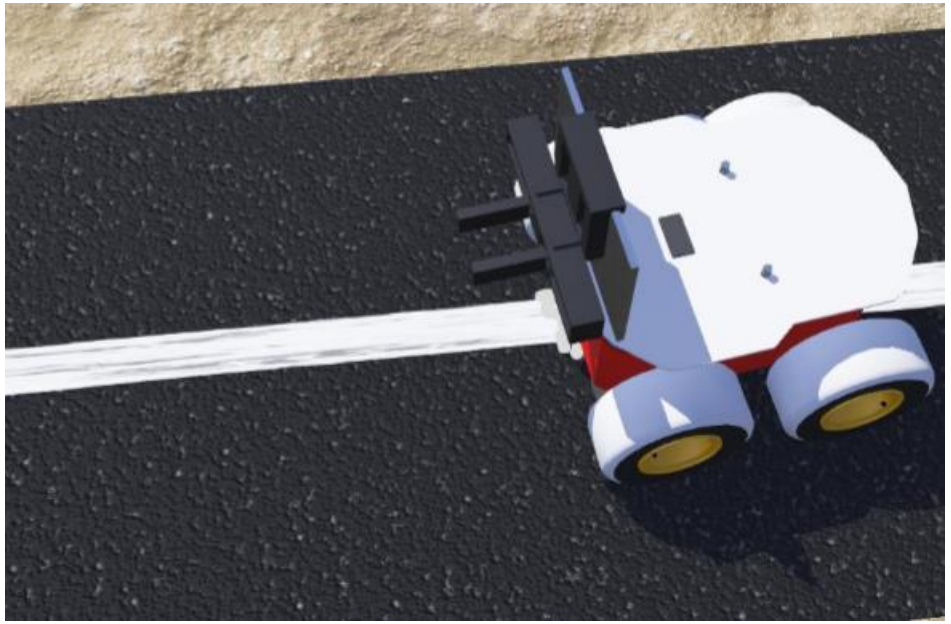
Final Patio 5

02

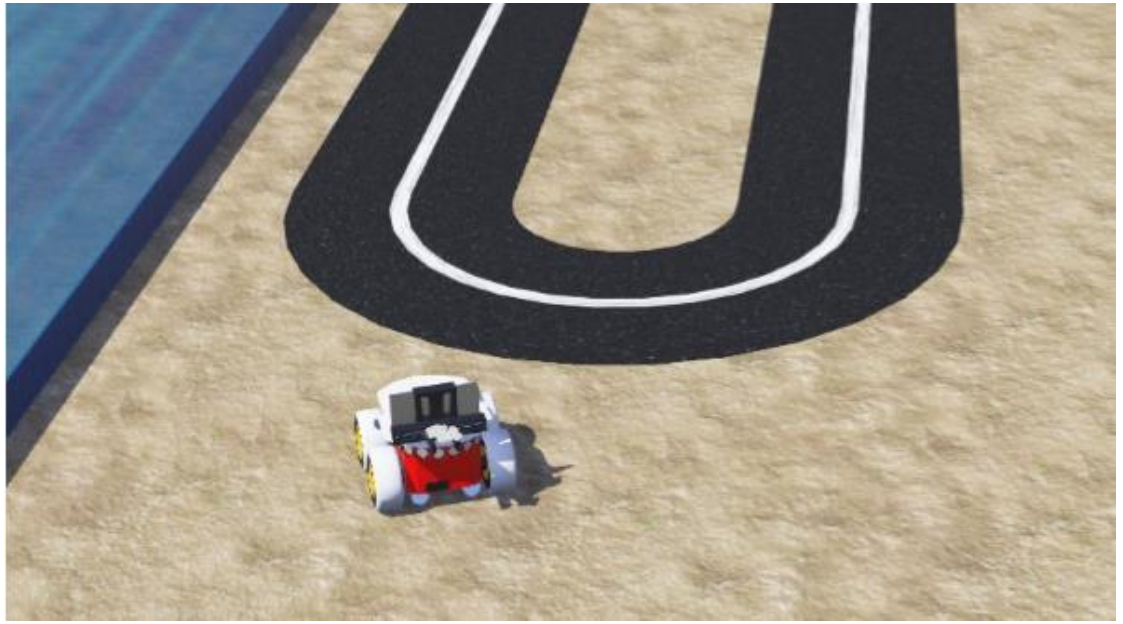
Environment Building

Debugging the equipped Pioneer 3-AT

Problem a:



Problem b:



Debug → Find Problems → Experiments → Communication & Optimization

Approach

Environment Building

Debugging the equipped Pioneer 3-AT

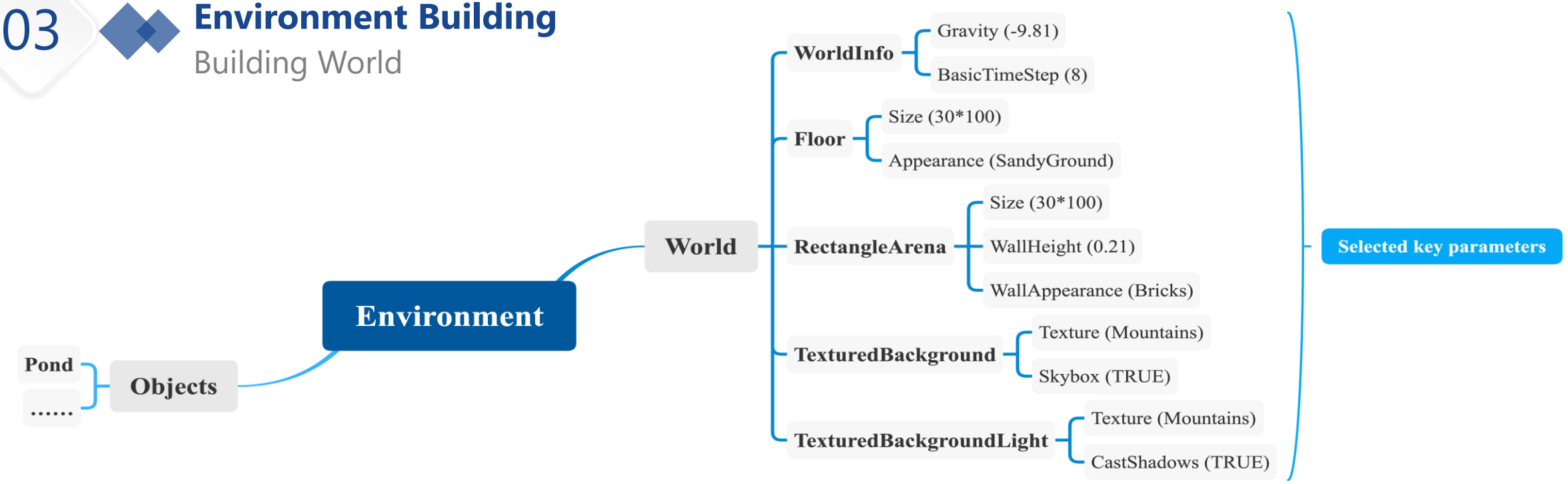
Mission completed!


03
Environment
Building & Camera
Recognition
Licheng Zhang

03

Environment Building

Building World



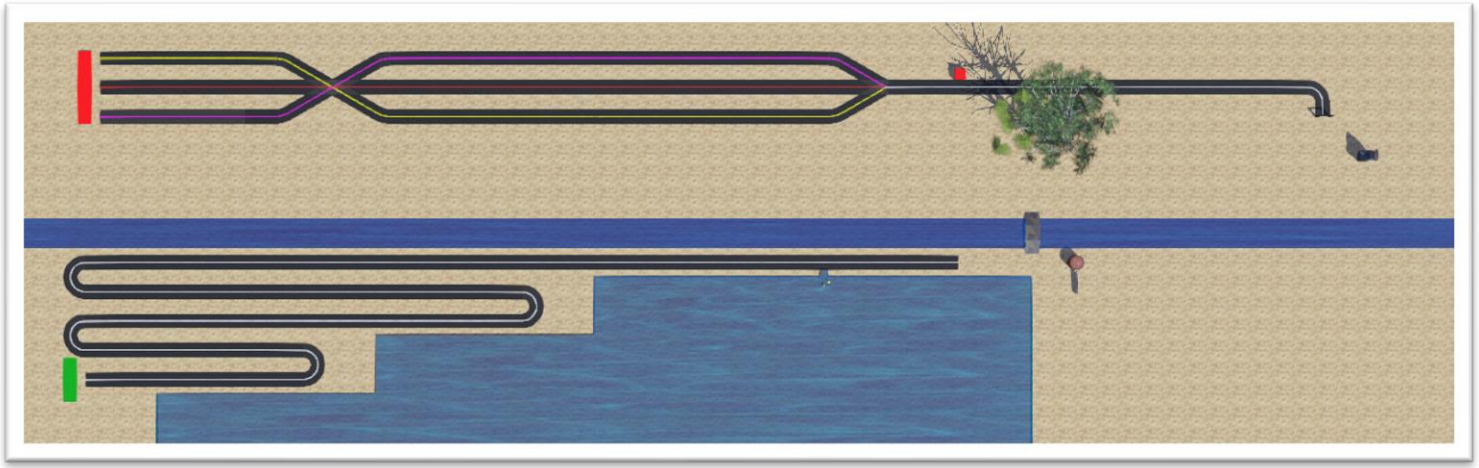


Figure1-Top View of Our Patio in Webots

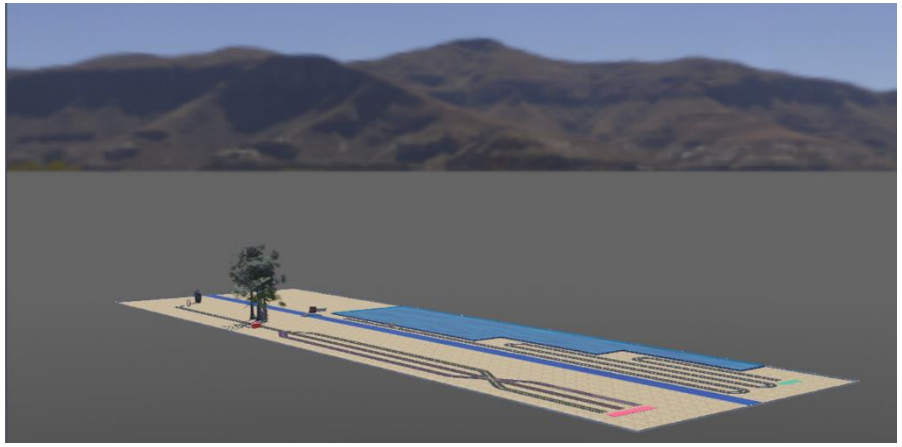
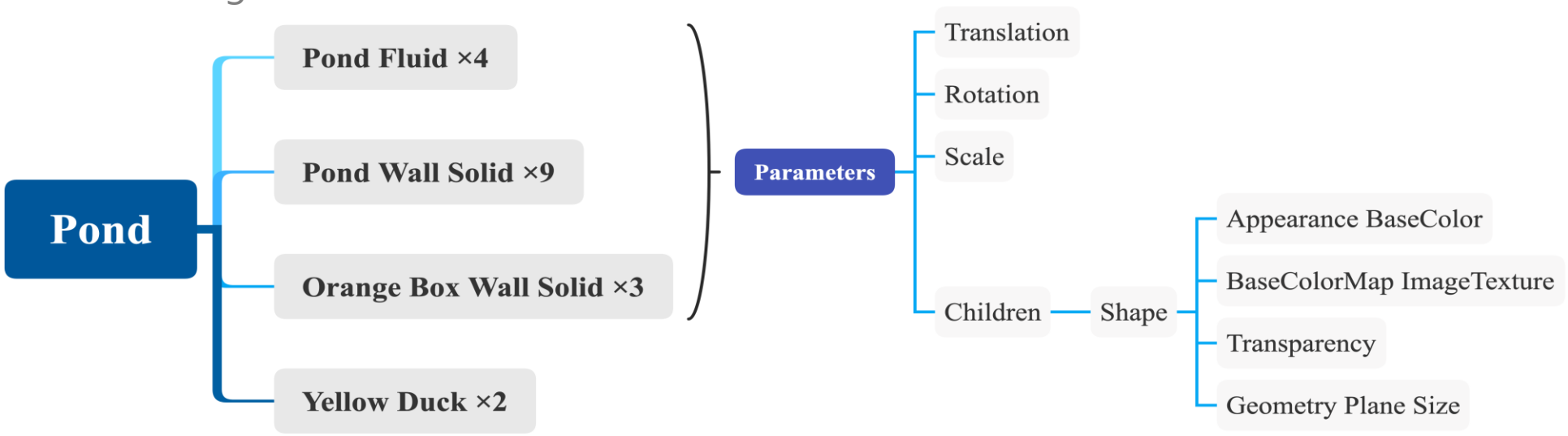


Figure2-Side View of Our Patio in Webots

03 Environment Building

Building Pond



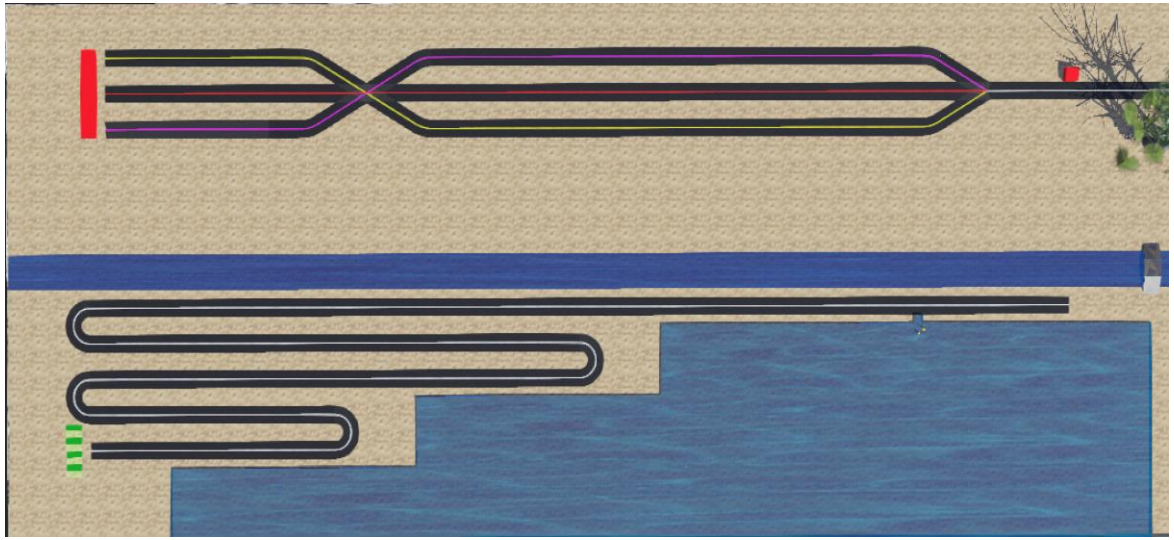


Figure 3 - Top View of Our Designed Pond in Webots

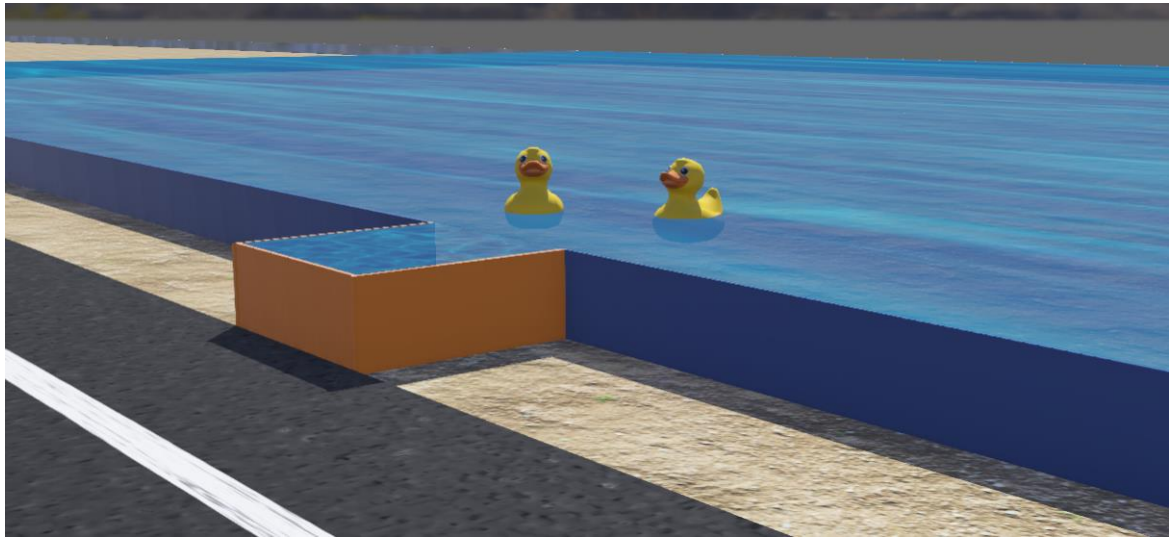


Figure 4 - Partial View of Our Designed Pond and Orange Box in Webots

03 Camera Recognition

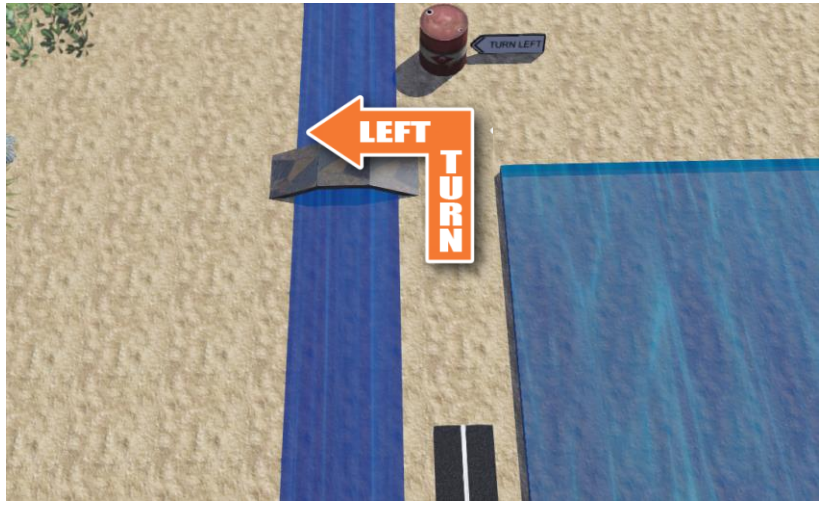


Figure5-An Example of Bridge Recognition and Left Turn

Figure6-An Example of Tree Recognition and Right Turn

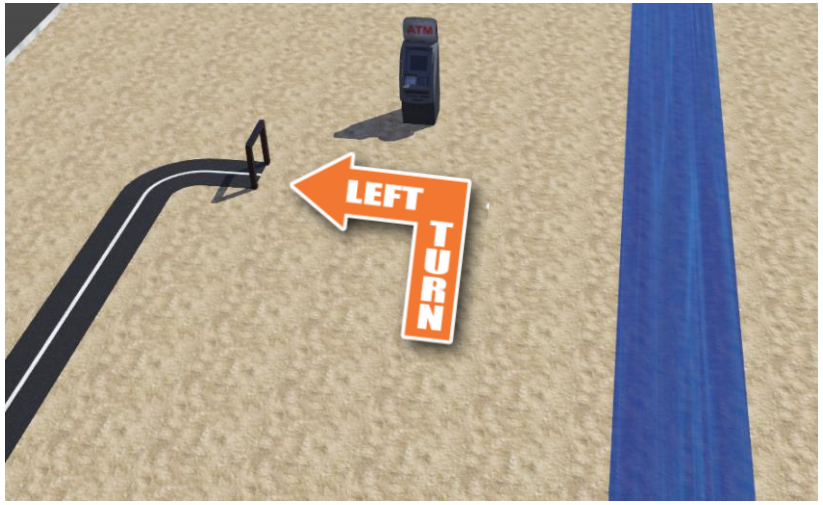
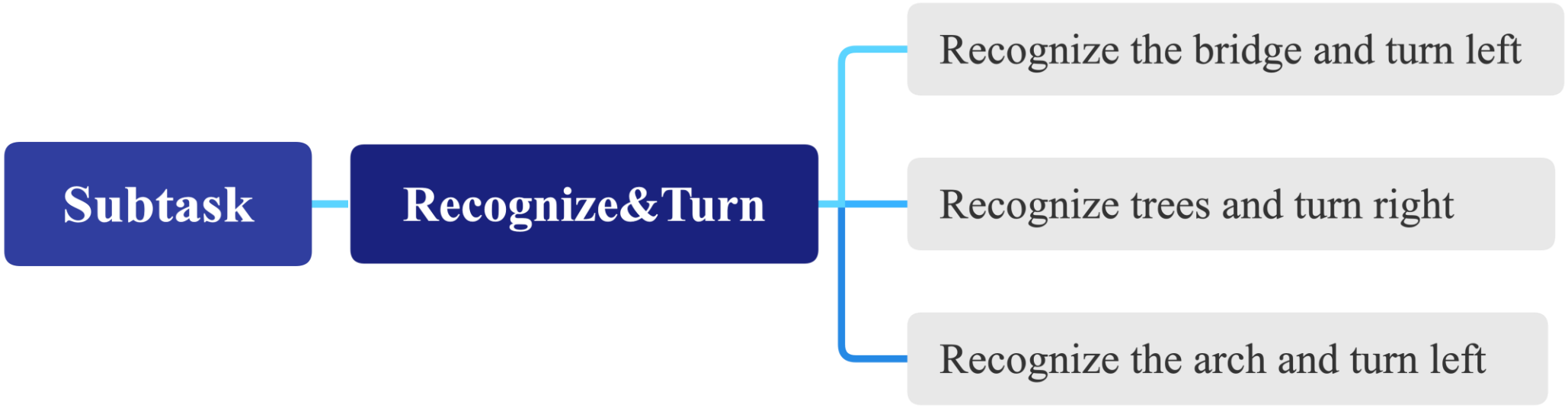
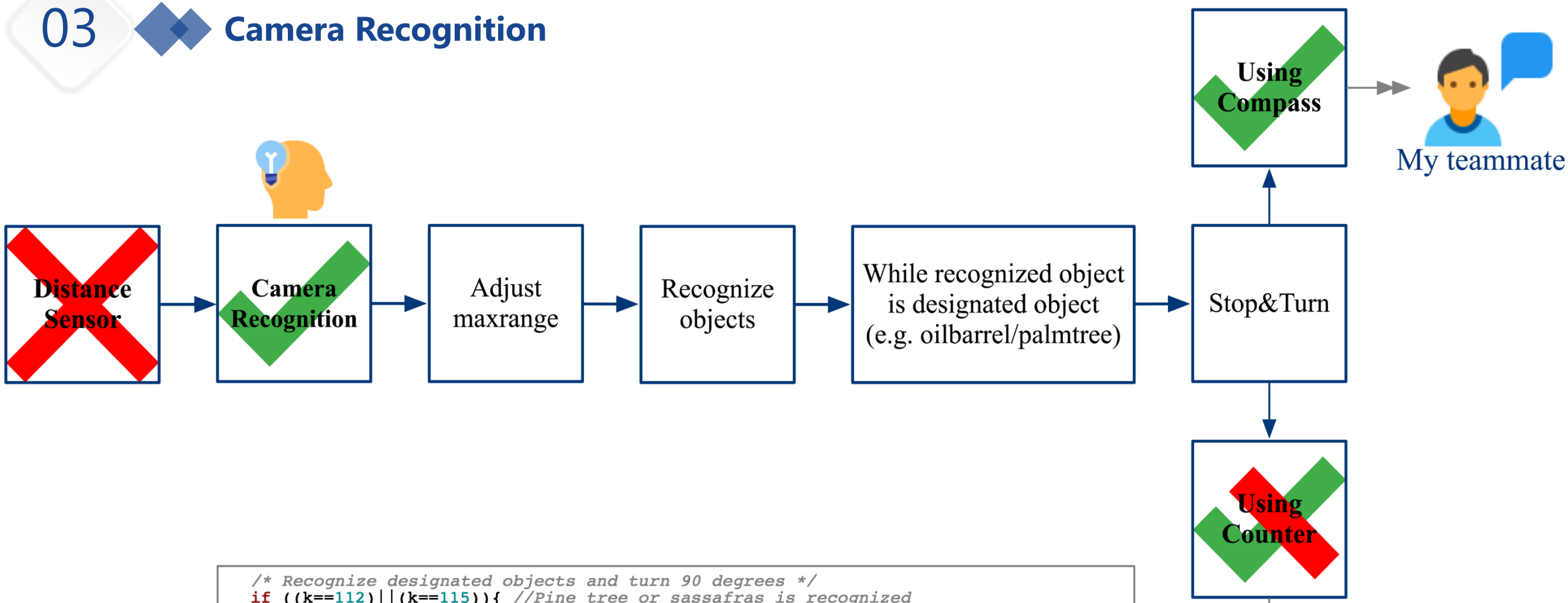


Figure7-An Example of Arch Recognition and Left Turn



03 Camera Recognition




```
/* Recognize designated objects and turn 90 degrees */
if ((k==112)||(k==115)){ //Pine tree or sassafras is recognized
while (wb_robot_step(TIME_STEP) != -1) {
q=q+1; //Set a counter
if (q<=3){
  wb_motor_set_velocity(left_motor, 10);
  wb_motor_set_velocity(right_motor, -10); //Set the motors speed to make a turn
  }
else {
  wb_motor_set_velocity(left_motor, 10);
  wb_motor_set_velocity(right_motor, 10); //Go straight after turning
  }
break;
}
```

Obstacle Avoidance

Yihan Yang

04

Obstacle Avoidance

Rover Obstacle Avoidance

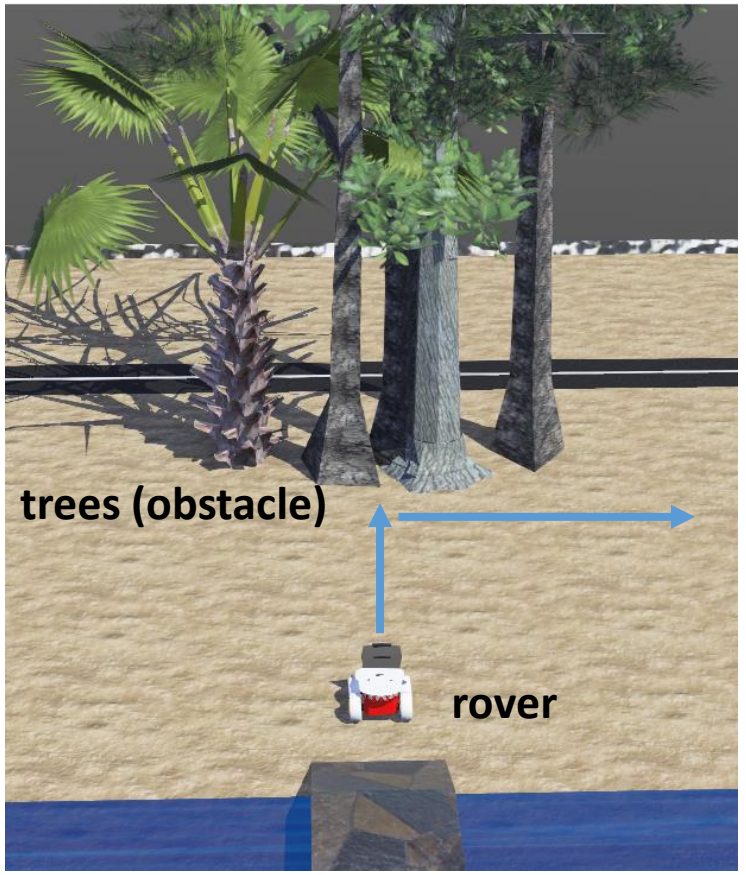
After finding the bridge and crossing it, the rover needs to detect the tree and then bypass the trees to avoid collision. To be more specific, when the rover moves down the bridge, the sensor will detect the trees. When the rover is close enough to the trees, it will turn 90 degrees to the right and move on to the next task area.

Environment

Pioneer 3-AT, Sonar distance sensor, Camera sensor, Compass.

Parameters

Symbol	Description
$v_{fl}, v_{fr}, v_{bl}, v_{br}$	Speed of four wheels
v_{max}	Maximum speed of wheels
v_{cr}	Cruising speed of wheels
S_i	Return value of sonar distance sensors
D_{th}	Distance threshold
i_k	Item number in simulation world
θ	Rotational angle of rover
R_{max}	Maximum range of camera detection



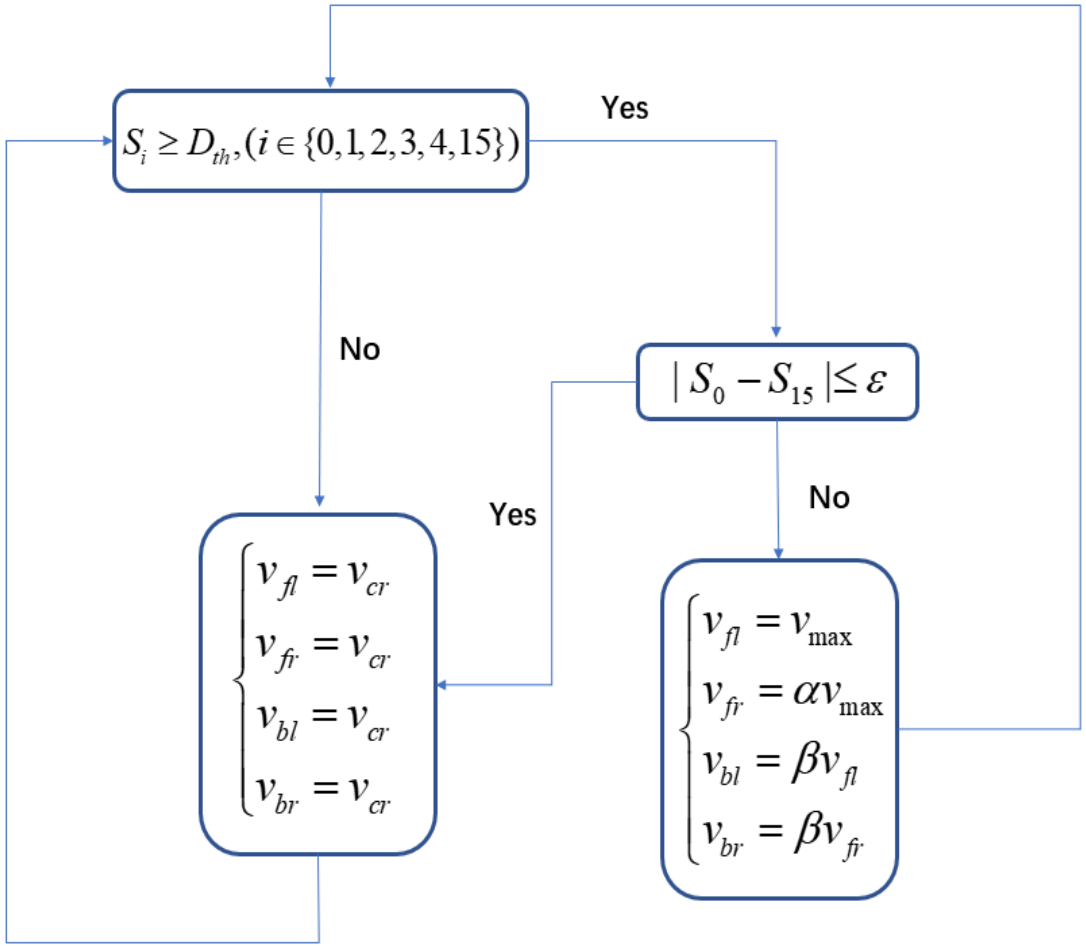
Simulation Environment

04

Obstacle Avoidance

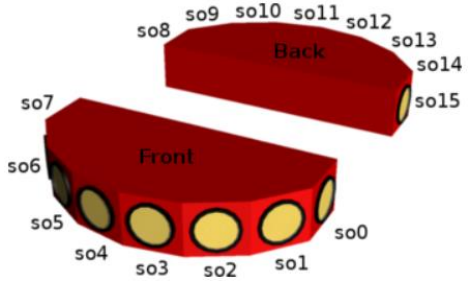
Method 1: Based on the sonar distance sensor

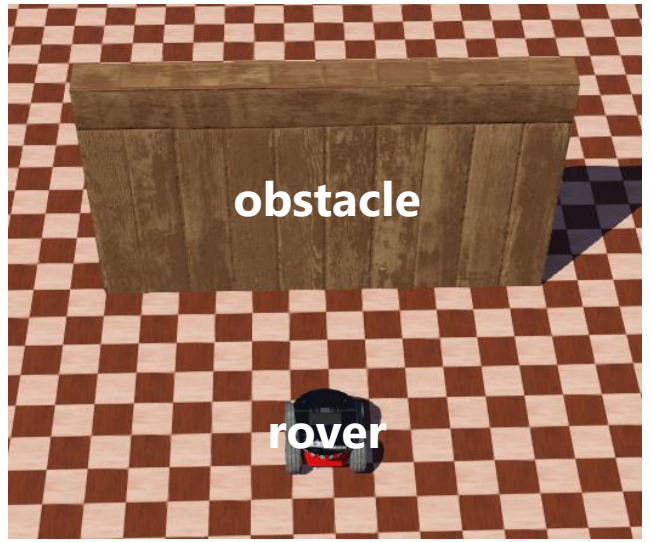
Algorithm Structure



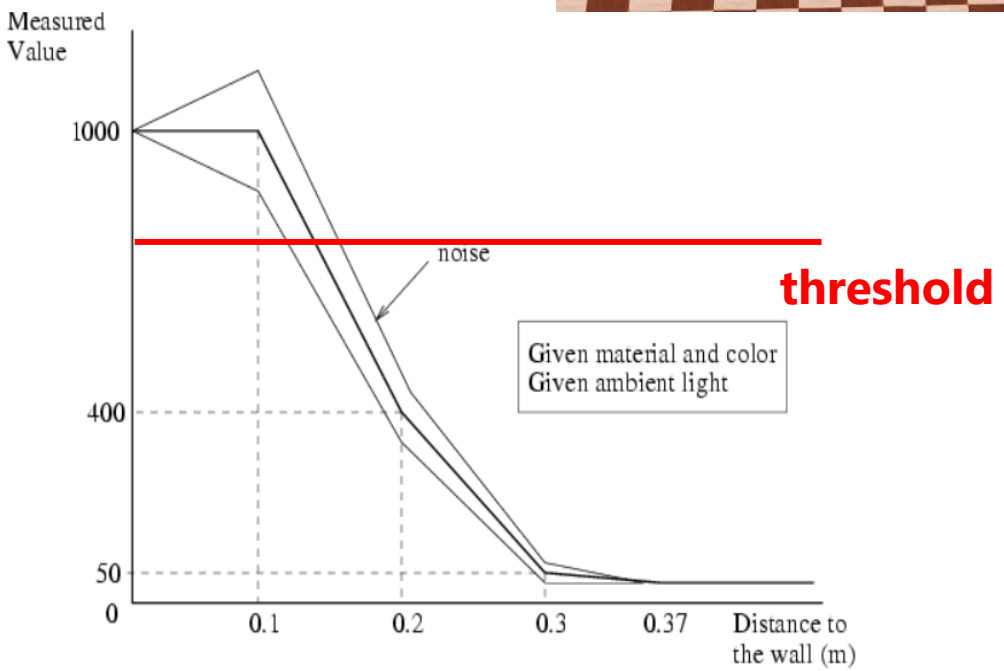
$$\alpha = 0.5, \beta = 0.9$$

Sensor Distribution





Lookup Table

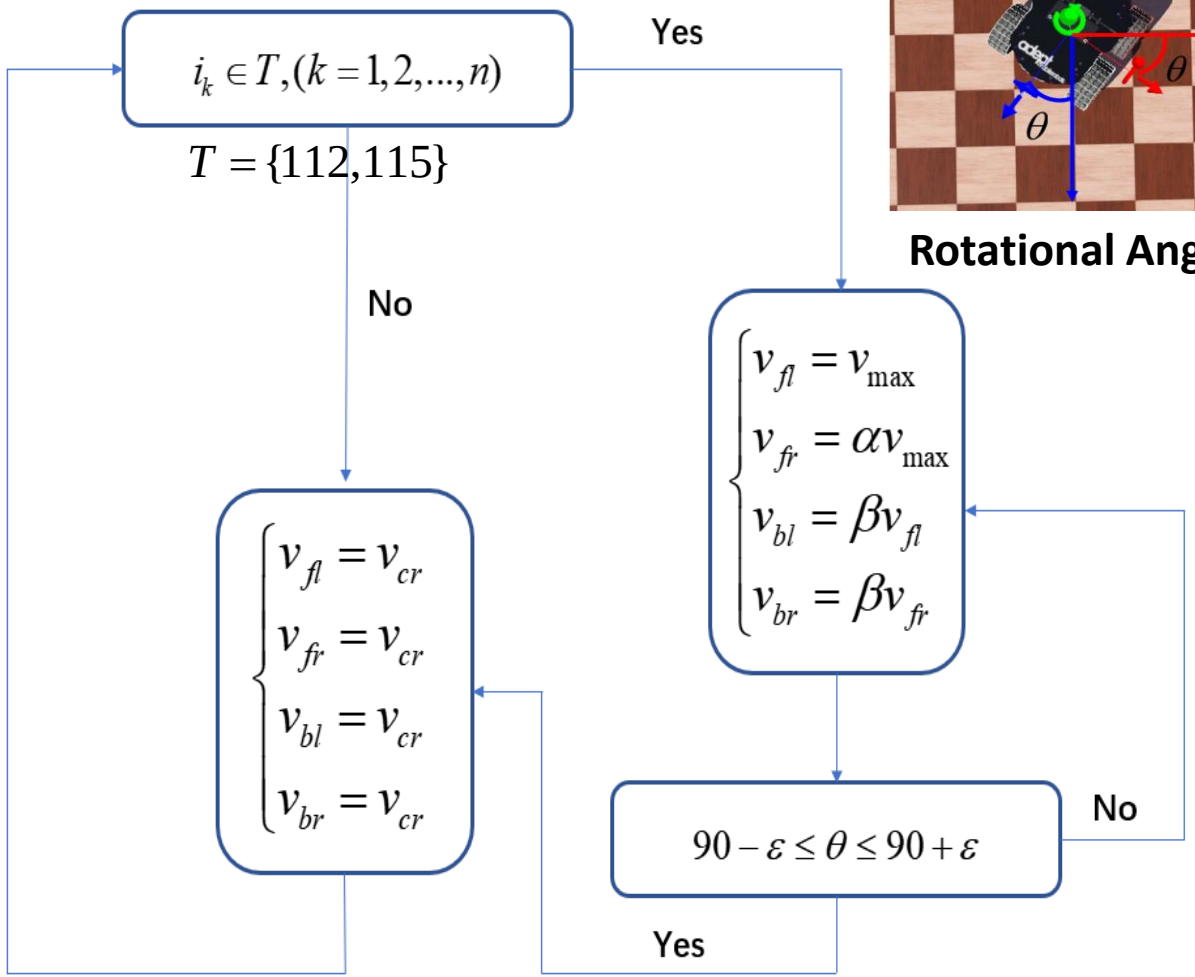


04

Obstacle Avoidance

Method 2: Based on camera recognition

Algorithm Structure



Rotational Angle

Item Number

```
Model of object 1: road line
k=114

Model of object 0: oil barrel
k=111

Model of object 0: ATM
k=65

Model of object 0: pine tree
k=112

Model of object 3: sassafras tree
k=115
```

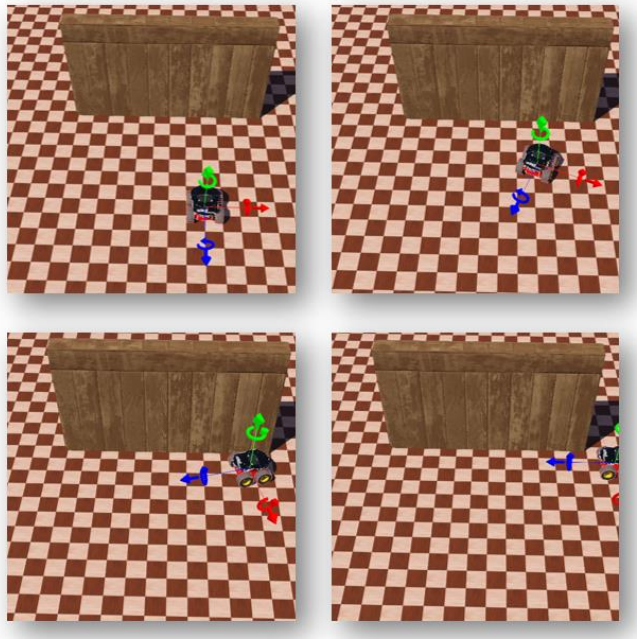
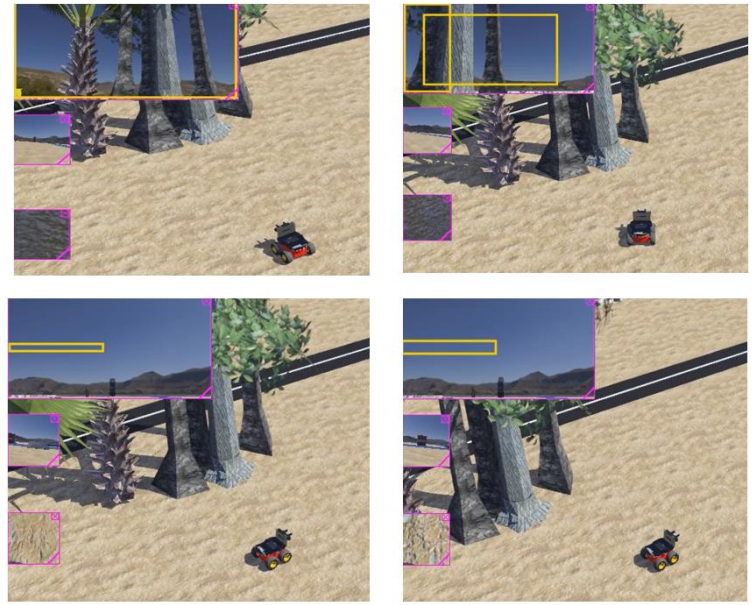
Controller Program

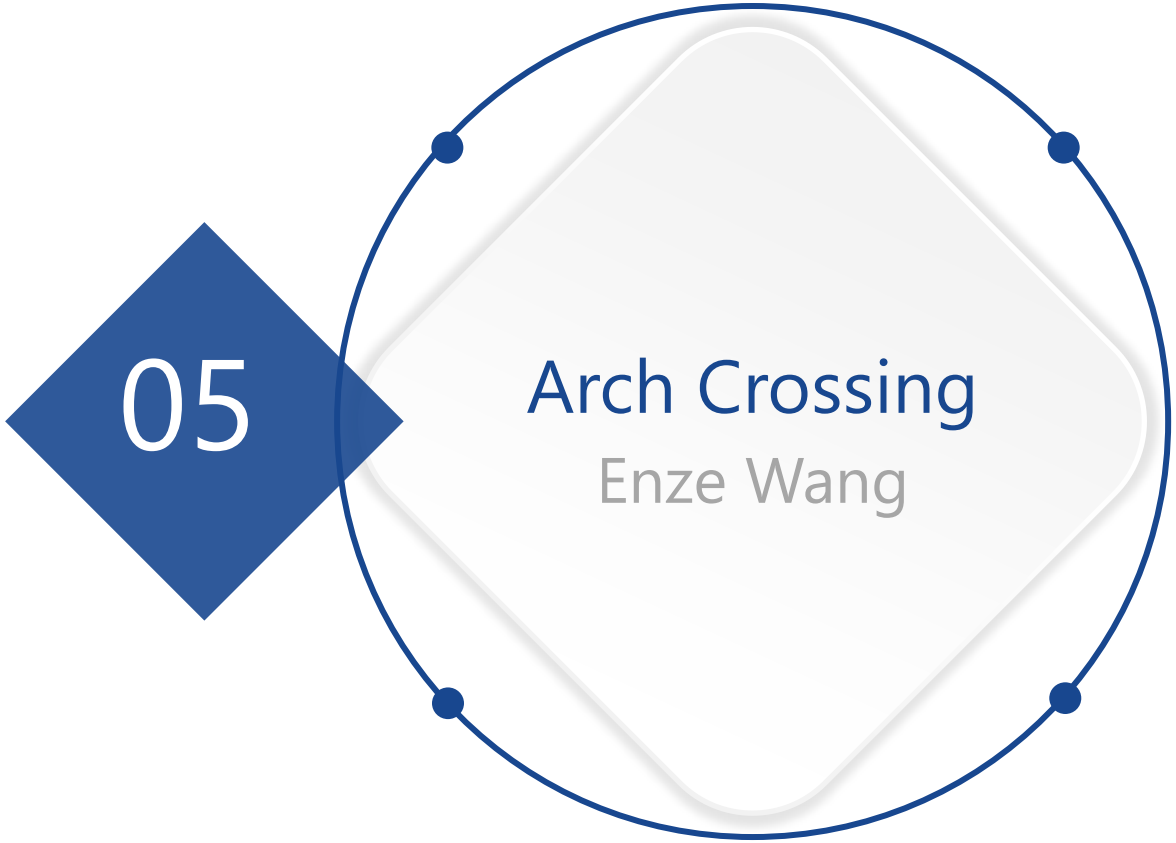
```
const WbCameraRecognitionObject* objects
= wb_camera_recognition_get_objects(camera);
for (i = 0; i < number_of_objects; ++i) {
 printf("Model of object %d: %s\n", i, objects[i].model);
 k = *(objects[i].model);
 if ((k == 112) || (k == 115)) {//tree detected
     find_tree = 1;
 }
}

if (find_tree) {
    avoid_tree = 1;
}

// turn right to avoid the tree
if ((West == 0) && avoid_tree) {
    left_speed = 2, right_speed = -2;
    left_speed = 2, right_speed = -2;

    wb_motor_set_velocity(front_left_wheel, left_speed);
    wb_motor_set_velocity(front_right_wheel, right_speed);
    wb_motor_set_velocity(back_left_wheel, left_speed);
    wb_motor_set_velocity(back_right_wheel, right_speed);
}
```

04

Obstacle Avoidance

Comparison of Method 1 and Method 2

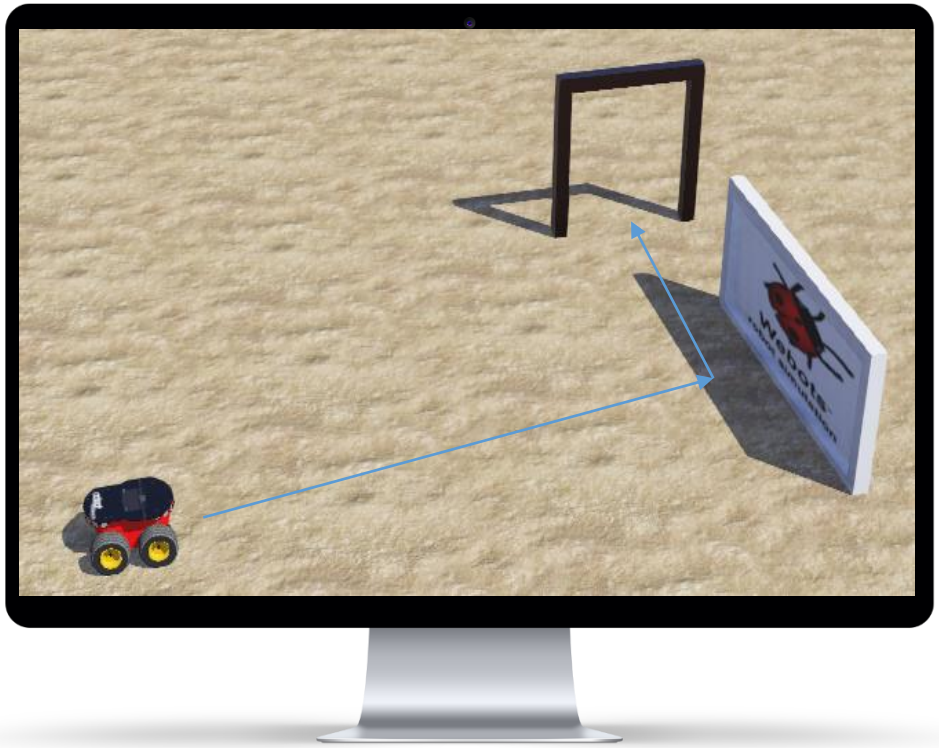
	Method 1 (Sonar Distance Sensor)	Method 2 (Camera Recognition)
Strength	• Smaller memory footprint • Shorter time delay	• High level of accuracy • Sensitive to the objects in the simulation world
Weakness	• Restricted to obstacles with smooth surface • Susceptible to the environment noise	• Larger memory footprint • Relatively limited application scope
Simulation Result		



Arch Crossing

Enze Wang

05 Arch Crossing



The arch

After the car avoids the trees, the arch is in front of it to the left.

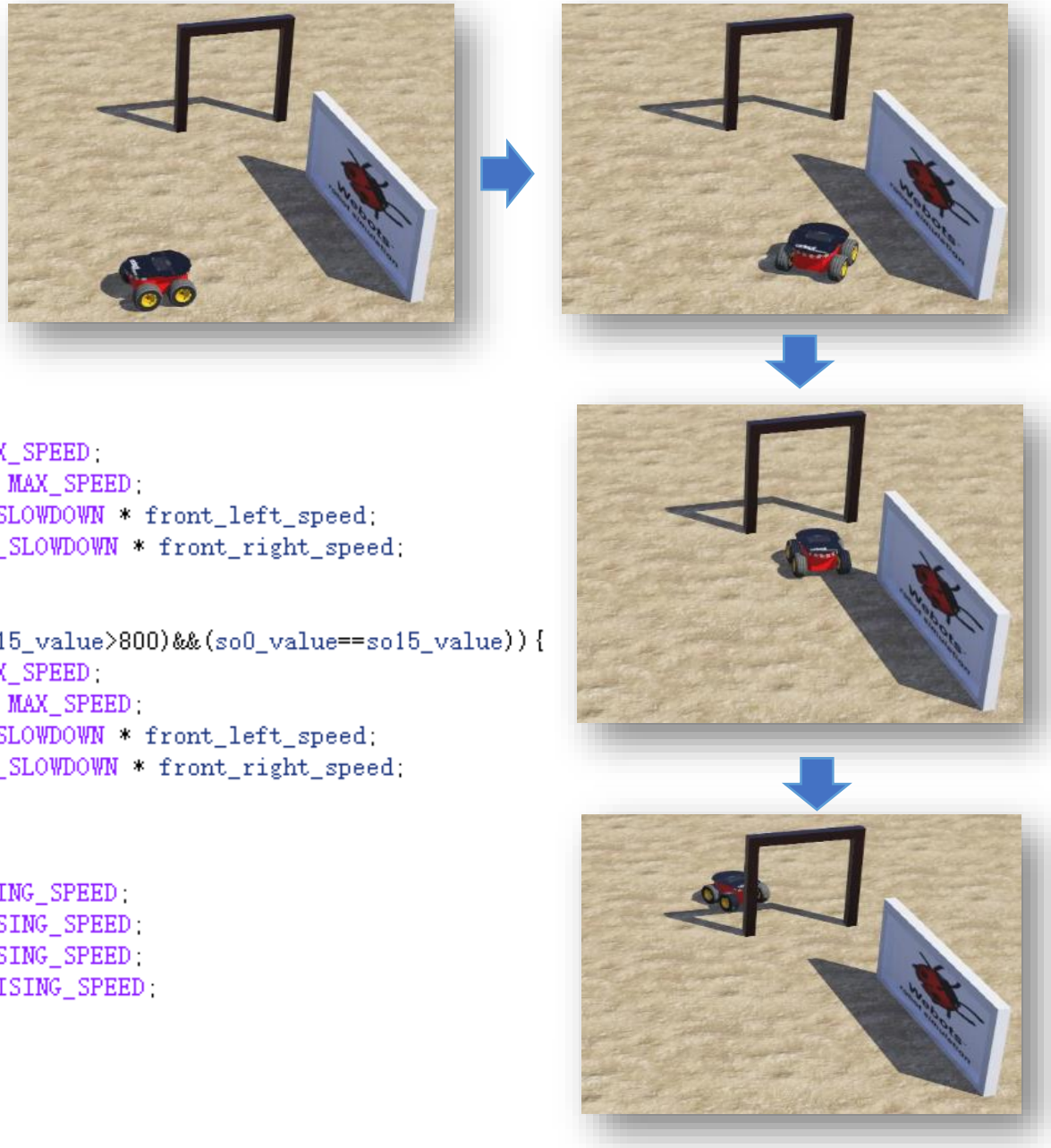
The sign

The sign is placed in a specific position which can help the rover find the arch and cross it.

Distance sensors

Distance sensors are used to detect the location of the sign.

05 Arch Crossing



Problem 1

It is hard to accurately control the car's turning angle because different sensors detect the sign at different times.

Problem 2

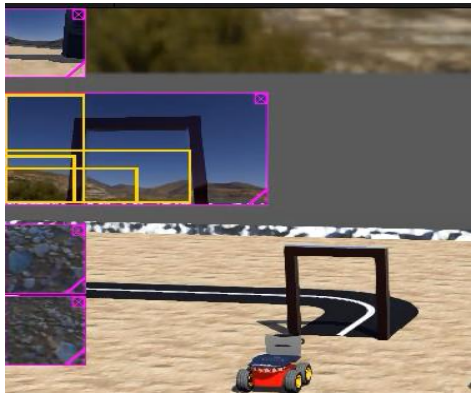
This code is similar with obstacle avoidance, which will cause the car to regard the columns on either side of the arch as obstacles and move away from it instead of crossing the arch.

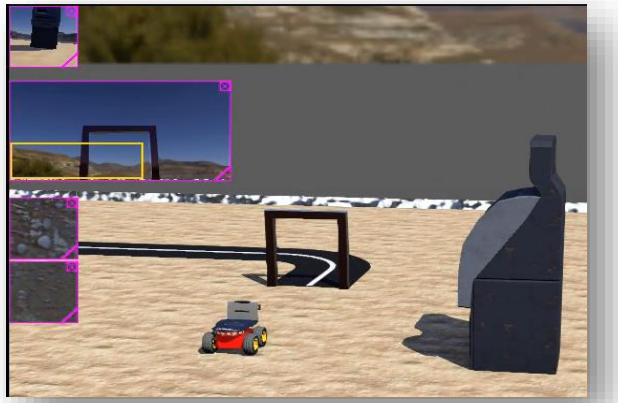
```
if (sensor_value > 800){
front_right_speed =  MAX_SPEED;
front_left_speed = 0.2* MAX_SPEED;
back_left_speed = BACK_SLOWDOWN * front_left_speed;
back_right_speed = BACK_SLOWDOWN * front_right_speed;
counter=1;
}
if((so0_value>800)&&(so15_value>800)&&(so0_value==so15_value)){
front_right_speed =  MAX_SPEED;
front_left_speed = 0.2* MAX_SPEED;
back_left_speed = BACK_SLOWDOWN * front_left_speed;
back_right_speed = BACK_SLOWDOWN * front_right_speed;
counter=0;
}
if (counter==0){
back_left_speed = CRUISING_SPEED;
back_right_speed = CRUISING_SPEED;
front_left_speed = CRUISING_SPEED;
front_right_speed = CRUISING_SPEED;
}
```

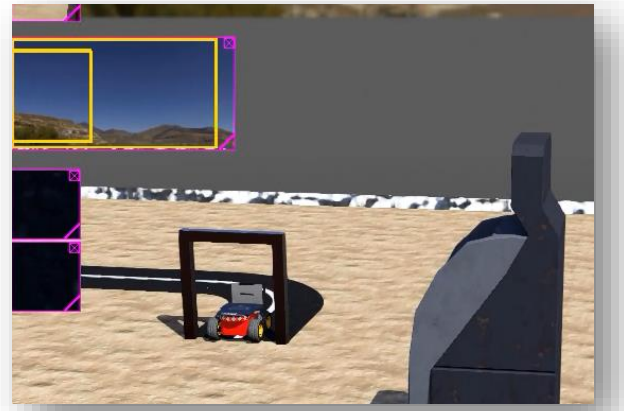
05 Arch Crossing

Camera

Camera is used to identify the sign on the ground.

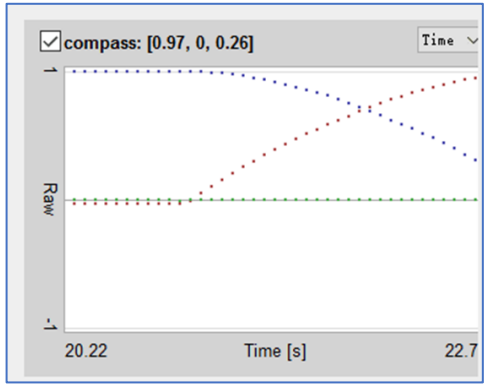


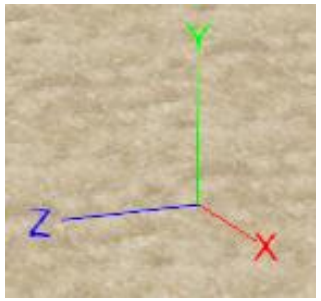




Compass

Compass is used to control the rover to turn 90 degrees.





East: -1,0,0 —— +Z-axis
South: 0,0,1 —— -X-axis
West: 1,0,0 —— -Z-axis
North: 0,0,-1 —— +X-axis

05 Arch Crossing

The whole process:

Camera identify the sign → Execute the left turn command → When the car facing south → Execute going straight command




	Counter	Distance Sensor	Compass
Strength	• Simple operation and code, turn faster.	• Sensitive to the objects in the simulation world	• Control the turning angle more accurately
Weakness	• Greatly influenced by the complex environment.	• Hard to accurately control the car's turning angle.	• More complex and turn slower.

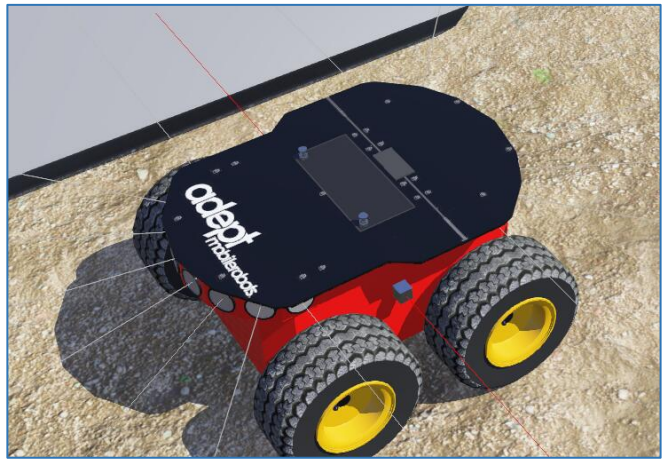

06
Bridge Crossing
Yuchen Zhang

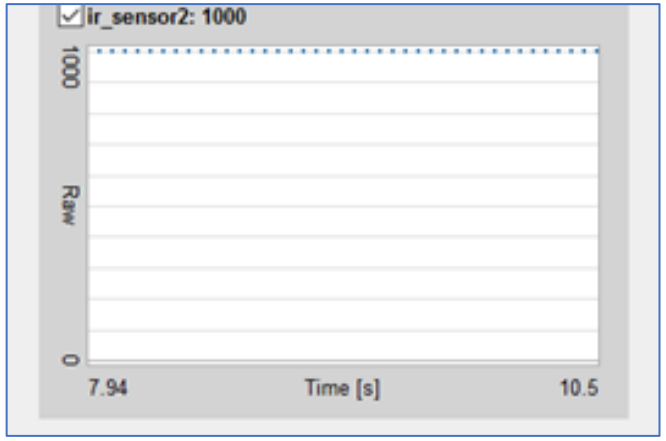
06

Bridge Crossing

Choosing Equipment

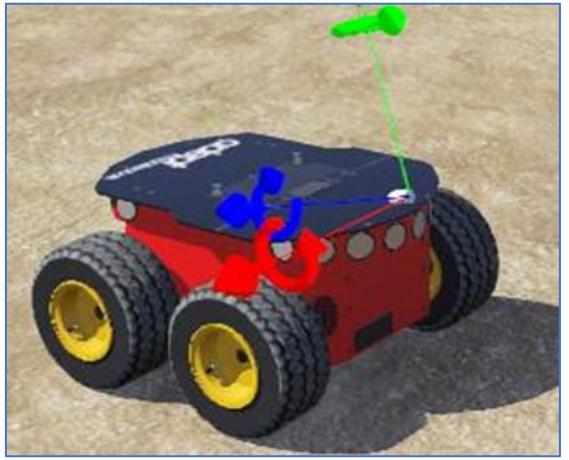
Method 1
Distance sensor (IR and sonar)





Can’t detect the bridge

Method 2
Camera



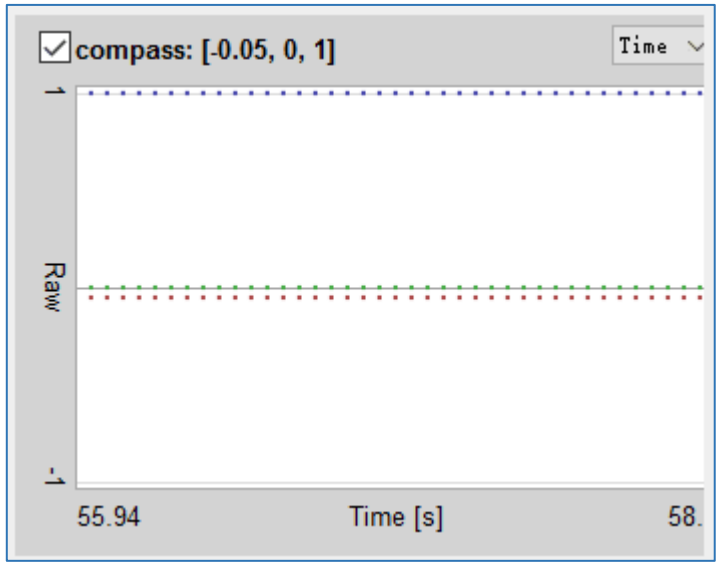
To have a better view

06

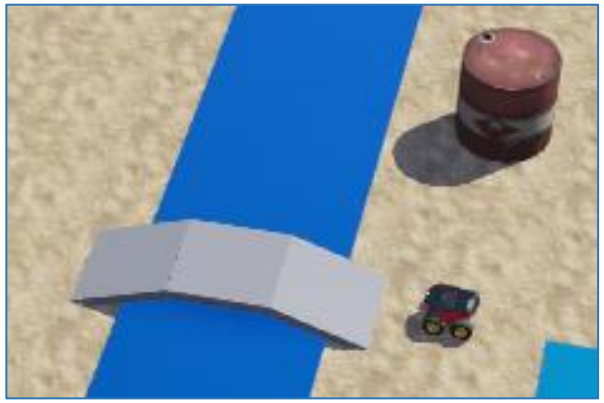
Bridge Crossing

Make Turn

PROBLEM:
Couldn’t aim at direction accurately



South: 0,0,1 North: 0,0,-1
East: -1,0,0 West: 1,0,0



Original code:

```
bool Not_South=(north[0]!=0)||(north[2]!=1);
bool Not_Nouth=(north[0]!=0)||(north[2]!=-1);
bool Not_East=(north[0]!=-1)||(north[2]!=0);
bool Not_West=(north[0]!=1)||(north[2]!=0);
```

To extend the scope:

```
bool Not_South=(abs(north[0])>=0.1)||(abs(north[2]-1)>=0.1);
bool Not_Nouth=(abs(north[0])>=0.1)||(abs(north[2]+1)>=0.1);
bool Not_East=(abs(north[0]+1)>=0.1)||(abs(north[2])>=0.1);
bool Not_West=(abs(north[0]-1)>=0.1)||(abs(north[2])>=0.1);
```

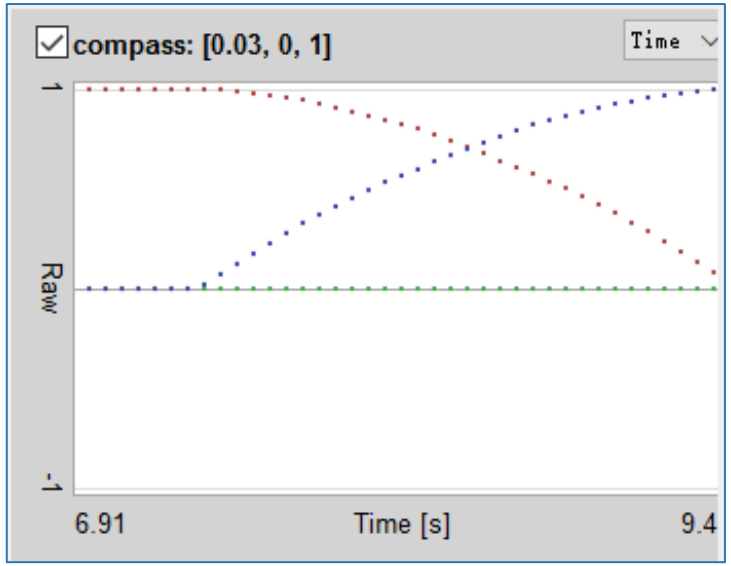
However, it couldn’t meet the requirement.

06

Bridge Crossing

Two Solutions of the Problem

1. Slow down the turning speed when the robot is about to finish rotation (without extending scopes)



Still couldn't meet the requirement

2. Define new directions after processing the value of compass and then extend its scopes

```
const double *north = wb_compass_get_values(compass);
double angle = atan2(north[0], north[2]);
double bearing=(angle-1.5708)/M_PI*180.0;
if(bearing<0.0){
  bearing+=360.0;
  }
  printf("bearing is %f\n",bearing);


bool South=(bearing>=266)&&(bearing<=269.6);
bool North=(bearing>=86)&&(bearing<=89.6);
bool East=(bearing>=176)&&(bearing<=179.6);
bool West=((bearing>=356)&&(bearing<=360))||((bearing>=0)&&(bearing<=1.6));
```

Successfully completed

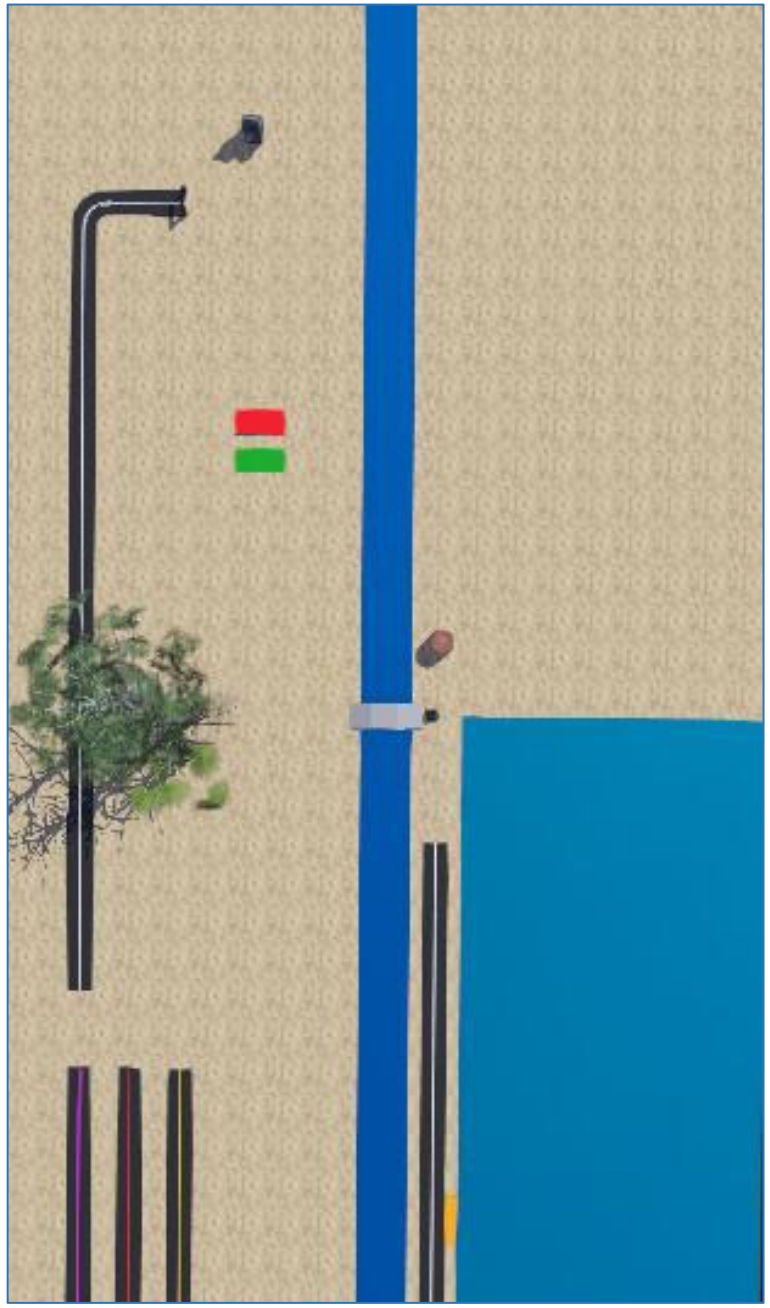
06 Bridge Crossing

Integration

Keep moving straight

```
if ((avoid_tree==0)&&(cross_bridge==0)&&(cross_arch==0)&&(food==0))&&(finish==0))){
  back_left_speed = 6.4, back_right_speed = 6.4;
  front_left_speed = 6.4, front_right_speed = 6.4;

  wb_motor_set_velocity(front_left_wheel, front_left_speed);
  wb_motor_set_velocity(front_right_wheel, front_right_speed);
  wb_motor_set_velocity(back_left_wheel, back_left_speed);
  wb_motor_set_velocity(back_right_wheel, back_right_speed);
  }
```

S

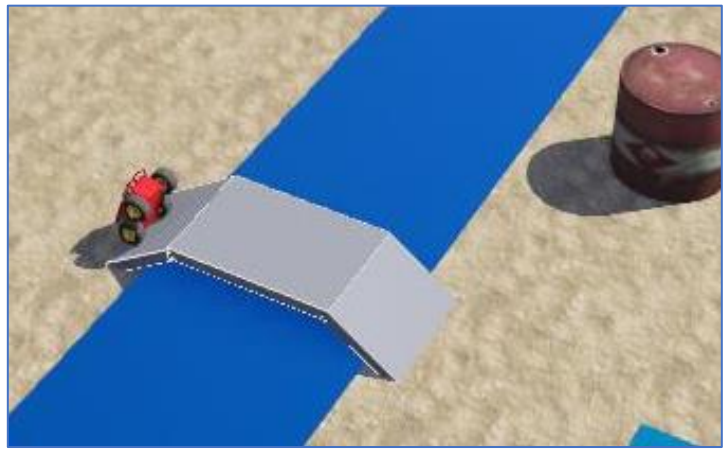


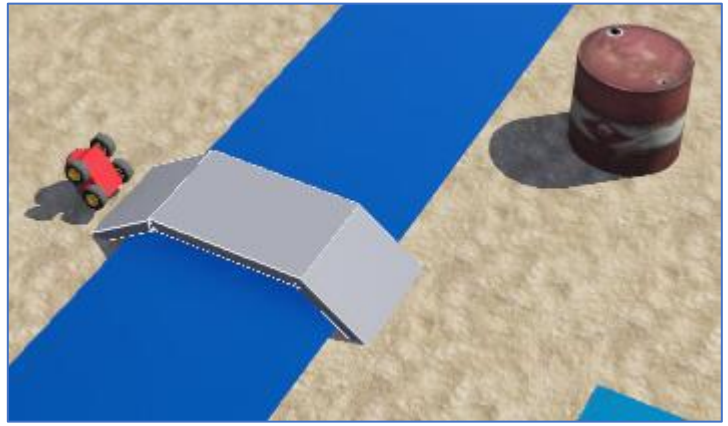
N

Make turn

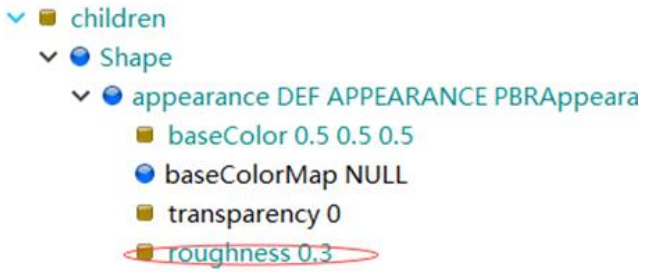
```
if ((South==0) && cross_bridge){
   back_left_speed = -2, back_right_speed = 2;
   front_left_speed = -2, front_right_speed = 2;

   wb_motor_set_velocity(front_left_wheel, front_left_speed);
   wb_motor_set_velocity(front_right_wheel, front_right_speed);
   wb_motor_set_velocity(back_left_wheel, back_left_speed);
   wb_motor_set_velocity(back_right_wheel, back_right_speed);}
```

06 Bridge Crossing





Change roughness of the bridge



Change the speed

Our robot would speed down when it is going down the bridge.

```
if ((b>=92)&&(b<=120)){
  back_left_speed = 3, back_right_speed = 3;
  front_left_speed = 3, front_right_speed = 3;

  wb_motor_set_velocity(front_left_wheel, front_left_speed);
  wb_motor_set_velocity(front_right_wheel, front_right_speed);
  wb_motor_set_velocity(back_left_wheel, back_left_speed);
  wb_motor_set_velocity(back_right_wheel, back_right_speed);}
```

Path Following

Jingyi Wang

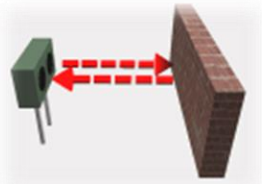
07

Path Following

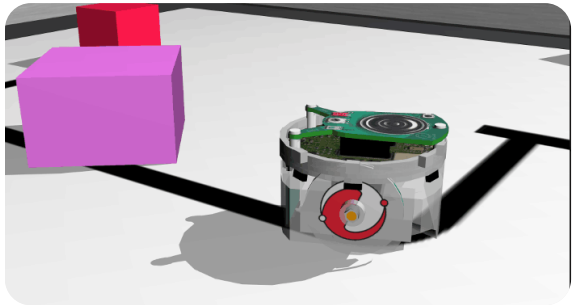
Contribution (Mainly in Task1, 2, 5)

Final Design for Line Follwing

Both in Task1 & Task5



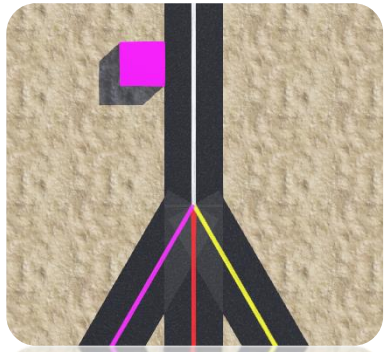
Case Study for Task1



Cooperation with Teammates

Help for color recognition in Task 2 & 5

Help for environment set in Task5



Debug & Improvement

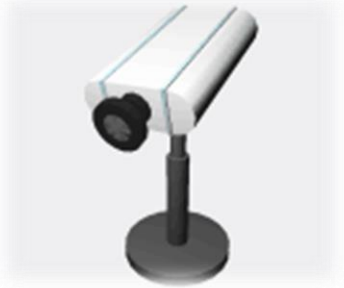
Change the speed expression

Error with fish food's weight

07

Path Following

Final Design for Line Following in Task 1&5



Sensor: Camera

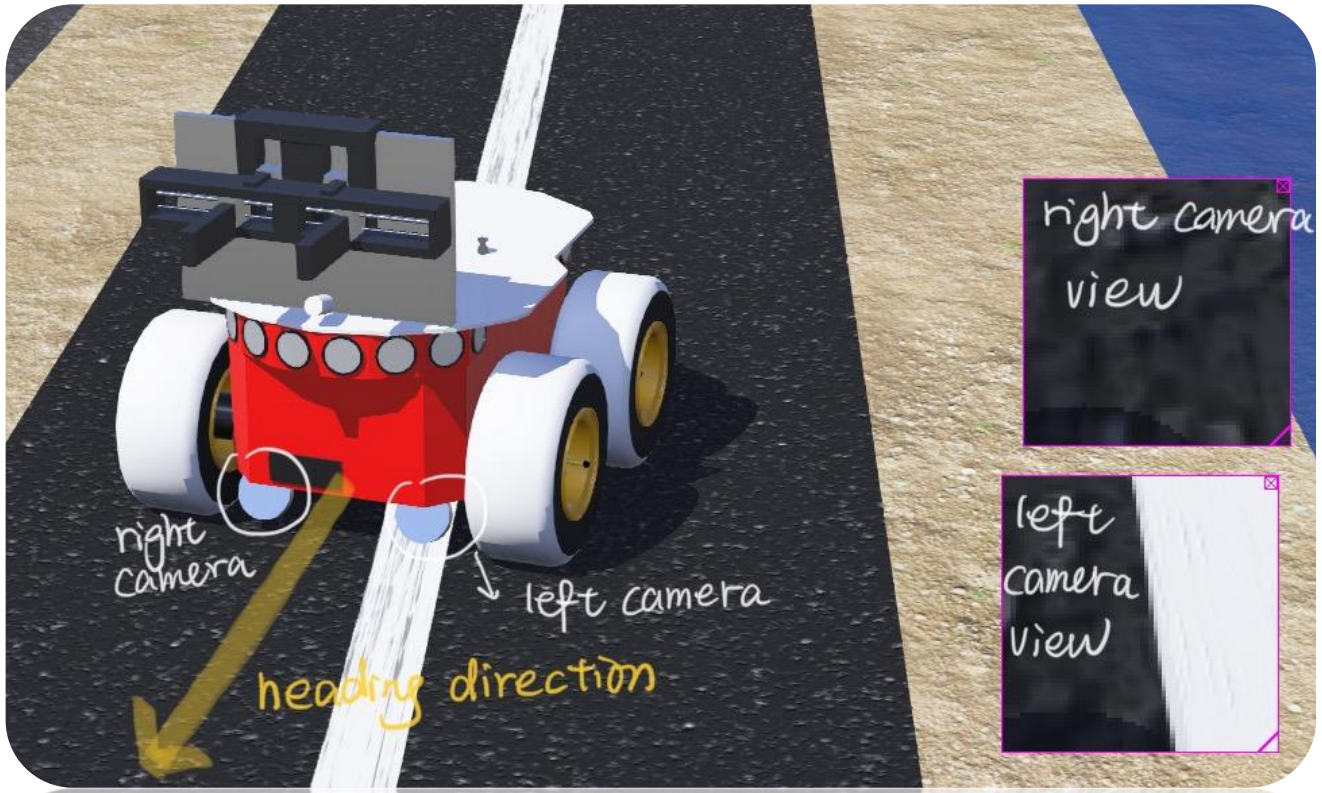
The Camera node can model a typical RGB camera

Basic Idea: Same as e-puck demo

Use the difference of two camera input to set the speed for each wheel

Core Function: Get_RGB

wb_camera_get_image
wb_camera_image_get_red
wb_camera_image_get_green
wb_camera_image_get_blue



07 Path Following

Main Code (Task 1&5)

Initialization & Enable

```
48 /* Get the camera device, enable it, and store its width and height */
49 WbDeviceTag camera_right = wb_robot_get_device("camera_right");
50 wb_camera_enable(camera_right, TIME_STEP);
```

```
297 ////////////////////////////////////////////////////the funtion of tracking line
298 int tracking(){
299   const unsigned char* image_r = wb_camera_get_image(camera_right);
300   const unsigned char* image_l = wb_camera_get_image(camera_left);
301   const unsigned char* image_m = wb_camera_get_image(camera_side);
302     /* Reset the sums */
303     red_l = 0;
304     green_l = 0;
305     blue_l = 0;
306
307     red_r = 0;
308     green_r = 0;
309     blue_r = 0;
310
311     red_s = 0;
312     green_s = 0;
313     blue_s = 0;
314
315     /*get the RGB value of the two camera*/
316
317     for (int x = 0; x < width; x++)
318       for (int y = 0; y < height; y++) {
319           red_r = wb_camera_image_get_red(image_r, width, x, y);
320           green_r = wb_camera_image_get_green(image_r, width, x, y);
321           blue_r = wb_camera_image_get_blue(image_r, width, x, y);
322
323           red_l = wb_camera_image_get_red(image_l, width, x, y);
324           green_l = wb_camera_image_get_green(image_l, width, x, y);
325           blue_l = wb_camera_image_get_blue(image_l, width, x, y);
326         }
```

Get Image & RGB Value

- *red_r* ***green***_r *blue_r*:

  RGB value derived from right camera

- *red_l green_l blue_l*:

  RGB value derived from left camera

- *width height*:

  The image width and height get via function

  "wb_camera_get_width"

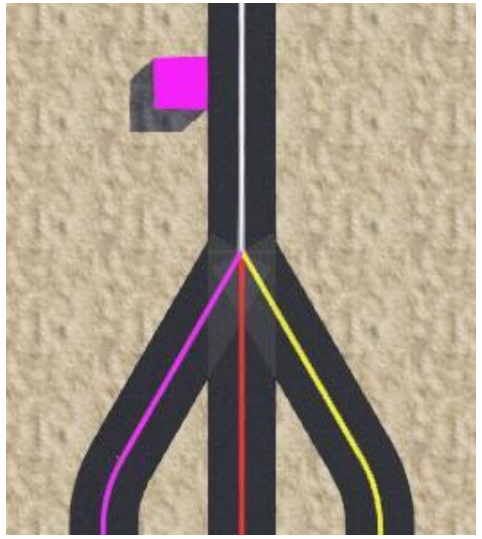
  "wb_camera_get_height "

07

Path Following

Main Code Cont. (Task 1&5)

Speed Modification



red: 1
green: 0
blue: 1

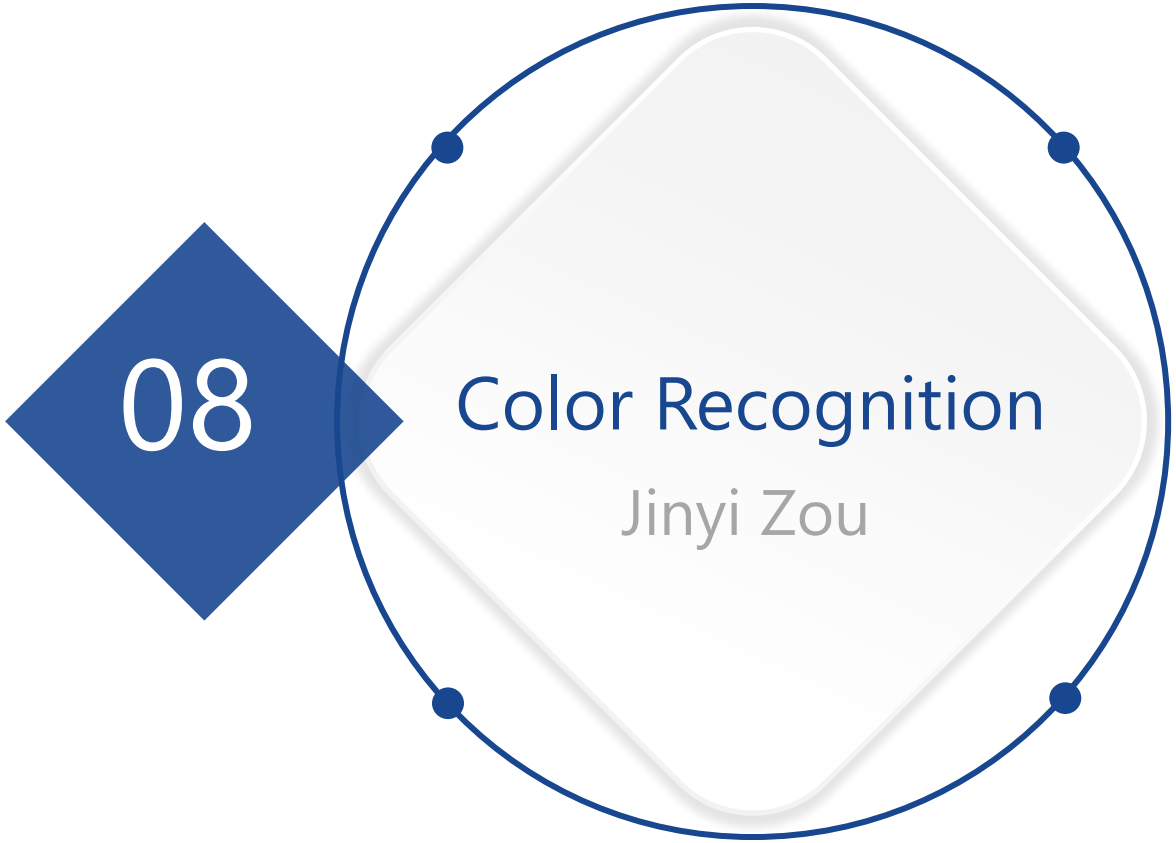
red: 0
green: 0
blue: 0

red: 1
green: 1
blue: 1

red: 1
green: 0
blue: 0

red: 1
green: 1
blue: 0

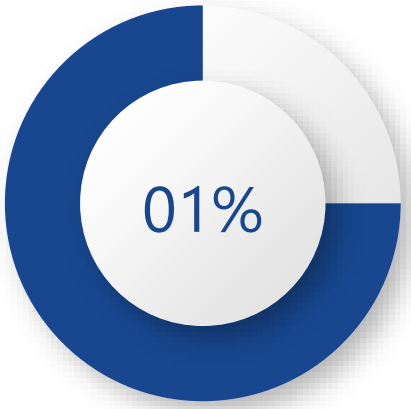
```
switch (key) {
  case'0'://red &white
    if (red_r - red_l > 30) {
      left_speed = 6.4, right_speed = 6.4-0.02*(red_r-red_l);
    }
    else if (red_l - red_r > 30) {
      left_speed = 6.4-0.02*(red_l-red_r), right_speed = 6.4;
    }
    else {
      left_speed = right_speed = 6.4;
    }
    break;

  case '1'://purple
    if (blue_r - blue_l > 30) {
      left_speed = 6.4, right_speed = 6.4-0.02*(blue_r-blue_l);
    }
    else if (blue_l - blue_r > 30) {
      left_speed = 6.4-0.02*(blue_l-blue_r), right_speed = 6.4;
    }
    else {
      left_speed = right_speed = 6.4;
    }
    break;

  case '2'://yellow
    if (green_r - green_l > 30) {
      left_speed = 6.4, right_speed = 6.4-0.02*(green_r-green_l);
    }
    else if (green_l - green_r > 30) {
      left_speed = 6.4-0.02*(green_l-green_r), right_speed = 6.4;
    }
    else {
      left_speed = right_speed = 6.4;
    }
    break;
```



Color Recognition

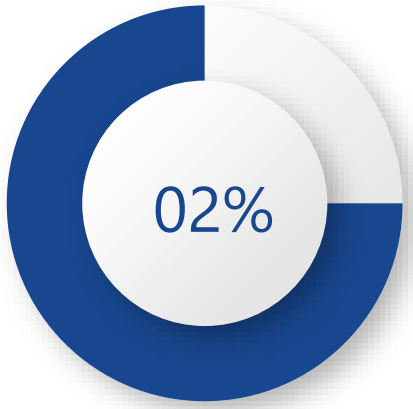
Jinyi Zou

08 Color Recognition



Self-study

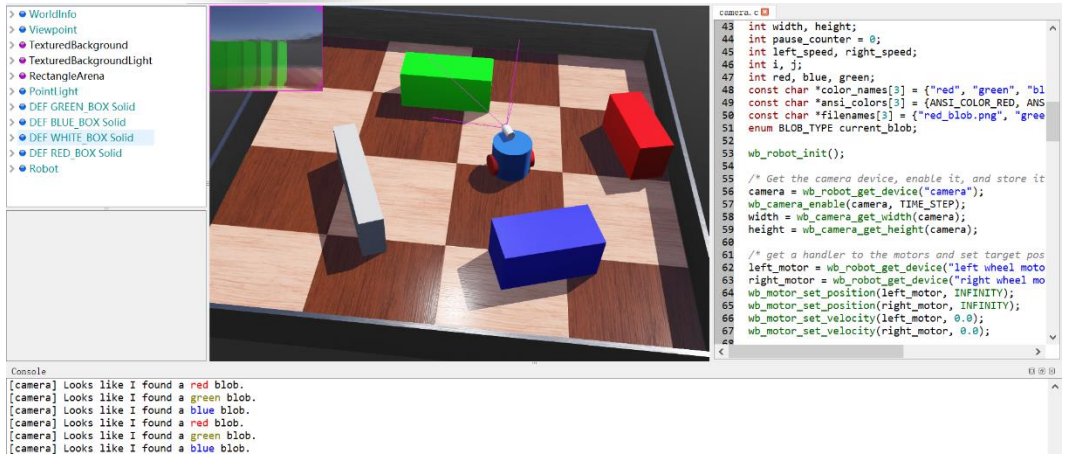
Organize resources
e.g. Webots documents, Github



Cooperation

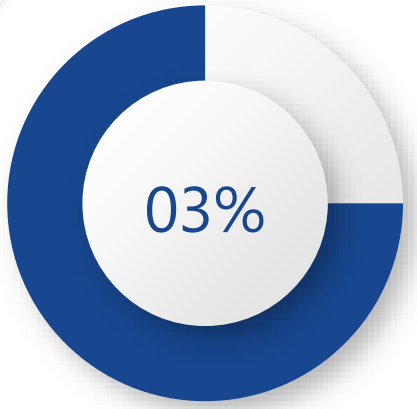
Face a problem:
Cannot read the exact RGB value

```
while (wb_robot_step(TIME_STEP) != -1) {
  const unsigned char *image = wb_camera_get_image(camera);

    red = 0;
    green = 0;
    blue = 0;

    for (int x = 0; x < width; x++)
    for (int y = 0; y < height; y++) {
      red = wb_camera_image_get_red(image, width, x, y);
      green = wb_camera_image_get_green(image, width, x, y);
      blue = wb_camera_image_get_blue(image, width, x, y);
      printf("red=%d, green=%d, blue=%d", red, green, blue);
        }
```




```
green=57, blue=63red=238, green=57, blue=64red=239, green=57, blue=64red=223, green=69, blue=75red=225, green=68,
blue=74red=232, green=68, blue=74red=233, green=68, blue=75red=234, green=69, blue=75red=233, green=69, blue=75re
green=68, blue=74red=228, green=68, blue=74red=228, green=67, blue=73red=227, green=67, blue=73red=228, green=67,
blue=75red=229, green=70, blue=76red=227, green=69, blue=76red=223, green=69, blue=75red=216, green=68, blue=74re
green=66, blue=72red=180, green=63, blue=69red=178, green=62, blue=68red=180, green=63, blue=69red=188, green=64,
blue=71red=216, green=65, blue=71red=223, green=65, blue=71red=228, green=65, blue=72red=232, green=66, blue=72re
green=61, blue=67red=231, green=57, blue=63red=230, green=55, blue=61red=233, green=60, blue=66red=237, green=65,
blue=71red=239, green=64, blue=71red=237, green=64, blue=70red=236, green=63, blue=69red=235, green=62, blue=68re
green=61, blue=67red=236, green=60, blue=66red=236, green=60, blue=66red=237, green=60, blue=66red=237, green=59,
blue=65red=237, green=58, blue=64red=237, green=58, blue=64red=237, green=57, blue=63red=237, green=57, blue=62re
green=52, blue=58red=232, green=49, blue=54Looks like I found a red blob.
```

08 Color Recognition

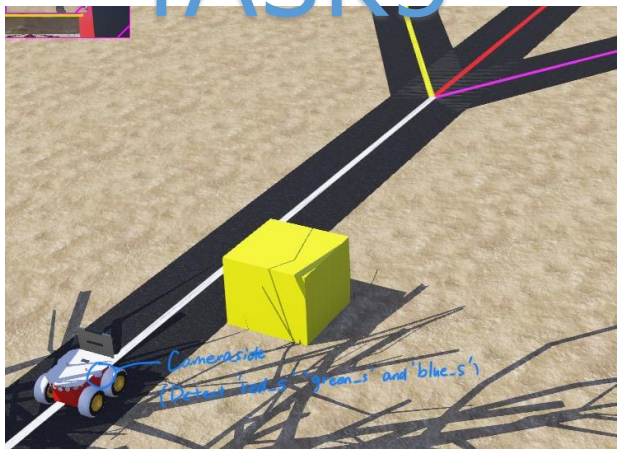


Color Recognition

Determine judgement condition. Combine ‘tracking’ and ‘recognition’ together.

TASK2

TASK5



Simulate in my world

```
if ((green > 220) && (red < 50)&&(blue <50)){
  back_left_speed = 0, back_right_speed = 0;
  front_left_speed = 0, front_right_speed = 0;
  wb_motor_set_velocity(front_left_wheel, front_left_speed);
  wb_motor_set_velocity(front_right_wheel, front_right_speed);
  wb_motor_set_velocity(back_left_wheel, back_left_speed);
  wb_motor_set_velocity(back_right_wheel, back_right_speed);
  }
if ((red > 1.5* green) && (red > 3 * blue)&&(green>1.5*blue)){

  back_left_speed = 0, back_right_speed = 0;
  front_left_speed = 0, front_right_speed = 0;
  wb_motor_set_velocity(front_left_wheel, front_left_speed);
  wb_motor_set_velocity(front_right_wheel, front_right_speed);
  wb_motor_set_velocity(back_left_wheel, back_left_speed);
  wb_motor_set_velocity(back_right_wheel, back_right_speed);
  }
```

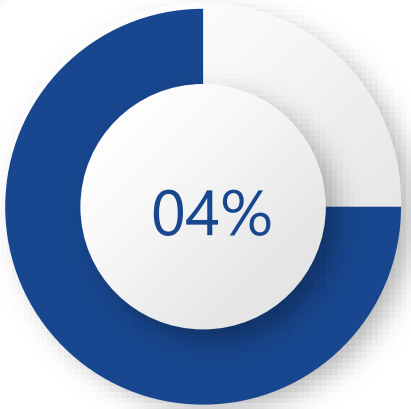
To detect Green.

To detect Orange.

```
344  if ((red_s> 3 * green_s) && (red_s> 3 * blue_s)){
345  key='0';//red
346 }
347
348
349  else if ((red_s> 3 * green_s) &&(blue_s> 3 * green_s)){
350  key='1';//purple
351 }
352
353
354  else if ((red_s> 3 * blue_s) &&(green_s> 3 * blue_s)){
355  key='2';//yellow
356 }
357
358  else if((red_s> 1.5 * green_s) && (red_s> 3 * blue_s)&&(green_s> 1.5 * blue_s
359  //orange box is detected, throw the fish food
360  key='3';}
```

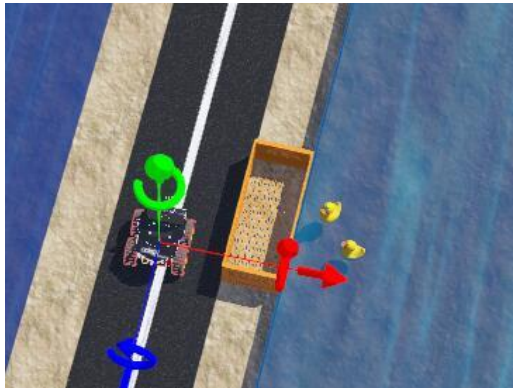
combine

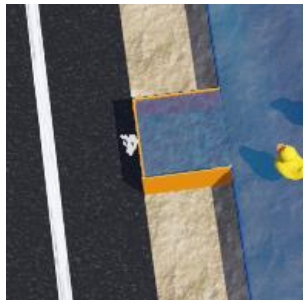
08 Color Recognition

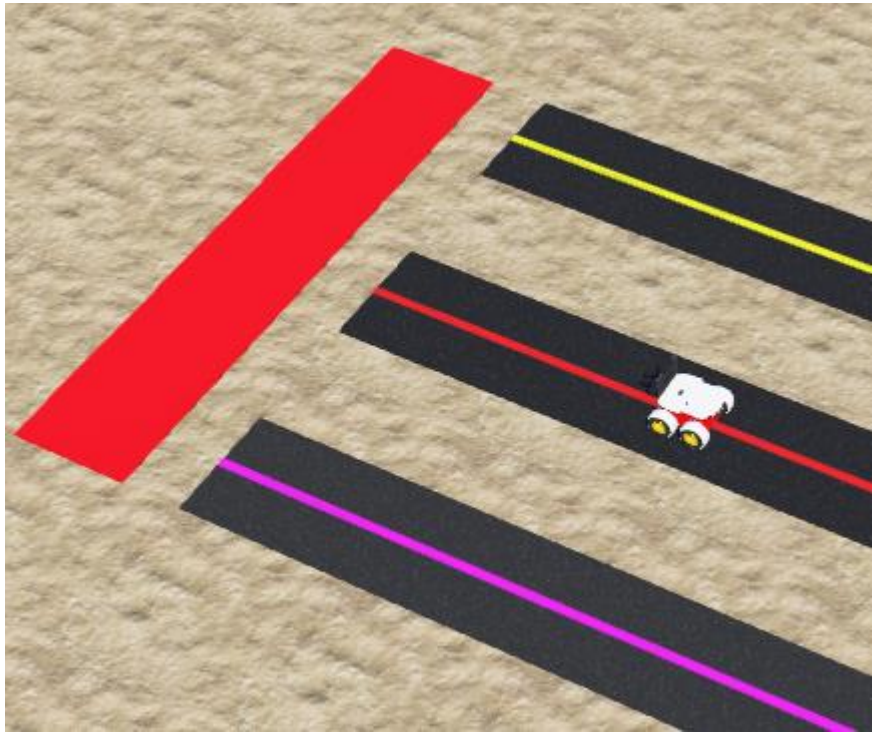


Problem Settled

- Change box size
- Fix the code about stopping the rover







```
if (seed) {
```

If(...).....

```
else if((red_l> 3 * green_l) && (red_l> 3 * blue_l)&&(red_r> 3 * green_r) && (re
//red box at the end is detected, stop
key='4';}

case '4':
          left_speed =0.0; right_speed = 0.0;
```

Object Releasing

Xinyu Zhang

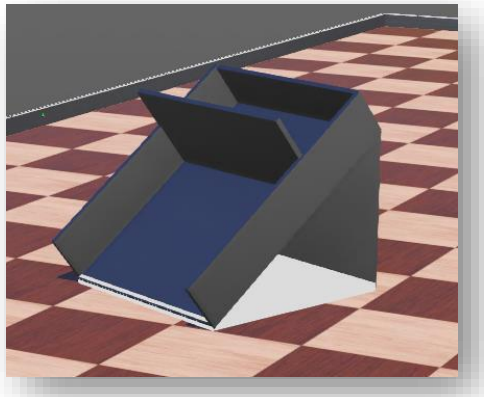
09

Object Releasing

Possible Methods

Compounded slope

- Easy for the fish food to roll down
- With a motor to control the gate to open



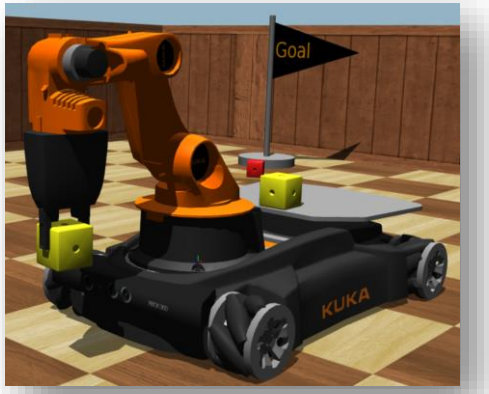
ROBOTIQ 3F Gripper

- 3-finger adaptive robot gripper
- Complex programming

ROBOTIQ 3F Gripper in Webots

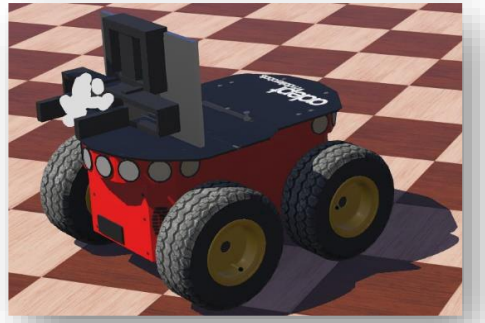
KUKAs' youBot

- With a mechanical arm on the robot
- Height exceeds the height of the arch



Pioneer 3-DX Gripper

- High suitability with the rover
- Controller easily modified
- More flexible and higher functional than the slope



09

Object Releasing

Code Behavior

Function Definition

Define each function for the gripper's actions.

```
static void initialize() {
  /* necessary to initialize Webots */
  wb_robot_init();

  time_step = wb_robot_get_basic_time_step();

  gripper_motors[0] = wb_robot_get_device("lift motor");
  gripper_motors[1] = wb_robot_get_device("left finger motor");
  gripper_motors[2] = wb_robot_get_device("right finger motor");
  wheel_motors[0] = wb_robot_get_device("left wheel");
  wheel_motors[1] = wb_robot_get_device("right wheel");
  // Specify velocity control mode
  wb_motor_set_position(wheel_motors[0], INFINITY);
  wb_motor_set_position(wheel_motors[1], INFINITY);
  wb_motor_set_velocity(wheel_motors[0], 0.0);
  wb_motor_set_velocity(wheel_motors[1], 0.0);
}

void step(double seconds) {
  const double ms = seconds * 1000.0;
  int elapsed_time = 0;
  while (elapsed_time < ms) {
    wb_robot_step(time_step);
    elapsed_time += time_step;
  }
}
```

```
void lift(double position) {
  wb_motor_set_velocity(gripper_motors[0], GRIPPER_MOTOR_MAX_SPEED);
  wb_motor_set_position(gripper_motors[0], position);
}

void moveFingers(double position) {
  wb_motor_set_velocity(gripper_motors[1], GRIPPER_MOTOR_MAX_SPEED);
  wb_motor_set_velocity(gripper_motors[2], GRIPPER_MOTOR_MAX_SPEED);
  wb_motor_set_position(gripper_motors[1], position);
  wb_motor_set_position(gripper_motors[2], position);
}

void moveForwards(double speed) {
  wb_motor_set_velocity(wheel_motors[0], speed);
  wb_motor_set_velocity(wheel_motors[1], speed);
}

void turn(double speed) {
  wb_motor_set_velocity(wheel_motors[0], speed);
  wb_motor_set_velocity(wheel_motors[1], -speed);
}

void stop(double seconds) {
  wb_motor_set_velocity(wheel_motors[0], 0.0);
  wb_motor_set_velocity(wheel_motors[1], 0.0);
  step(seconds);
}
```

09

Object Releasing

Code Behavior

Action Implementation

Implement each function to carry out the releasing task.

```
int releasing_food() {
    stop(0.5);
    moveFingers(0.02);
    step(0.5);
    turn(2.0);
    step(2.0);
    moveForwards(1.0);
    step(2.5);
    stop(0.5);
    moveFingers(0.06);
    step(0.5);
    stop(1.0);
    moveForwards(-1.0);
    step(2.5);
    stop(0.5);
    turn(-2.0);
    step(2.0);
    stop(0.5);
    moveFingers(0.04);
    step(0.5);
    food=0;
    return 0;
}
```

10

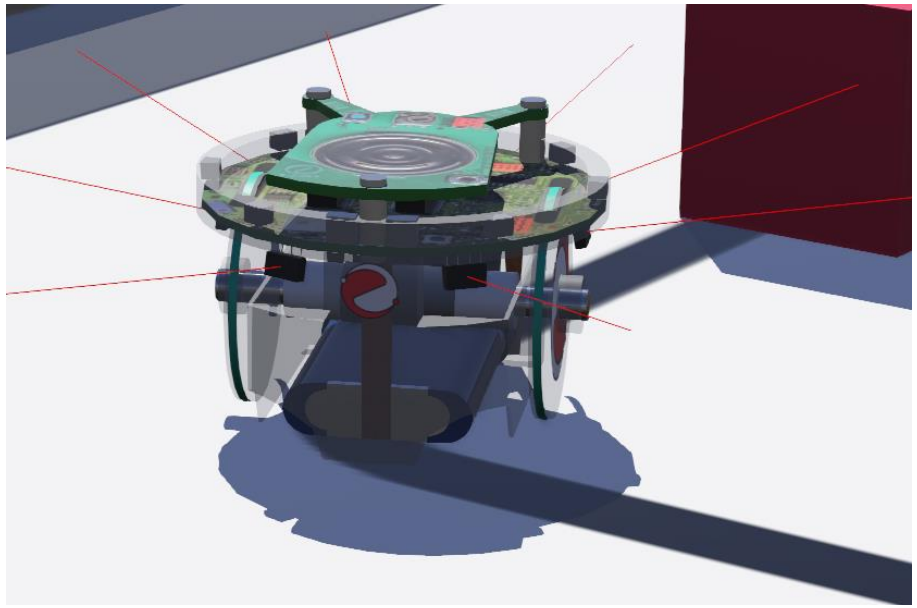
Scheme and assembly of Robot

Xinyue Lan

Scheme of robot and Assembly of code

Choose a proto

Which proto should we use as the basis of our rover? – Pioneer 3AT



Advantage: available tracking tutorial; extensible

Disadvantage: small size as a table-based robot (7cm*7cm*6cm)

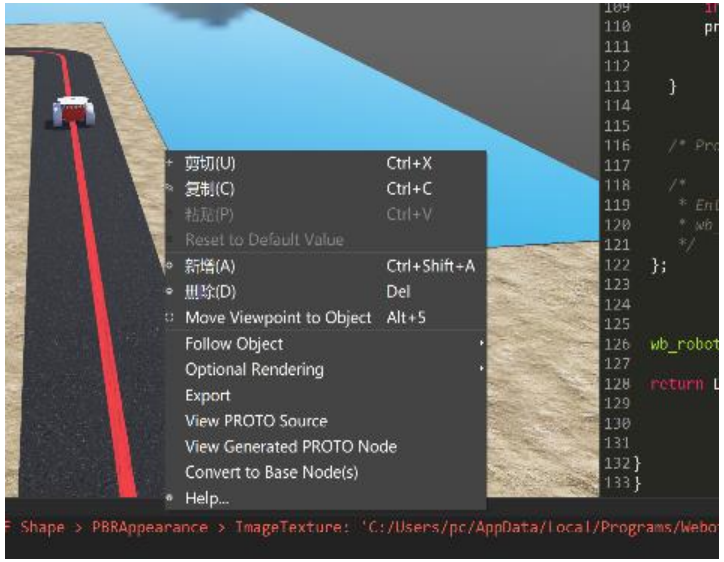
Advantage: boarded sonars; extensible; proper size

Disadvantage: relatively less tutorials

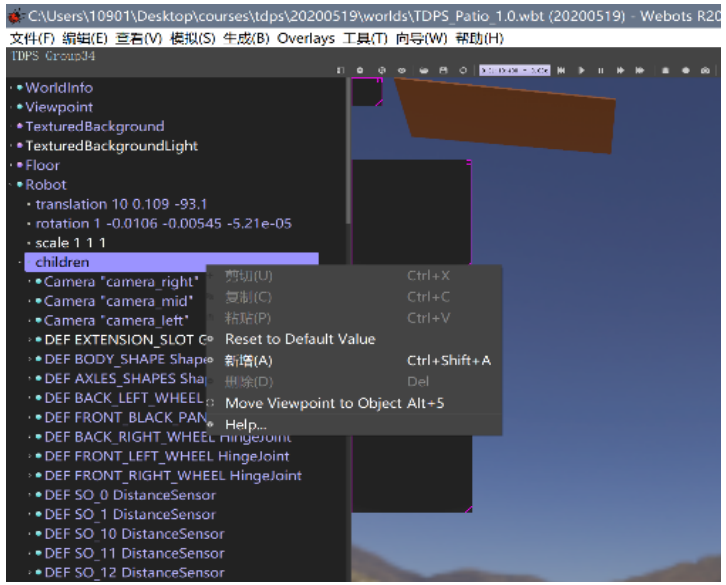
Scheme of robot and Assembly of code

Give tutorials

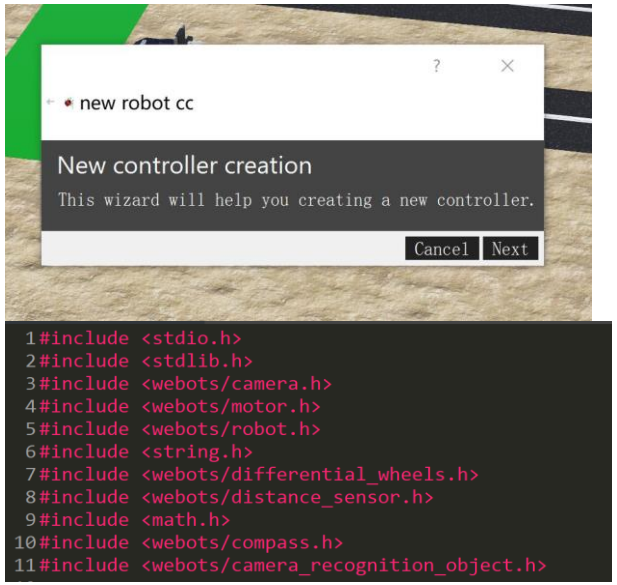
How to programs on nodes in Webots? – tutorials about Webots



Add a robot node into world



Convert a robot to base nodes



Set up a new controller and call libraries of sensors

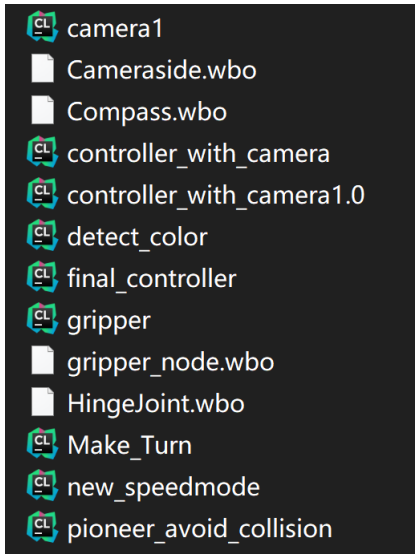
Scheme of robot and Assembly of code

Assemble the robot

How to assemble the code of each part and test them? – organize the structure of code

Setting 'signs' for function calling

Individual controllers for each task → Define functions in program → Insert the code into functions → Call the functions properly in the main loop




```
////////////////////////////////the function of object recognition
 int object_recognition() {
 int number_of_objects = wb_camera_recognition_get_number_of_objects(
```

```
    }
/////////The main function of releasing fish food
    int releasing_food() {
                  stop(0.5);
```

```
////////////////////////////////////////////the funtion of tracking
  int tracking(){
   const unsigned char* image_r = wb_camera_get_image(camera_right);
   const unsigned char* image_l = wb_camera_get_image(camera_left);
```

```
/* Main Loop ---------------------------
while (wb_robot_step(TIME_STEP) != -1) {
          tracking();
          object_recognition();
}
```

```
int seed=1;

int food=1;

int finish=0;
```

```
if (find_arch) {
          cross_arch = 1;
          finish=1;
```

```
 if(food==1){
releasing_food();}
```

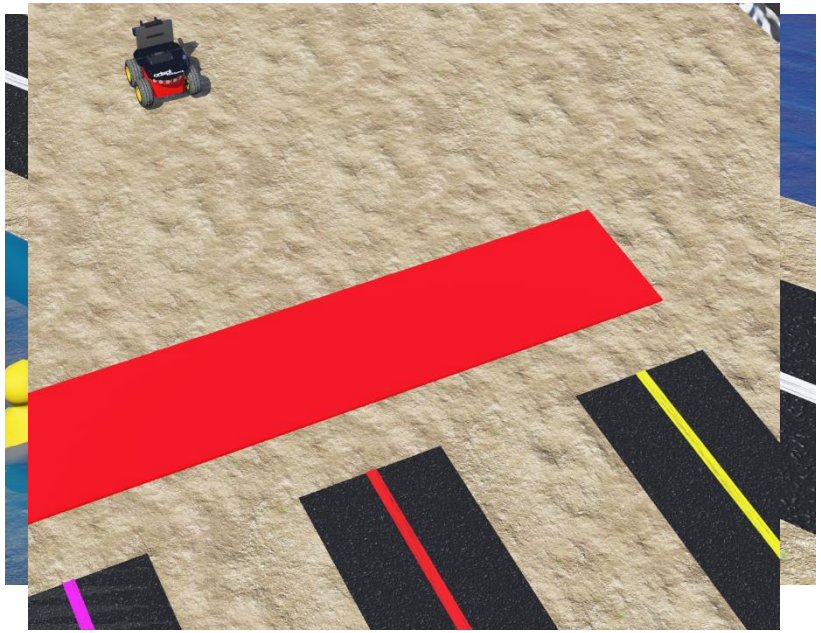
```
if ((red_s> 3 * green_s) && (red_s> 3 * blue_s)){
key='0';//red
seed=0;}
```

Scheme of robot and Assembly of code

Test and consummation

Is the program efficient for the tasks in the patio? – test and detailed correction

- Why does the simulation speed keep very low?
- How to ensure that fish food would fall in pond rather than out of pond?
- Why does the rover deviate from the track after equipping the fish food?
- How to let the camera recognition start after releasing fish food?
- Why does the rover turn again after releasing fish food due to the detection of orange box again?
- How to let the rover stop at the destination rather than rushing out?
- …



The only solution to above problems: TEAMWORK

GLASGOW FANTASTIC IO

TEAM 34

Fantastic 10

The End

Thanks!

University of Glasgow

电子科技大学 University of Electronic Science and Technology of China