LACOC - Locomotion of Artificial Creatures Online Contest

Kubota Lab., Tokyo Metropolitan University, JAPAN

Organizing Team Members
Advisor: Prof. Naoyuki Kubota, Prof. Naoyuki Takesue, Prof. Kazuyoshi Wada;
Chair: Dr. Wei Hong Chin; Co-Chair: A.R. Anom Besari;
Facilitator: Noriko Takase, Azhar Aulia Saputra, Fernando Ardilla, Mohamad Yani, Li Lianchao

This workshop provides the participants with the practice on the design of locomotion patterns for multi-legged robots using ODE (Open Dynamics Engine, https://www.ode.org/). Participants don't need the high level of programming skill but they are asked to install ODE on Windows, Macintosh or UNIX PC beforehand.

1. Introduction

Various types of robots have been applied to educational fields. Basically, there are three different aims in robot edutainment (education with entertainment). One is to develop knowledge and skill of students through the project-based learning by the development of robots (Learning on Robots). Students can learn basic knowledge on robotics itself by the development of a robot. The next one is to learn the interdisciplinary knowledge on mechanics, electronics, dynamics, biology, and informatics by using robots (Learning through Robots). The last is to apply human-friendly robots instead of personal computers for computer assisted instruction (Learning with Robots). A student learns (together) with a robot. In addition to this, such a robot can be used for supporting teachers by the teaching to students and the monitoring of the learning states of students. An educational partner robot can teach something through interaction with students in daily situation. Furthermore, the robot can observe the state of friendship among students. This is very useful information for teachers, because it is very difficult for a teacher to extract such information from the daily communication with students. The human-robot co-learning is a kind of Learning with Robots and we have developed various types of robot partners. To enhance the natural communication and interaction with students, we have to design the robot motion. Therefore, we focus on Learning through Robots in this workshop.

This workshop provides the participants with the practice on the design of locomotion patterns for multi-legged robots using ODE (Open Dynamics Engine, https://www.ode.org/) from the viewpoint of Learning through Robots. Basically, participants don't need the programming skill, but we assumed that participants install ODE on Windows, Macintosh or UNIX PC beforehand. First, participants learn the basic mathematical formulation of robot geometry and kinematics by trigonometric functions. Next, participants understand how to conduct multi-legged locomotion by computer simulations with ODE, and design locomotion patterns by text files as a group work. Finally, participants join a flag strike robot contest.

2. Open Dynamics Engine (ODE)

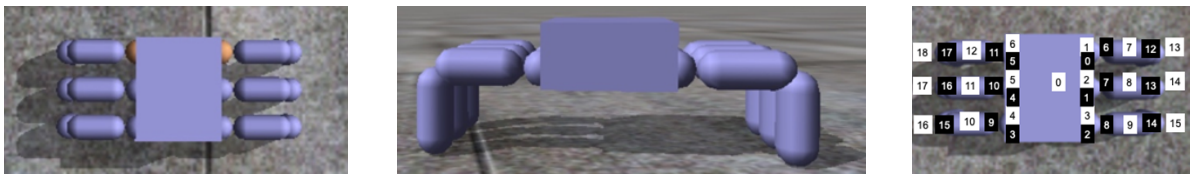
We conduct programming by using ODE (Open Dynamics Engine, https://www.ode.org/), which is available for free. ODE is a suitable platform for learning physics because it can perform various 3D dynamics simulations at high speed on Windows, Linux and Mac. Since both 3D graphics using Open GL and collision detection mechanism are incorporated, ODE can facilitate hardware design of complicated shapes such as locomotion robots, and difficult design and motion planning of control systems with multiple degrees of freedom. In general, we would like to expand the design to complicated locomotion patterns after understanding the local physical phenomena by simulating the physical system from the design of a simple robot with few degrees of freedom. However, thanks to using ODE, it is possible for us to proceed from the intuitive design of locomotion patterns in the opposite way to the understanding of the physical system simulation and the theoretical design of robot shapes and the control system design.

3. Multi-legged Locomotion

3-1. Six-legged Robot Simulation

We use a six-legged robot shown in Fig.3-1. The robot has 3 joints on each leg. Each leg is assigned its corresponding identification number (white number) by clockwise from the right front leg in Fig.3-1(c). Basically, we need to set 18 joint angles to make a posture in the locomotion. The color of 1st and 6th parts of the robot is different from others for you to know the moving direction.

Kinematic model of the robot's leg is shown in Fig. 3-2. The initial posture (every joint angle is 0°) is shown in Fig.3-1(a). The structure of each leg is the same with others. In Fig. 3-2, the link length of the leg L_0 is 15[mm], 1st link L_1 and 2nd link L_2 are 70[mm], respectively. Each joint is assigned its corresponding identification number (black number) by clockwise from the right front leg in Fig.3-1(c). Fig. 1-3 shows an example where θ_0 is set 0°, θ_6 is set 30°, and θ_{12} is set 30°. The axis of each revolute joint is as shown in Fig.3-3. The movable range is between -150° and +150°.



(a) Top view (b) Front view (c) ID

White: Parts ID, Robot_Body
Black: Joint ID, R_joint

Fig.3-1 Six-legged Locomotion Robot

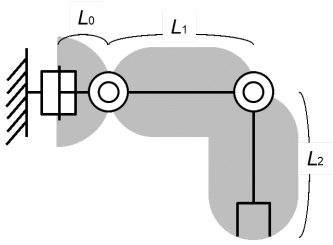


Fig.3-2 Kinematic Robot Leg Model

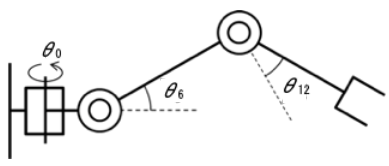


Fig.3-3 Around the Axis (Right Front Leg)

3-2. Control of Six-legged Locomotion Robot

You can control the locomotion by keyboard inputs using the locomotion data (Loco-data.txt). We aim to design the locomotion patterns in the following. In this workshop, you can use two different types of motion control by keyboard; (A) posture-level control and (B) locomotion-level control. One number key is used to move to a single posture in (A) posture-level control, while one number key is used to take a series of postures step by step in (B) locomotion-level control.

In the beginning, you can use (A) posture-level control to confirm the design of a single posture one by one. The robot can move forward by the slow keyboard inputs of [1] – [2] – [3] – [4], while the robot can move back by the slow keyboard inputs of [4] – [3] – [2] – [1]. The robot can turn right by the keyboard inputs of [5] – [6] – [7] – [8], while the robot can turn left by the slow keyboard inputs of [8] – [7] – [6] – [5]. The robot can take the regular position by [0]. Thus, the robot can be controlled by a time series of postures.

In case of (B) locomotion-level control, one key is corresponding to the automatic change of postures used in the posture-level control. For example, [2] in (B) locomotion-level control is the sequential postures of [1] – [2] – [3] – [4] in (A) posture-level control.

4. Design of Locomotion Patterns & Robot

The design process of locomotion patterns and robot is divided into the following phases.

4-1. Exercise of Posture Design

We use “Loco-data.xlsx” to support the design of “Loco-data.txt”. “Loco-data.xlsx” is composed of 2 sheets: “EX (input)” and “Loco-data”. If you change the values in the “EX” sheet (Fig.4-1), its change is reflected in “Loco-data” automatically.

After of changing the value, you should move to ”Loco-data” sheet and select “File”→ ”Save as”. Then, you should choose “Loco-data.txt” as “File Type”→” tub separate text data on Excel's ribbon menu. To avoid overwriting the locomotion data file that was created in the past, please change the name of previous files as a backup beforehand. Please set text file that was made own in same directory (or folder) with the executable file.

Num of Posture	9																	
0 Key	45	0	-45	-45	0	45	20	20	20	20	20	20	0	0	0	0	0	0
1 Key	0	0	-45	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
2 Key	45	0	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
3 Key	45	0	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
4 Key	0	0	-45	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
5 Key	45	0	0	0	0	45	20	20	20	20	20	20	40	0	40	0	40	0
6 Key	45	45	0	-45	0	0	20	20	20	20	20	20	40	0	40	0	40	0
7 Key	45	45	0	-45	0	0	20	20	20	20	20	20	0	40	0	40	0	40
8 Key	45	0	0	0	0	45	20	20	20	20	20	20	0	40	0	40	0	40
ID of Joint	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Position	R,F	R,C	R,B	L,B	L,C	L,F	R,F	R,C	R,B	L,B	L,C	L,F	R,F	R,C	R,B	L,B	L,C	L,F
	Root						1st						2nd					

Fig.4-1 Excel（「Laco-data.xlsx」）

4-2. Exercise of Locomotion Design

1. Posture Design of One Leg (Trial Phase):

   Focus on joints of the right-hind-leg (Black 0, 6 and 12) of the [1] key. Change their values randomly, and confirm the actual posture of their change. You can intuitively understand the relationship between joint angles and its corresponding posture by trial and error.

2. Posture Design of Three Legs (Thinking Phase):

   Focus on joints of three legs swing (e.g., left-middle-leg, right-fore-leg and right-hind-leg) of the [1] key to design . Basically, the robot can stand when three legs contact on the ground.

3. Half cycle of Locomotion Design by two postures (Trial Phase)

Focus on the design of half cycle of locomotion by two postures of [1] and [2] keys. The position of robot should move forward after taking two postures.

4. One cycle of Locomotion Design by four postures (Thinking Phase)

   The robot should repeatedly and smoothly change half cycle of locomotion from right side to left side. Furthermore, you have to consider the continuity from the end of one locomotion step to the start of the next.

5. Concept Design of Locomotion (Imagination Phase)

   Natural creatures can take various types of locomotion patterns. This means that it is difficult for you to consider how to design. Therefore, you should consider the concept for the locomotion design, e.g., [pretty], [elegant], [modern], [fearful], [suspicious], [strange], … Decide the design concept in a group discussion.

6. Creative Design of Locomotion (Creativity Phase)

   Decide the role of locomotion design in a group; Forward [1-4] and Right-turn [5-8].

We will use locomotion-level continuous control using the file “LocoTra-data*.txt” in the online robot contest. The file is composed of a set of postures corresponding to one keyboard input from [0] to [9] at maximal in the following.

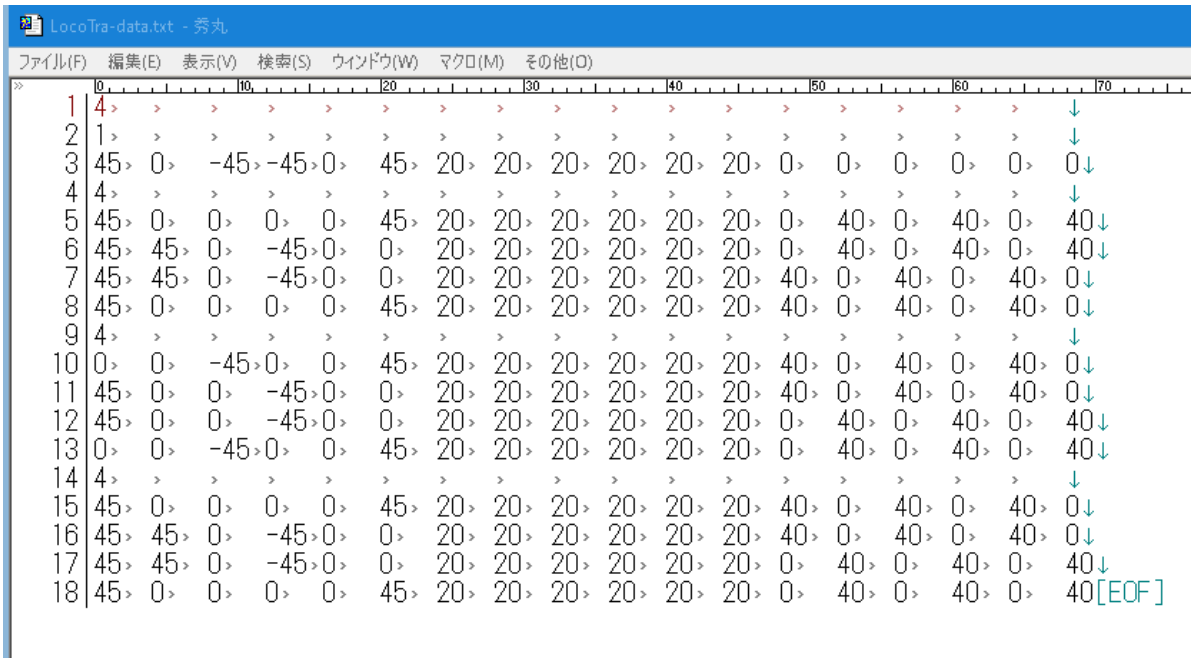


Fig.4-2 Data structure for the Locomotion-level Continuous Control

L.1 [4] : The number of keys used for the control, e.g. 4 keys from [0] to [3] in this example.
L.2 [1] : The number of postures used for the locomotion corresponding to the key [0].
L.3 : Posture data set (Joint angle group of 1 posture corresponding to L.2.)
L.4 [4] : The number of postures used for the locomotion corresponding to the key [1].
L.5-8 : Posture data set (Joint angle group of 4 postures corresponding to L.4.)
L.9 [4] : The number of postures used for the locomotion corresponding to the key [2].
L.10-13 : Posture data set (Joint angle group of 4 postures corresponding to L.9.)
L.14 [4] : The number of postures used for the locomotion corresponding to the key [3].
L.15-18 : Posture data set (Joint angle group of 4 postures corresponding to L.14.)

In this example, the robot can move forward by the keyboard inputs [2]. The robot can turn right by the keyboard inputs of [3], while the robot can turn left by the keyboard inputs of [1]. The robot can take the regular position by [0].

4-3. Design of Robot

If you want to create your own more special locomotion robot in this contest, you may also refer to the instructions in this part. If you just want to use prefabricated robot provided by us in this contest and only design your own robot's locomotion method, you may skip this part. If you modify the robot6legs.h and mainR*.cpp files we provided and re-compile the robot, your will get your own unique robot. We will introduce the parts needed to modify the robot.

Please open robot6legs.h, you will see some codes you may want to modify.

1. The initial size of the robot. (length, width, height, etc.)

```
#define ROBOT_BODY_W 0.1        // width of robot body
#define ROBOT_BODY_D 0.12       // length of robot body
#define ROBOT_BODY_H 0.05       // height of robot body
#define ROBOT_LEG_R 0.015       // length of robot leg0
#define ROBOT_LEG_L 0.04        // length of robot leg1 & leg2
```

2. The initial location of robot's each body parts. (x, y, z)

```
dReal Start_Pos[3] = { 0.0, 0.0, ROBOT_LEG_L + ROBOT_LEG_R };
```

Using this value, the robot body (main body) and the initial position of each leg are specified as follows.

```
dReal Robot_Pos[ROBOT_NUM][3] = {
//the position of the main body
{ Start_Pos[0], Start_Pos[1], ROBOT_BODY_H * 0.5 + Start_Pos[2] }, ...
//the position of the front leg
{ 0.5*ROBOT_BODY_W + Start_Pos[0], 0.5*ROBOT_BODY_D - ROBOT_LEG_R +
Start_Pos[1], ROBOT_LEG_R + Start_Pos[2] }, ...
};
```

3. Weight of each body part

```
dReal Robot_Weight[ROBOT_NUM] = { 0.125, 0.055, 0.055, ...};
```

4. Joint settings

```
#define RJOINT_NUM 18                //Total number of joints to make in the
robot
...
dJointID Rjoint [RJOINT_NUM];        //Variable number for joint
...
/*
Decide and declare the position (x coordinate, y coordinate, z coordinate)
of the joint to be made in the robot, target angle (initial angle at
initialization), current angle, and angular velocity.
*/
dReal Rjoint_x [RJOINT_NUM] = { 0.5*ROBOT_BODY_W + Start_Pos[0],
0.5*ROBOT_BODY_W + Start_Pos[0], ...};
dReal Rjoint_y [RJOINT_NUM] = { 0.5*ROBOT_BODY_D - ROBOT_LEG_R +
Start_Pos[1], Start_Pos[1], ...};
dReal Rjoint_z [RJOINT_NUM] = { ROBOT_LEG_R + Start_Pos[2], ROBOT_LEG_R +
Start_Pos[2], ...};
dReal Joint_Angle [RJOINT_NUM] = { 0.25 * M_PI, 0.0, ...};
dReal pre_Joint_Angle [RJOINT_NUM] = { 0.25 * M_PI, 0.0, ...};
dReal Joint_AnglVel [RJOINT_NUM] = { 0.0};
```

5. function for robot design in robot6legs.h

```
//This part of the function is in robot6legs.h
/*
Those are the functions for generating Locomotion Robot. If you want to
change the feet number of the robot, besides the modification for those
fuction here, you also need to modify the joints number, joint connection
method, Loco*.txt and LocoTra*.txt to compile the robot normally.
Therefore, it is not recommended to make too many changes in this workshop.
*/
void make_Robot()
void drw_Robot()
```

The next part of the content needs to be modified in mainR*.cpp.

6. some import function for robot design in mainR*.cpp

```
//This part of the  function is in mainR*.cpp
...
double AngMin=1.2,              //  modify this parameter can change the
movement speed of the robot.
...
void readdata0()                //  read posture-level control data (Loco-
data-6legs.txt)
void readdata1()                //  read locomotion-level data (LocoTra-
data-6legs.txt)
void control0()                 //  Transition to the next Posture
void control1()                 //  One step locomotion-based Control
void command_func0(int cmd)     //  posture-level contorl
void command_func1(int cmd)     //  Locomotion-level contorl
```

4-4. Compile Your Own Robot

We assume that you have built your own local compilation environment according to the environment construction guide. If you have not built the ode compilation environment, please see the environment construction guide link showed below. We recommend using VitualBox in this workshop to reduce the problems that may be encountered in setting up the ODE environments. In this competition, we only support ubuntu 18.04+, macOS 10.14+, and build environment under VirtualBox, and issues related to compilation.

VirtualBox: http://www.google.com/url?q=http%3A%2F%2Fwww.comp.sd.tmu.ac.jp%2FCcS2020%2Fode-virtualbox.pdf&sa=D&sntz=1&usg=AFQjCNFRGSfwnSrkHgjfsM0fgFZxVyZ2WQ

Physical machine: http://www.google.com/url?q=http%3A%2F%2Fwww.comp.sd.tmu.ac.jp%2FCcS2020%2Fode-installation.pdf&sa=D&sntz=1&usg=AFQjCNHelDZYE_TBDW03bMShdsuGLJZ-Fw

Compile Command in Linux Terminal:

```
g++ -I/usr/local/include -o main.out  main.cpp -L/usr/local/lib -lode -
ldrawstuff -lglut -lGL -lGLU -lX11 -lm -lpthread -DdDOUBLE -std=c++11
```

Compile Command in Mac Terminal:

```
g++ -I/usr/local/include -L/usr/local/lib -lode -ldrawstuff -lm -framework GLUT
-framework OpenGL -o main.out main.cpp
```

Run program:

```
./main.out
```

When you execute the generated file, a new window of the robot simulation by ODE is generated. You can control the robot by keyboard on the window. The first robot demo is shown in Fig.4-3

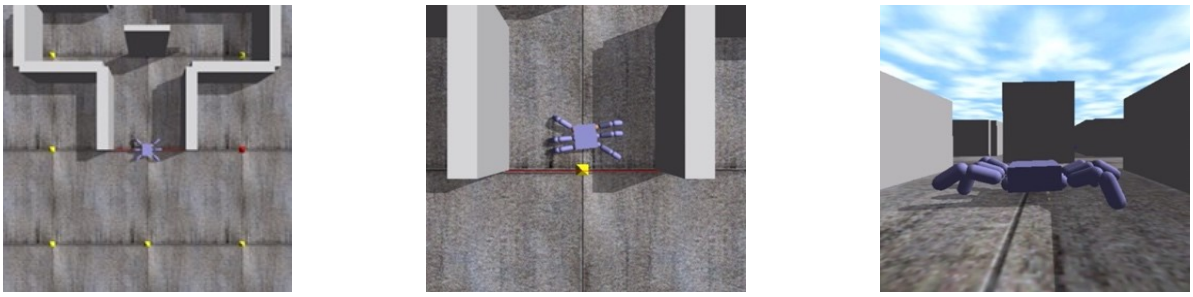


Fig.4-3 First Robot Demo

-[ ] Space key, view angle in simulation change by 3 type.

-[1-4] [5-8] Number key, the robot moves according to Locomotion text file (we will explain this in the following).

-[z] key, Exit the simulation with the output of data file, Pos-data.txt

-[x] key, the robot restart from start position.

You can use the posture-level control (Section 3.2) to move the robot. The robot can move forward by the slow keyboard inputs of [1] – [2] – [3] – [4], while the robot can move back by the slow keyboard inputs of [4] – [3] – [2] – [1]. The robot can turn right by the keyboard inputs of [5] – [6] – [7] – [8], while the robot can turn left by the slow keyboard inputs of [8] – [7] – [6] – [5]. The robot can take the regular position by [0].

5. Online Robot Contest

We have prepared two games for the Contest, Robot-Othello-Game & Flag-Strike-Game. Here we will explain the software related information that needs to be used in this contest.

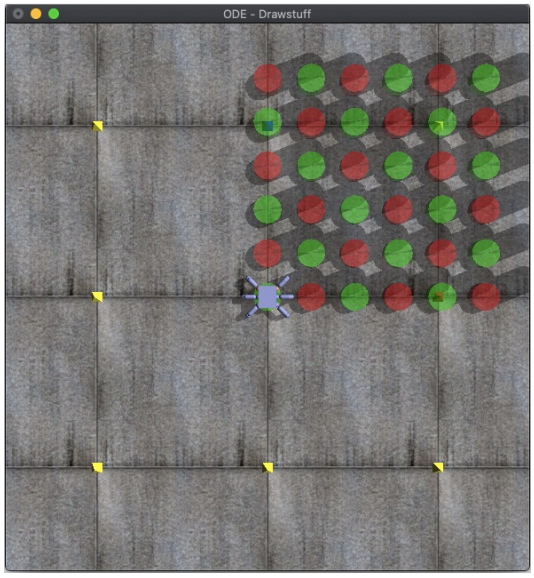


Fig.5-1 Robot-Othello-Game

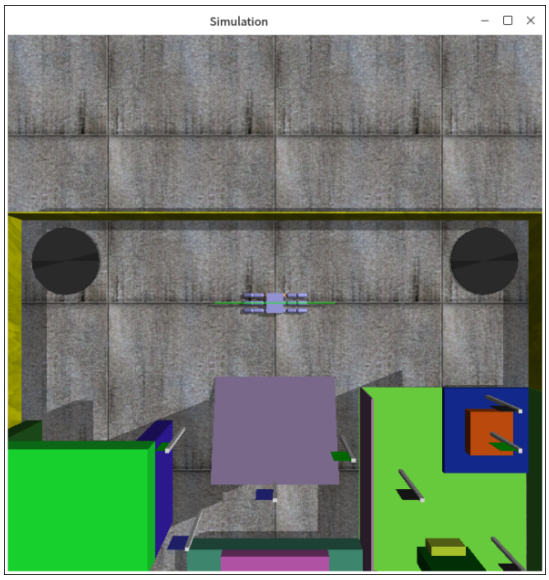


Fig.5-2 Flag-Strike-Game

5-1. Robot-Othello-Game Rules

1. Othello field is consists of 6x6 circular cells.
2. Time limit is 3 minutes.
3. The starting point is one robot left and the other robot in the right.
4. When your robot passes through the color of your cell and the color of your opponent's color, the cell is reversed and becomes your cell.
5. The robot can walk and flip on the opponent's colored disc vertically and horizontally.
6. The score is calculated from the number of discs you have.
7. The winner is the one who gets more discs at the end of time.
8. In the event of an unexpected event, the referee will consult and make a decision.

5-2. Flag-Strike-Game Rules

Robot-Othello-Game Demo & Scene are shown in Fig.5-3 & 5-4.

1. Time limit is 3 minutes.
2. When the center of the robot body enters the point position at the four corners, the color changes and it means that you have taken points.
3. However, the other robot can regain the same point position.
4. Even within the time limit, the match ends when any of the robots has acquired all four-point positions at the four corners.
5. The score of the flag that taken by your robot becomes your score until the end of the match.
6. There are 20 flags with different point in each color.
7. The score is calculated from the number of points you have.
8. The winner is the one who gets more points at the end of time.
9. In the event of an unexpected event, the referee will consult and make a decision.

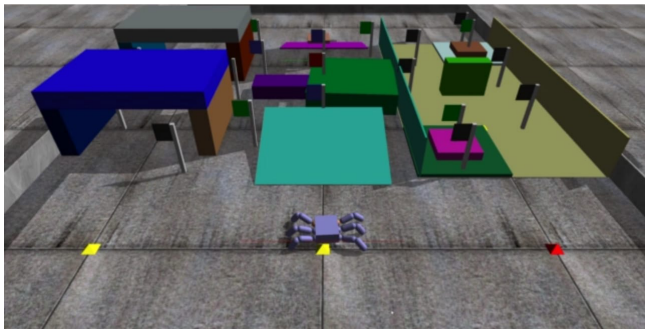


Fig.5-3 Robot-Othello-Game Demo

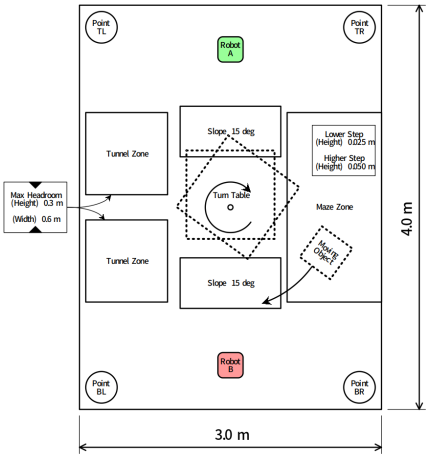


Fig.5-4 Flag-Strike-Game Scene

5-3. Software System

Contstants control the robot program running on our server through the LACOC_Controller. The screen of the contest will be shared with all spectators through zoom.The structure of contest system is shown in Fig.5-5.

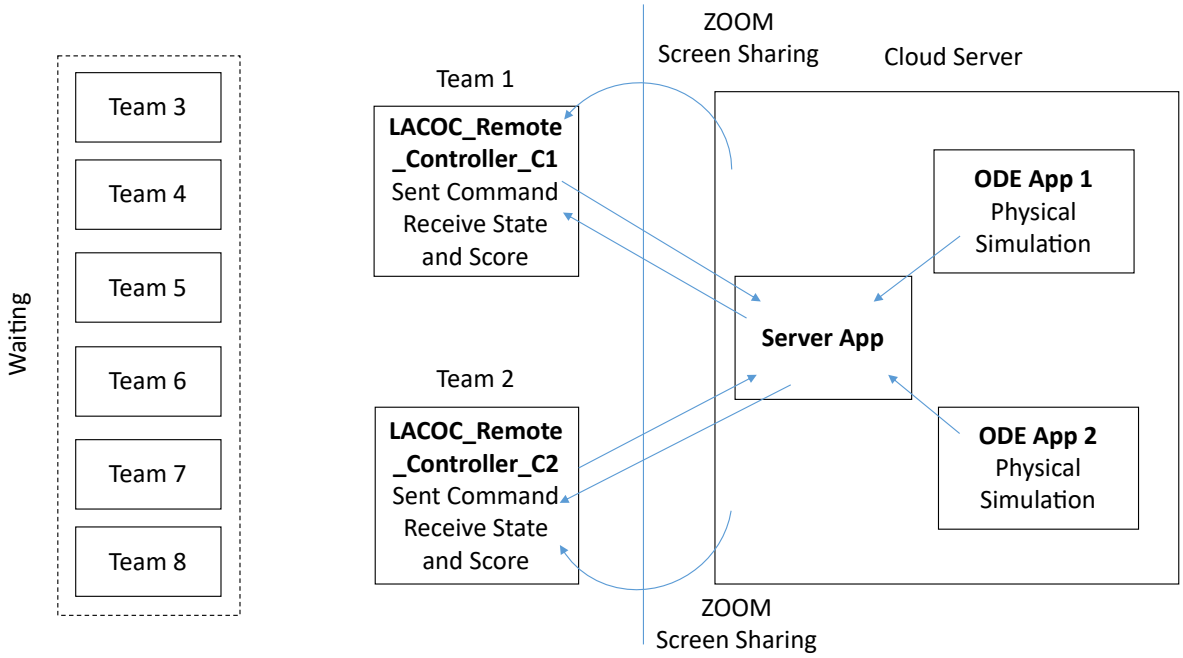


Fig.5-5 Architecture of Software System

5-4. Introduction for Software Interface

In order to get a better ranking at this contest, besides the design of Locomotion patterns, participants also need to understand the basic functions of the following three display windows before the competition. We will explain with flag-strike contest as an example.

The control interface has two windows, which will run on your local machine. The figures is shown below. Fig.5-6 is LACOC_Remote_Controller, which can be used to start the control interface of ode, display the contest state, set some basic parameters, and control the robot movement. Fig.5-7 is ODE Remote-App, which can display the game related information in animation and operate the robot with keyboard.

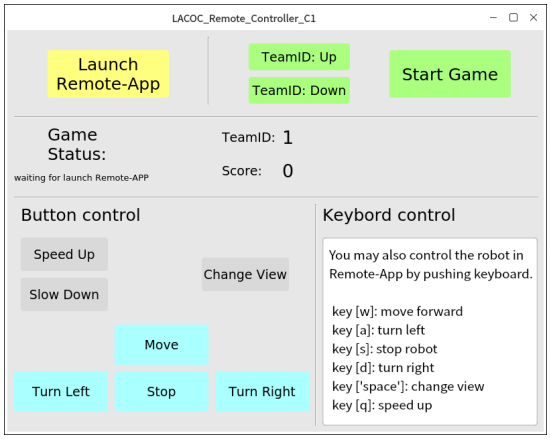


Fig.5-6 LACOC_Remote_Controller

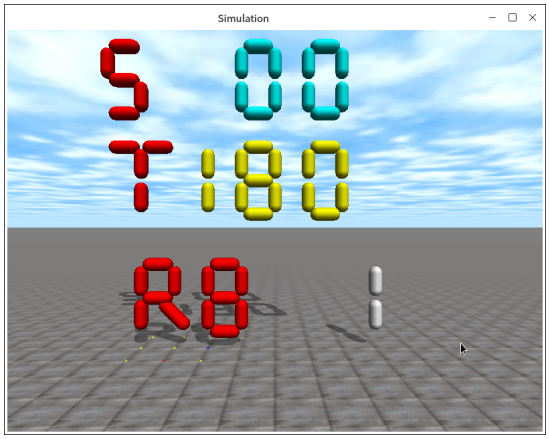


Fig.5-7 ODE Remote_APP

Operation method and explanations for LACOC_Remote_Controller is shown in Fig.5-8.

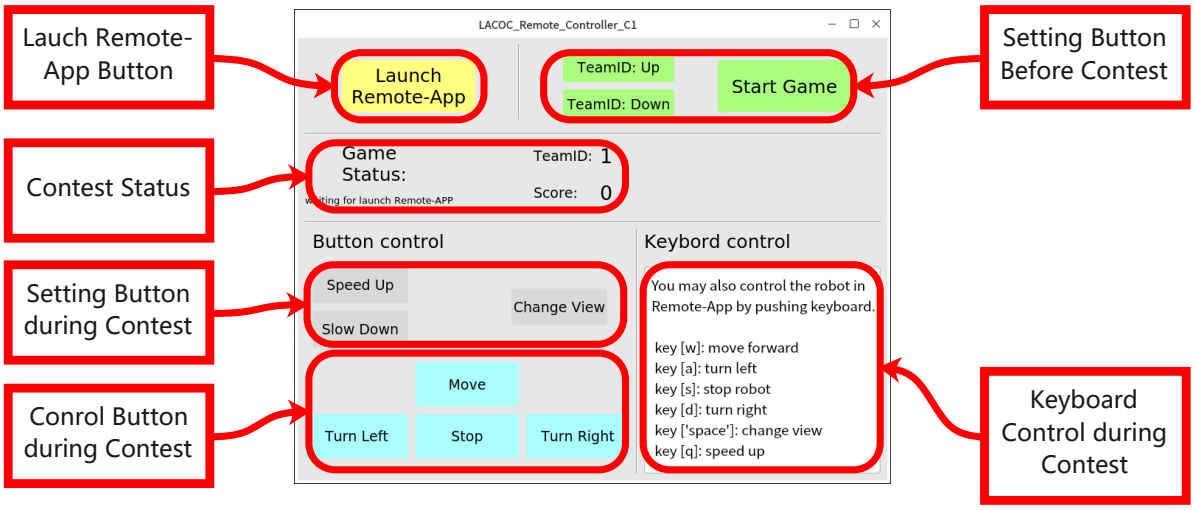


Fig.5-8 LACOC_Remote_Controller's Explanation

After you press the Launch Remote-App Button, the Remote-App built by ODE will start, which is shown in Fig.5-9. The Keyboard control scheme used in Contest is shown in Fig.5-10.

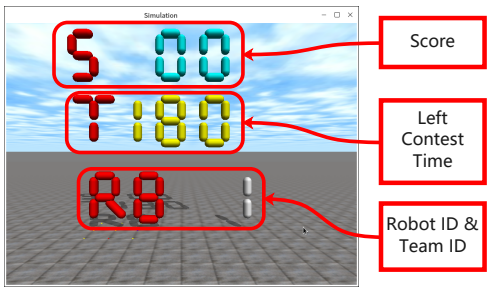


Fig.5-9 Remote-APP's Explanation

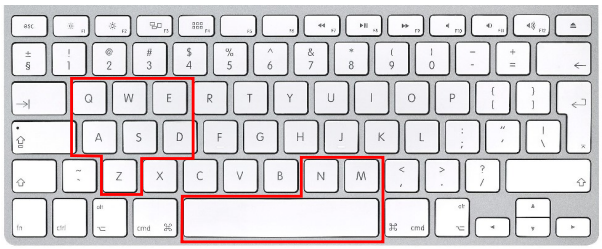


Fig.5-10 Keyboard Control Scheme

Fig.5-11 is the Flag-Strike-Game contest windows which are ruuning on our server. The contest will run two flag-strike-App at the same time, which represents different contestants. Each contestant will control his/her corresponding robot movements through his/her own LACOC_Remote_Controller. This screen will be shared to all participants in the workshop through zoom.

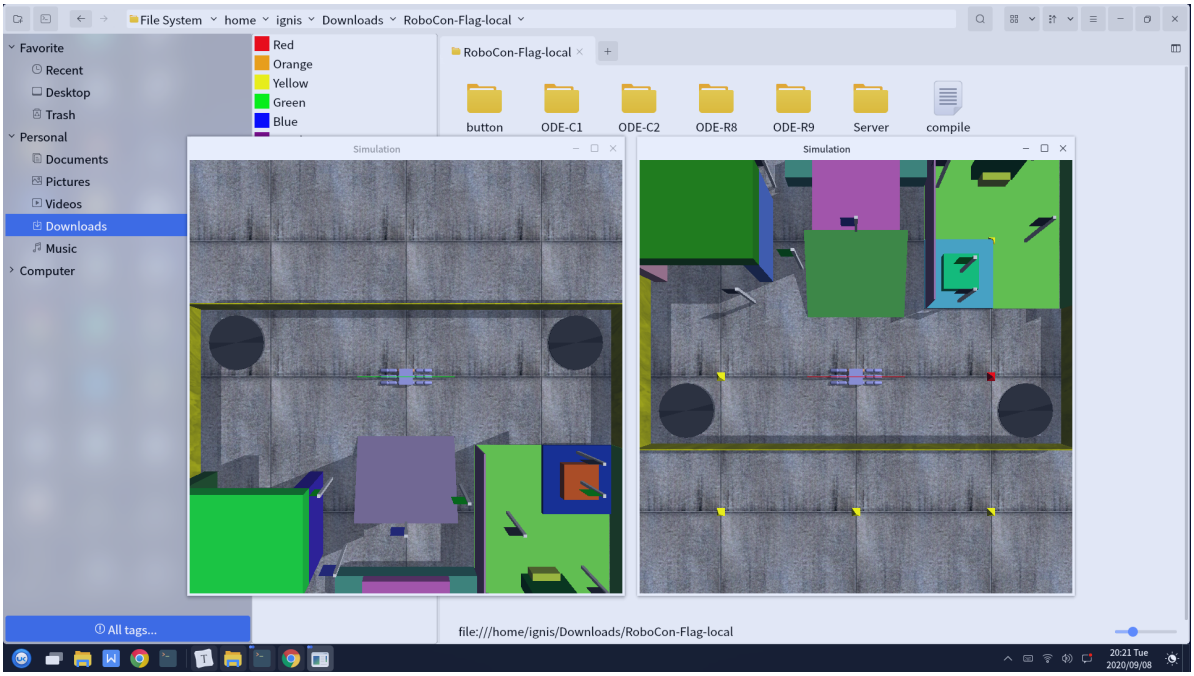


Fig.5-11 Flag-Strike-Game Running on Server

Support By: