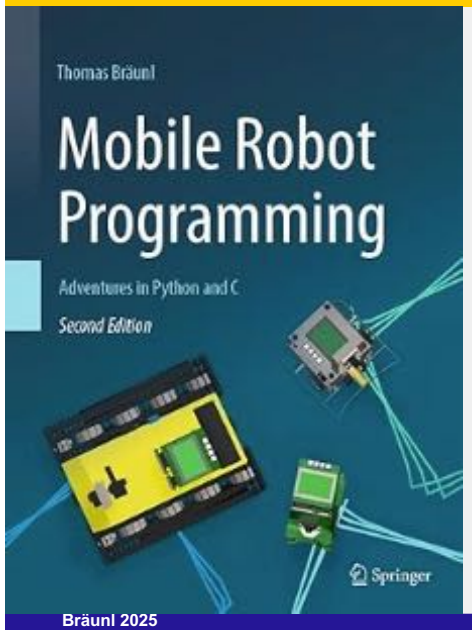
EyeSim
Mobile Robot Simulation

Professor Thomas Bräunl
The University of Western Australia
http://roblab.org/eyesim/
http://roblab.org/eyebot/



Install EyeSim for

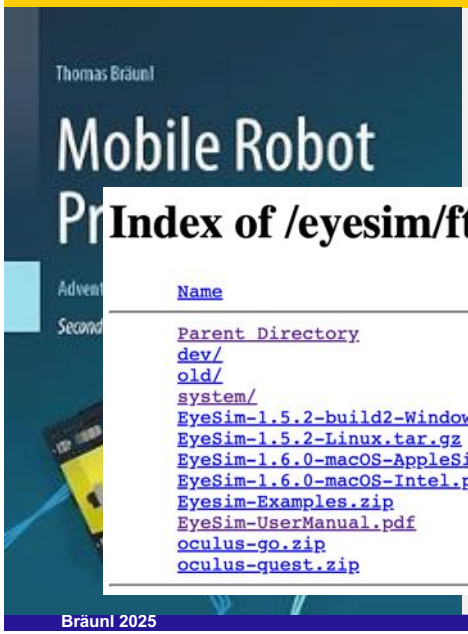
- Mac
- Windows
- Linux

from:

http://ROBlab.org/eyesim/ftp

and get example files:

https://roblab.org/eyesim/ftp/EyeSim-Examples.zip



http://RobLab.org/eyesim/ftp

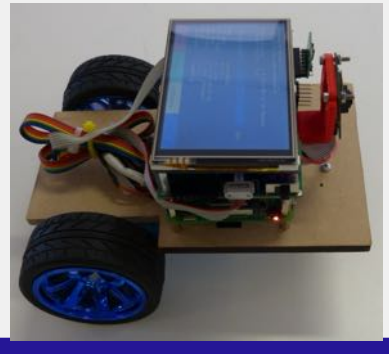
Win+WSL: Use Linux version

Index of /eyesim/ftp

Name	Last modified	Size
Parent Directory		-
dev/	2023-10-28 08:00	-
old/	2025-03-22 20:56	-
system/	2023-10-28 07:58	-
EyeSim-1.5.2-build2-Windows.exe	2023-12-05 18:13	209M
EyeSim-1.5.2-Linux.tar.gz	2023-10-28 07:50	241M
EyeSim-1.6.0-macOS-AppleSilicon-build2.pkg	2025-03-23 00:45	107M
EyeSim-1.6.0-macOS-Intel.pkg	2025-03-22 20:54	107M
Eyesim-Examples.zip	2023-10-28 07:54	60M
EyeSim-UserManual.pdf	2025-03-22 20:55	720K
oculus-go.zip	2023-10-28 07:53	202M
oculus-quest.zip	2023-10-28 07:40	697M

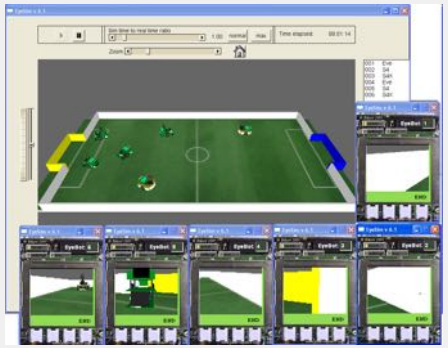
EyeBot Mobile Robots

- Raspberry Pi + EyeBot-I/O-Board
- Camera, 4 x infrared distance sensors
- Differential drive, 2 motors with encoders
- RoBIOS API

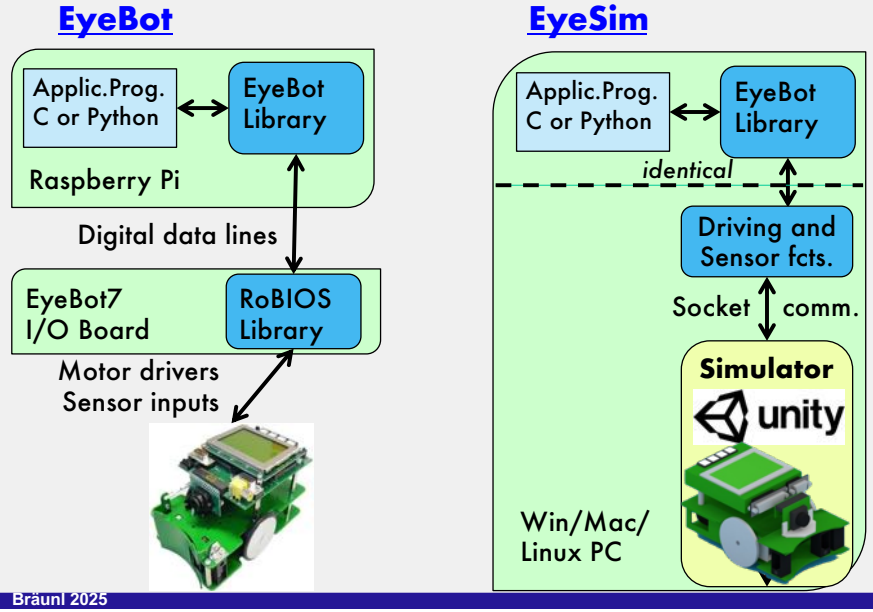


EyeSim VR Robot Simulator

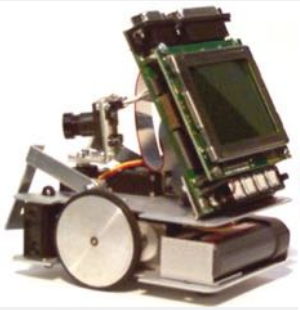
- Free simulator for Mac, Windows
- Simple to use, easy API
- Realistic physics model
- Simulation of all sensors, incl. camera
- VR version for Oculus Quest



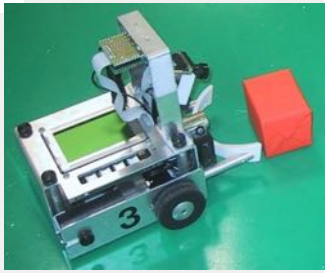
EyeBot and EyeSim



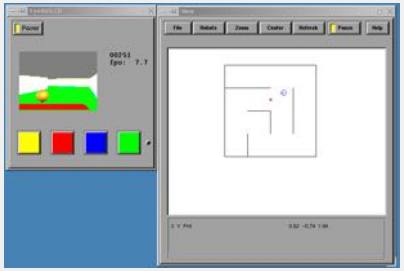
EyeBot Generations

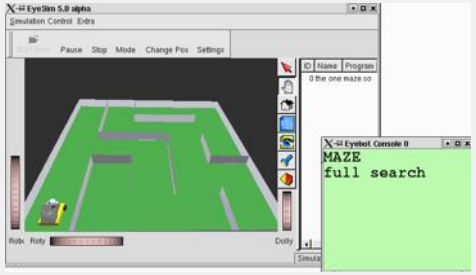


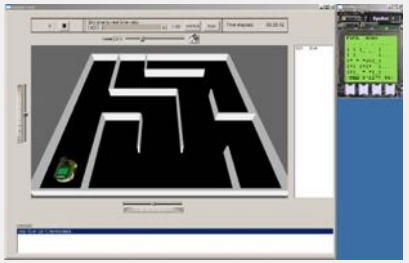
Designs:
Jörg Henne, Frank Sautter,
Joshua Pettit, Richard Meager,
Thomas Bräunl

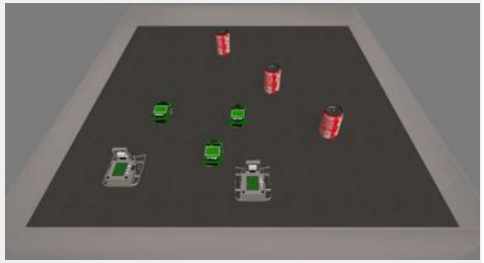


EyeSim Generations



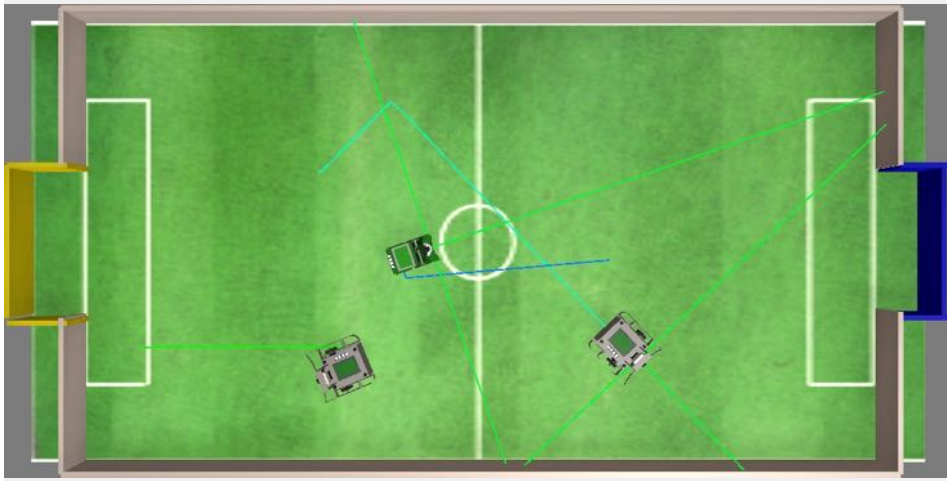






Implementations: E1 Elliot Nicholls, Win 1998, E5 Axel Waggershauser, Linux 2002,
E6 Andreas Koestler, Win-XP 2004, E7 Travis Povey, Unity: Win/Mac/Linux 2017

EyeSim



Multiple robots with track and sensor visualization

RoBIOS API roblab.org/eyebot/robios.html

Distance Sensors

Position Sensitive Devices (PSDs) are using infrared beams to measure distance and need to be calibrated in HDT to get correct
LIDAR (Light Detection and Ranging) is a single-axis rotating laser scanner.

```
int PSDGet(int psd);                              // Read distance value in mm from PSD sensor [1..6]
int PSDGetRaw(int psd);                           // Read raw value from PSD sensor [1..6]
int LIDARGet(int distance[]);                     // Measure distances in [mm]; default 360° and 360 points
int LIDARSet(int range, int tilt, int points);  // range [1..360°], tilt angle down, number of points
```

Camera

```
int CAMInit(int resolution);    // Change camera resolution (will also set IP resolution)
int CAMRelease(void);           // Stops camera stream
int CAMGet(BYTE *buf);          // Read one color camera image
int CAMGetGray(BYTE *buf);      // Read gray scale camera image
```

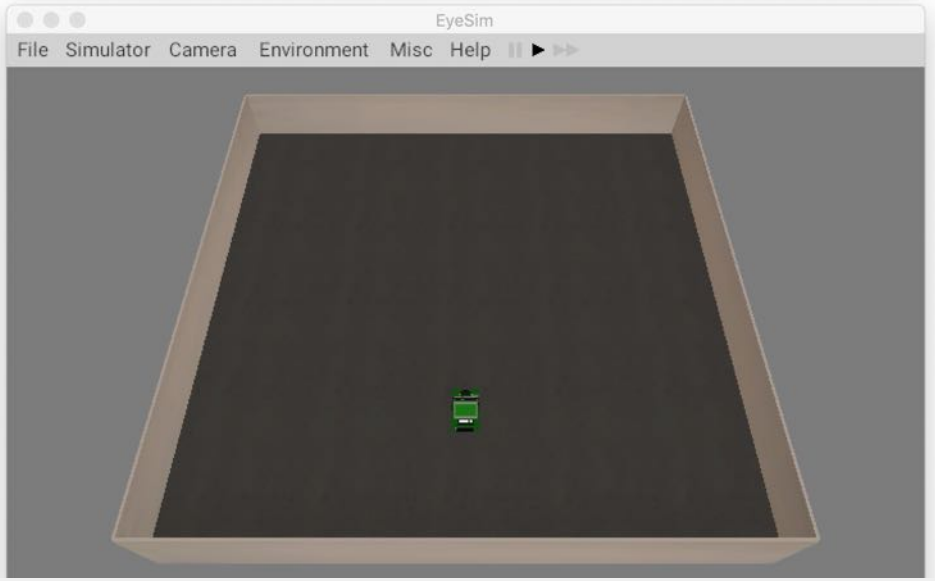
V-Omega Driving Interface

This is a high level wheel control for differential driving. It always uses motor 1 (left) and motor 2 (right).
Motor spinning directions, motor gearing and vehicle width are set in the HDT file.

```
int VWSetSpeed(int linSpeed, int angSpeed);     // Set fixed linSpeed  [mm/s] and [degrees/s]
int VWGetSpeed(int *linSspeed, int *angSpeed);  // Read current speeds [mm/s] and [degrees/s]
int VWSetPosition(int x, int y, int phi);       // Set robot position to x, y [mm], phi [degrees]
int VWGetPosition(int *x, int *y, int *phi);    // Get robot position as x, y [mm], phi [degrees]

int VWStraight(int dist, int lin_speed);        // Drive straight, dist [mm], lin. speed [mm/s]
int VWTurn(int angle, int ang_speed);           // Turn on spot, angle [degrees], ang. speed [degrees/s]
int VWCurve(int dist, int angle, int lin_speed);// Drive Curve, dist [mm], angle (orientation change) [degrees
int VWDrive(int dx, int dy, int lin_speed);     // Drive x[mm] straight and y[mm] left, x>|y|
int VWRemain(void);                             // Return remaining drive distance in [mm]
int VWDone(void);                               // Non-blocking check whether drive is finished (1) or not (0)
int VWWait(void);                               // Suspend current thread until drive operation has finished
int VWStalled(void);                            // Returns number of stalled motor [1..2], 3 if both stalled,
```

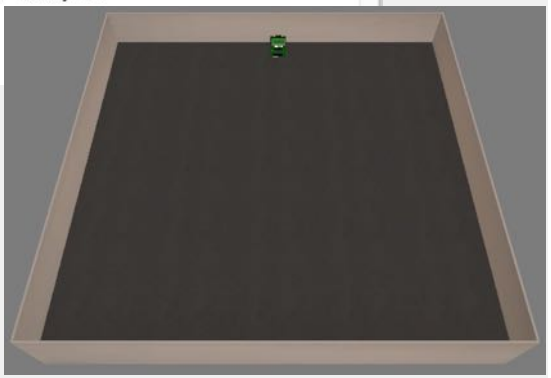

1. Simple Driving

Starting: Click or type EyeSim



Driving ... in Python

```
tb — Python — 71×24
tb-pro:~ tb$ python3
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 05:52:31)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from eye import *
>>> VWSetSpeed(100,0)
Connection established. Handshaking...
Handshake complete. Waiting for server ready...
Server ready. Beginning control.
0
>>>
```



Driving ... in C

Need to create file: straight.c

```
#include "eyebot.h"

int main ()
{ VWSetSpeed(100, 0);
}
```


Compiling and Running C

```
tmp — -bash — 52×26
tb-pro:tmp tb$ gccsim straight.c -o straight.x
tb-pro:tmp tb$ ./straight.x
Connection established. Handshaking...
Handshake complete. Waiting for server ready...
Server ready. Beginning control.
tb-pro:tmp tb$
```


EyeSim Start

A. Manual

- Start EyeSim Program
  - Load environment (or use default)
  - Add any objects, walls or markers
  - Add at least one robot
- Open command window
  - Compile: `gccsim myprog.c -o myprog.x`
    or just: `make`
  - Run: `./myprog.x`

B. Automatic

- Create SIM file
- Run SIM file: `eyesim myprog.sim`

Starting with a SIM File

```
# World description file (maze or world)
world box.maz

# Objects placed in driving area
can 1500 1500 0

# Robot description
S4     800 250 90 search.x
```

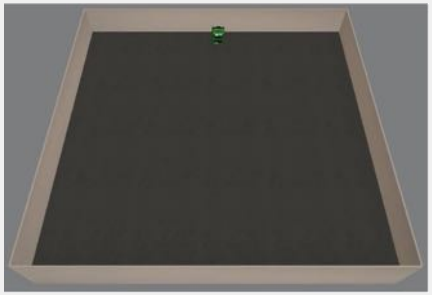
Start with:
% eyesim my.sim

EyeSim Challenge – Stop in Time

Extend program so that robot stops before colliding with wall.

Useful command:

- PSDGet(PSD_Front);



Read Distance Sensors

Python

```
from eye import *

VWSetSpeed(100,0)
while PSDGet(PSD_FRONT) > 200:
VWSetSpeed(0,0)
```

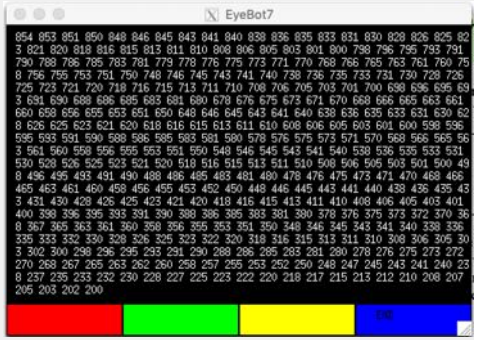
C/C++

```
#include "eyebot.h"

int main()
{ VWSetSpeed(100,0);       // drive
  while (PSDGet(PSD_FRONT) > 200) ;
                           // wait
  VWSetSpeed(0,0);         // stop
}
```

Check and Print

```
from eye import *
LCDMenu("","","","END")
VWSetSpeed(100,0)
dist = 1000
while dist > 200:
  dist = PSDGet(PSD_FRONT)
  LCDPrintf("%d ", dist)
VWSetSpeed(0,0)
KEYWait(ANYKEY)
```

```
#include "eyebot.h"
int main()
{ int dist;
  LCDMenu("","","","END");
  VWSetSpeed(100,0); //drive
  do
  { dist=PSDGet(PSD_FRONT);
    LCDPrintf("%d ", dist);
  } while (dist > 200);
  VWSetSpeed(0,0);   //stop
  KEYWait(ANYKEY);
}
```



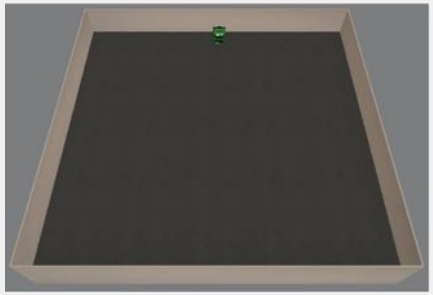
EyeSim Challenge – Stop & Return

Extend program so that:

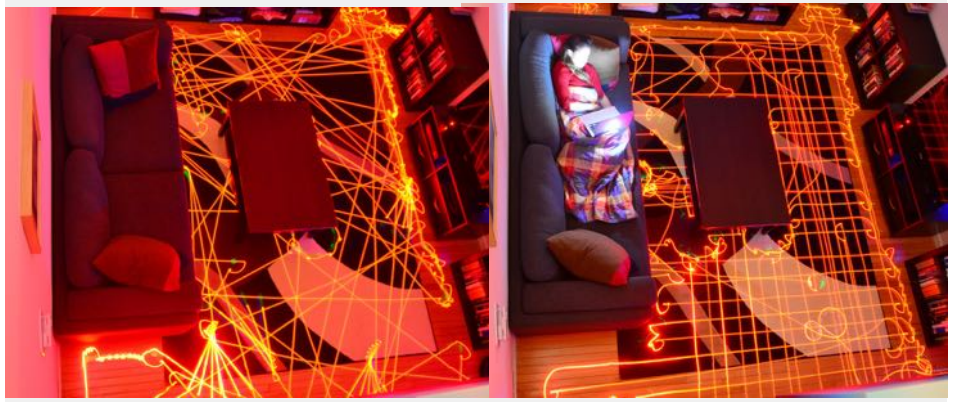
1. Robot stops before colliding with wall
2. Robot turns 180° then drives back to start and stops there

Useful commands:

- PSDGet(PSD_Front);
- VWTurn(180, 45); VWWait();
- VWStraight(100, 50);
- VWGetPosition(&x, &y, &p);
- ...



Random Drive – Structured Drive



Photos by Evan Ackerman / IEEE Spectrum 2016

Random Drive in Python

```
from eye import *
from random import *

safe=300
LCDMenu("","","","END")
while(KEYRead() != KEY4):
    OSWait(100)
    if(PSDGet(PSD_FRONT)>safe and PSDGet(PSD_LEFT)>safe and
       PSDGet(PSD_RIGHT)>safe):
        VWStraight(100,200)
    else:
        VWStraight(-25,50)
        VWWait()
        dir=int(180*(random()-0.5))
        VWTurn(dir,45)
        VWWait()
```

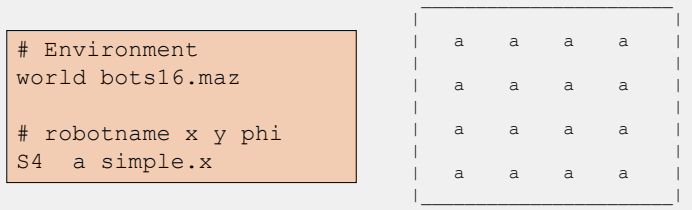
Random Drive in C

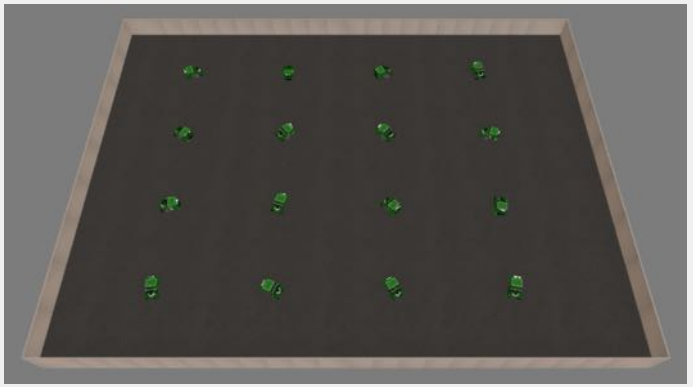
```
#include "eyebot.h"
#define SAFE 300

int main ()
{ BYTE img[QVGA_SIZE];
  int dir, l, f, r;
  LCDMenu("", "", "", "END");
  CAMInit(QVGA);
  while(KEYRead() != KEY4)
  { CAMGet(img);    // demo
    LCDImage(img);  // only
    l = PSDGet(PSD_LEFT);
    f = PSDGet(PSD_FRONT);
    r = PSDGet(PSD_RIGHT);
    LCDSetPrintf(18,0, "PSD L%3d F%3d R%3d", l, f, r);
    if (l>SAFE && f>SAFE && r>SAFE)
      VWStraight(100, 200); // start driving 100mm at 200mm/s
    else
    { VWStraight(-25, 50); VWWait(); // back up
      dir = 180 * ((float)rand()/RAND_MAX-0.5);
      LCDSetPrintf(19,0, "Turn %d", dir);
      VWTurn(dir, 45); VWWait(); // turn [-90..+90]
      LCDSetPrintf(19,0, "          ");
    }
    OSWait(100);
  } // while
}
```

2. Robot Swarms

Robot Swarm Setup

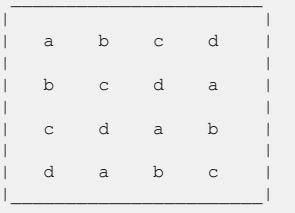
```
# Environment
world bots16.maz

# robotname x y phi
S4   a simple.x
```

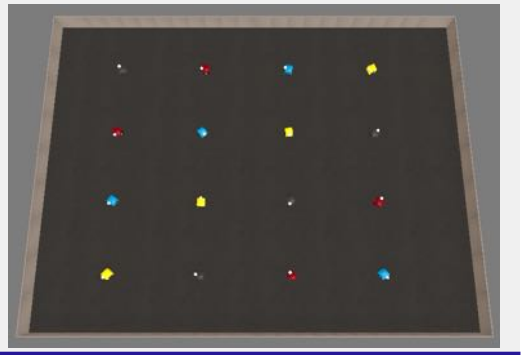




Robot Swarm Setup

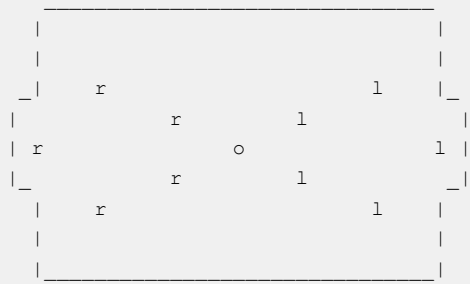
```
# Environment
world bots4x4.maz

# robot definitions
robot robots/Cubot.robi
robot robots/Cubot-r.robi
robot robots/Cubot-b.robi
robot robots/Cubot-y.robi

# robotname placeholder exec.
Cubot    a  simple.x
Cubot-r  b  simple.x
Cubot-b  c  simple.x
Cubot-y  d  simple.x
```



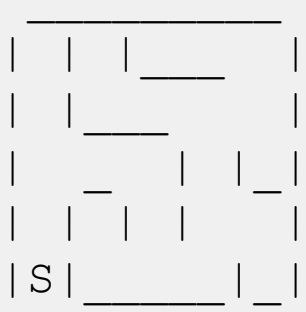


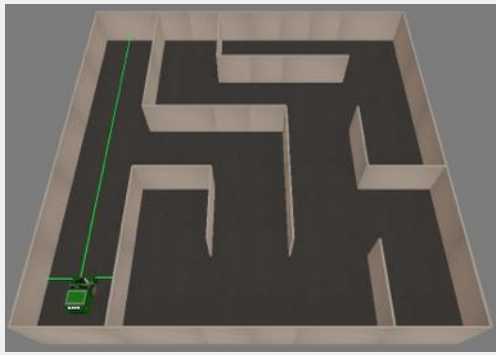
Five-a-side Soccer

```
# Environment
world soccer5-5.maz
# robotname x y phi
S4      l 180 swarm.x
Labbot r   0 swarm.x
```

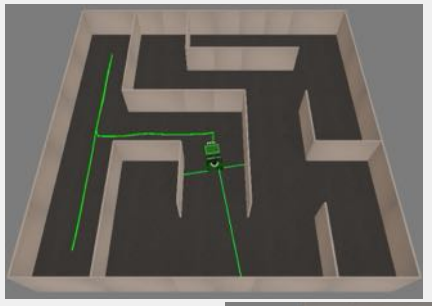


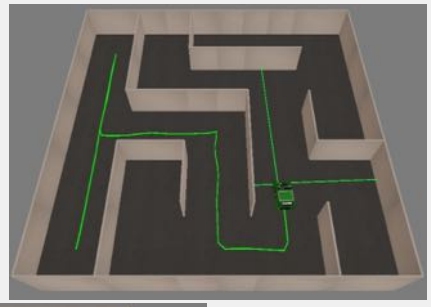
Micromouse Mazes

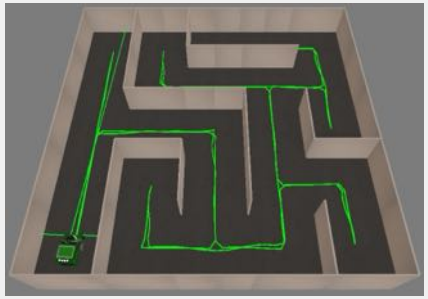




```
# Environment
world small.maz

# Robot description file using "S" in maze as start position
S4 S 90 maze_left.x
```





Driving Multiple Robots

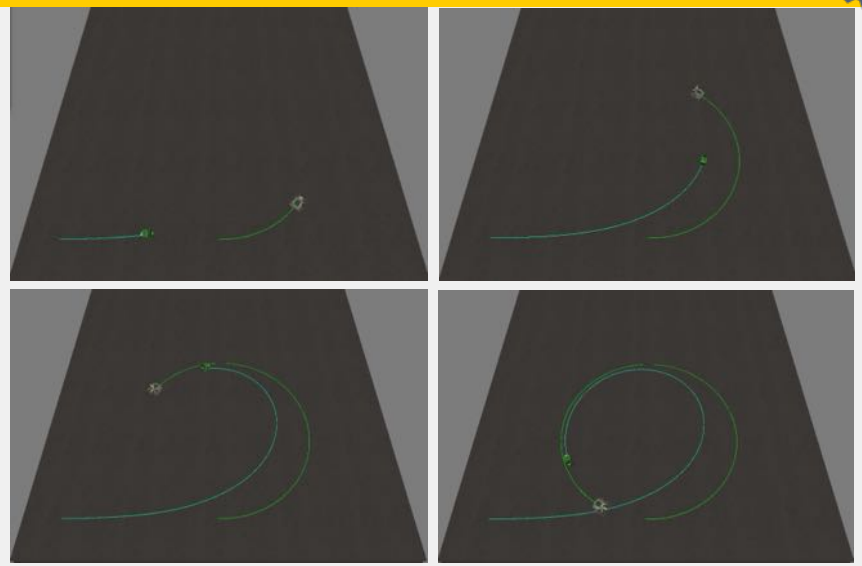
```
# Environment
world $HOME/worlds/small/Soccer1998.wld

settings VIS TRACE

# robotname x y phi
LabBot   400  400   0 randomdrive.py
S4       700  700  45 randomdrive.py
LabBot  1000 1000  90 randomdrive.py
```

Follow Me

```
# Environment
world Field.wld

# robots
Labbot  2000 500 0  leader.py
S4       500 500 0  follower.x
```

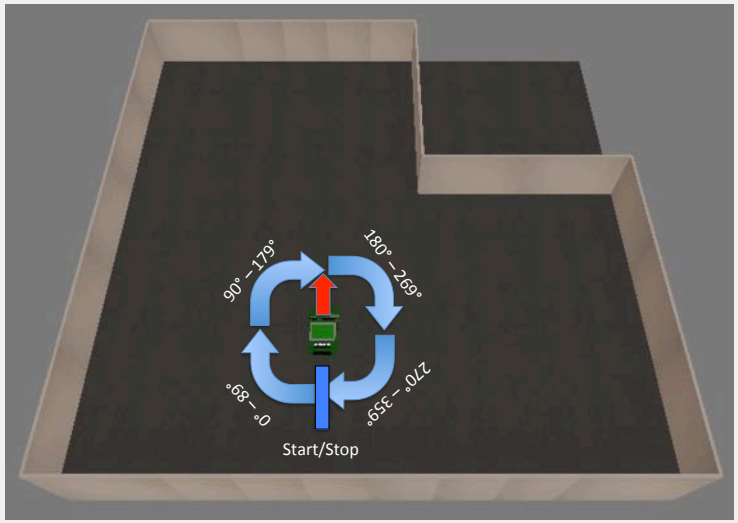
```
from eye import *
VWSetSpeed(300, 15)
```

```
#include "eyebot.h"

int main ()
{ int i, min_pos, scan[360];

  while (KEYRead()!=KEY4)
  { LCDClear();
    LCDMenu("", "", "", "END");
    LIDARGet(scan);
    min_pos = 0;
    for (i=0; i<360; i++)
      if (scan[i] < scan[min_pos]) min_pos = i;
    VWSetSpeed(300, 180-min_pos);
    OSWait(100); // 0.1 sec
  }
}
```



3. Lidar

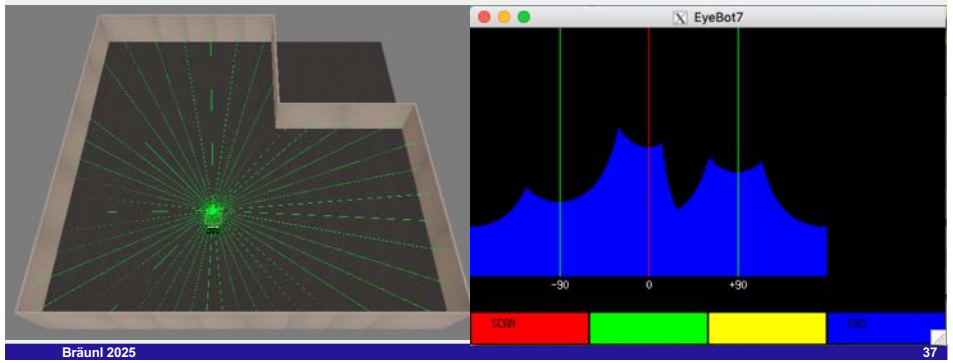
Lidar Scanner

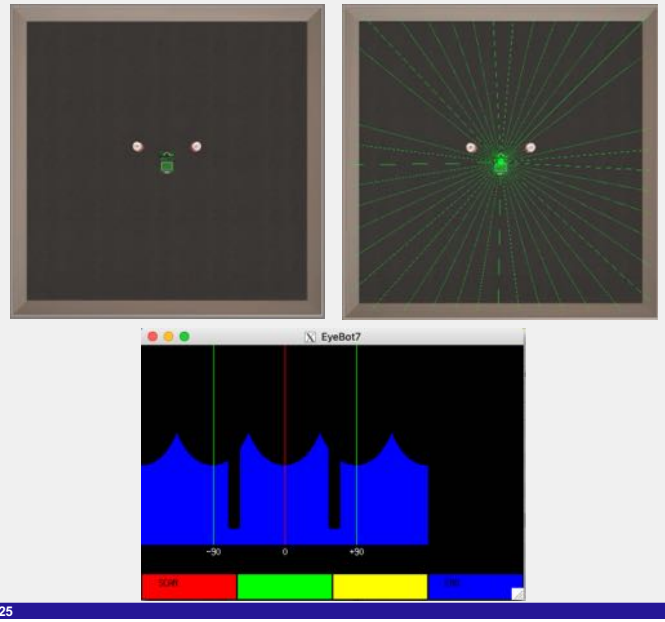


Read Lidar in C and Python

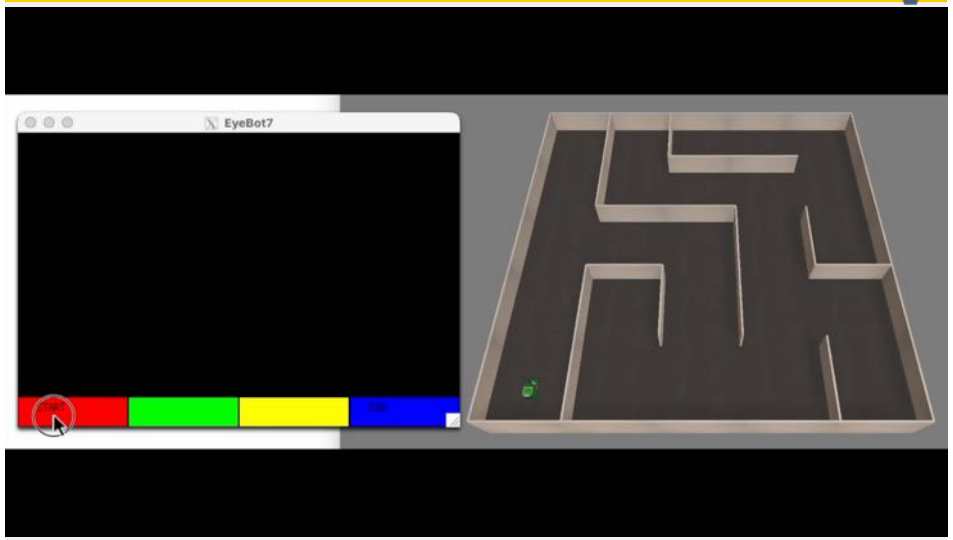
```
#include "eyebot.h"
int main ()
{ int i, scan[360];
  do
  { LCDClear();
    LCDMenu("SCAN", "", "", "END");
    LIDARGet(scan);
    for (i=0; i<360; i++)
      LCDLine(i,250-scan[i]/10, i,250, BLUE);
  } while (KEYGet() != KEY4);
}
```

```
from eye import *
LCDMenu("SCAN", "", "", "END")
while KEYGet() != KEY4:
  LCDClear()
  LCDMenu("SCAN", "", "", "END")
  scan = LIDARGet()
  for i in range(90,270):
    LCDLine(i,250-int(scan[i]/10),
            i,250, BLUE)
```

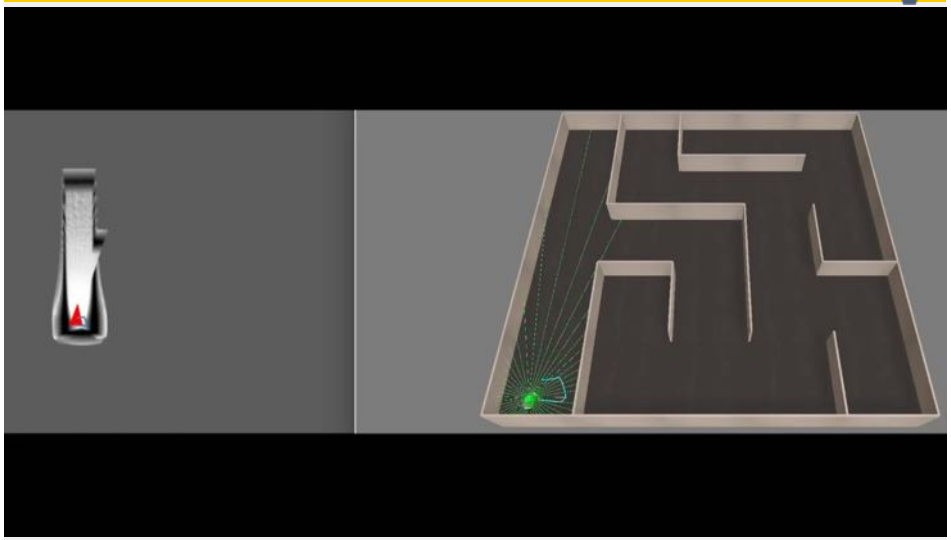




Maze without SLAM



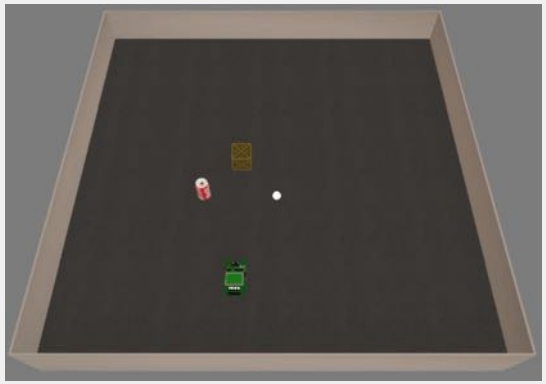
Maze with SLAM



4. Camera and Edges

Robot Vision

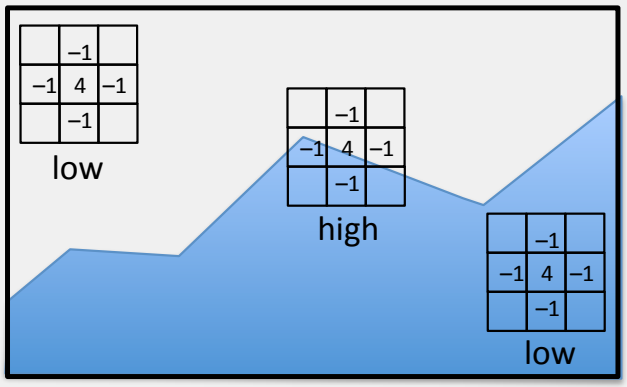
Camera and Screen Functions

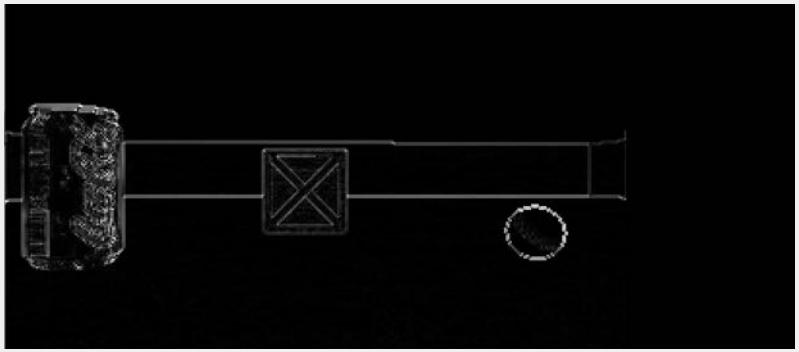



```
from eye import *

CAMInit(QVGA)
while True:
  img = CAMGet()
  LCDImage(img)
```

```
#include "eyebot.h"

int main()
{ BYTE img[QVGA_SIZE];

  CAMInit(QVGA);
  while (1)
  { CAMGet(img);
    LCDImage(img);
  }
}
```


Edge Detection



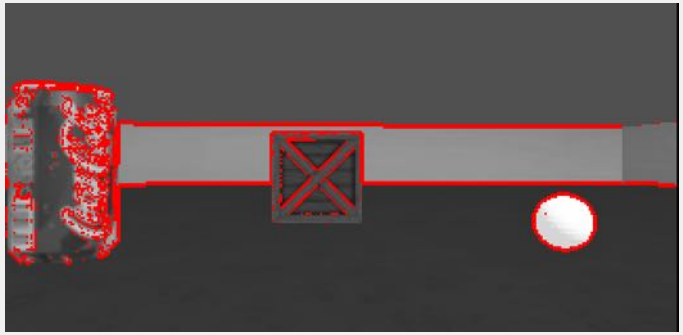


Laplace Edge Detection Programs

```
from eye import *

CAMInit(QVGA)
while True:
  gray = CAMGetGray()
  edge = IPLaplace(gray)
  LCDImageGray(edge)
```

```
#include "eyebot.h"

int main()
{ BYTE img[QVGA_PIXELS], edge[QVGA_PIXELS];

  CAMInit(QVGA);
  while (1)
  { CAMGetGray(img);
    IPLaplace(img, edge);
    LCDImageGray(edge);
  }
}
```


Edge Overlay

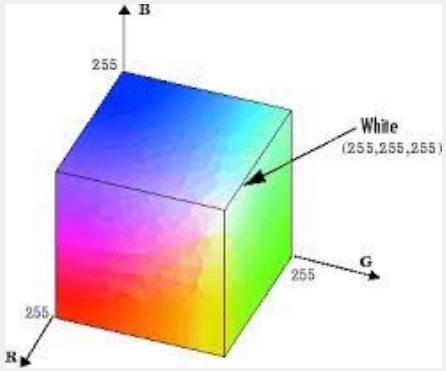


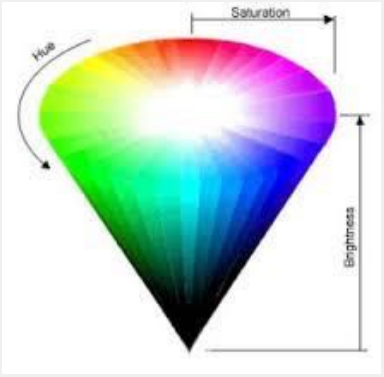
RoBIOS Image Processing Library

```
int   IPSetSize(int resolution);                        // Set IP resolution
int   IPReadFile(char *filename, BYTE* img);            // Read PNM file
int   IPWriteFile(char *filename, BYTE* img);           // Write color file
int   IPWriteFileGray(char *filename, BYTE* gray);      // Write gray file
void  IPLaplace(BYTE* grayIn, BYTE* grayOut);           // Laplace edges
void  IPSobel(BYTE* grayIn, BYTE* grayOut);             // Sobel edge
void  IPCol2Gray(BYTE* imgIn, BYTE* grayOut);           // color to gray
void  IPGray2Col(BYTE* imgIn, BYTE* colOut);            // gray to color
void  IPRGB2Col (BYTE* r, BYTE* g, BYTE* b, BYTE* imgOut); // 3*gray to col
void  IPCol2HSI (BYTE* img, BYTE* h, BYTE* s, BYTE* i);    // RGB to HSI
void  IPOverlay(BYTE* c1, BYTE* c2, BYTE* cOut);           // Overlay col
void  IPOverlayGray(BYTE* g1, BYTE* g2, COLOR col, BYTE* cOut); // Ov. gray
COLOR IPPRGB2Col(BYTE r, BYTE g, BYTE b);                  // RGB to color
void  IPPCol2RGB(COLOR col, BYTE* r, BYTE* g, BYTE* b);    // color to RGB
void  IPPCol2HSI(COLOR c, BYTE* h, BYTE* s, BYTE* i);      // RGB to HSI
BYTE  IPPRGB2Hue(BYTE r, BYTE g, BYTE b);                  // RGB to hue
void  IPPRGB2HSI(BYTE r, BYTE g, BYTE b, BYTE* h, BYTE* s, BYTE* i); // hue
```


5. Object Tracking by Color

Color Models

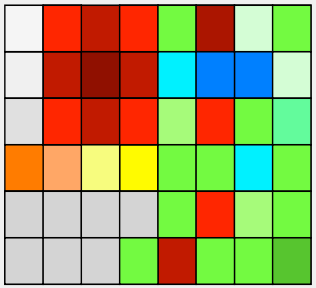




Color spaces **RGB** vs. **HSI**
red, green, blue hue, saturation, intensity

Images: MSU.edu, tomjewett.com

RGB → Hue Conversion



90	60	64	62	30	54	32	33
100	60	59	61	82	78	80	35
110	58	60	61	31	64	29	30
99	80	99	80	30	33	82	31
255	255	255	255	31	60	28	29
255	255	255	30	63	30	33	32

Matching Pixels

0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0

Histograms

1. Do Sums
2. Find Max

0	1	1	1	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0

3
3
4
0
1
1

0	3	3	3	1	2	0	0

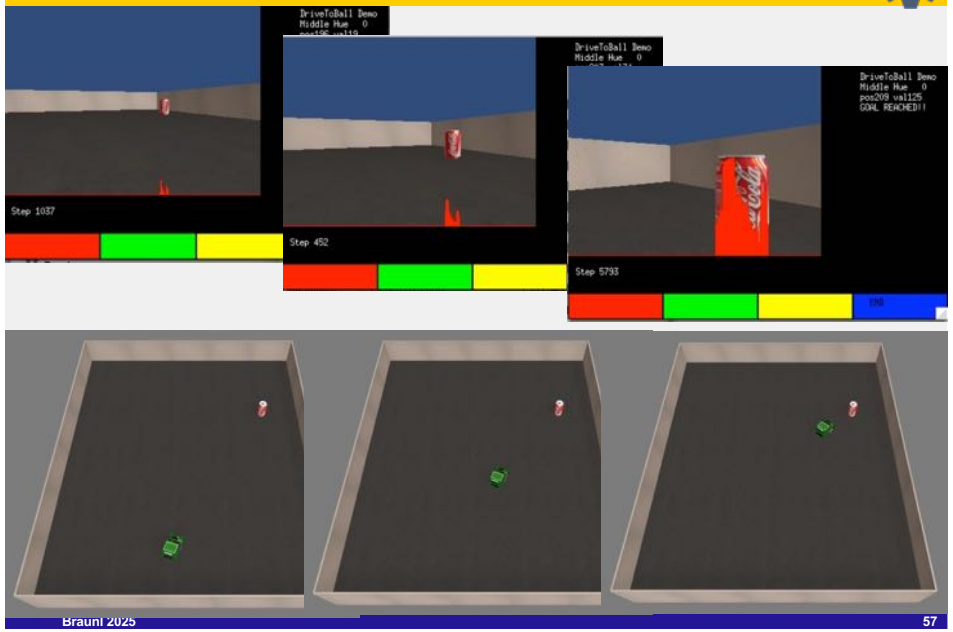
```
void GenHistogram(VGAcol img, int hue, line hist, int thres)
{ int x,y, pos, diff;
  for (x=0; x<CAMWIDTH; x++)
  { hist[x] = 0;
    for (y=0; y<CAMHEIGHT; y++)
    {  pos  = y*CAMWIDTH + x;
       diff = abs(img[pos] - hue);
       if ( ((diff < thres) || (255-diff < thres))
            && (img[pos] != NO_HUE))
         hist[x]++;
    }
  }
}
```



```
int FindMax(line hist)
{ int i, pos=0, val=hist[0]; /* init */
  for (i=1; i<CAMWIDTH; i++)
    if (hist[i] > val)
    { pos = i;
      val = hist[i];
    }
  return pos;
}
```



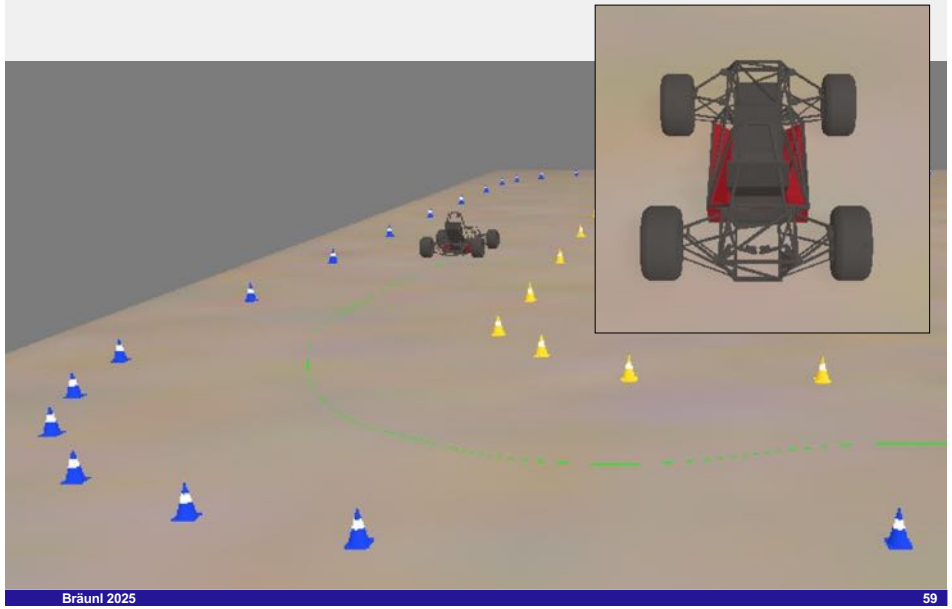
```
if (pos < CAMWIDTH/3) VWTurn(10, 30);                 // left
      else { if (pos > 2*CAMWIDTH/3) VWTurn(-10, 30); // right
             else VWStraight( 50, 100);               // straight
           }
```

Outlook Autonomous Driving

Carolo Cup and Audi Autonomous Driving Cup

Outlook Formula-SAE



6. Submarines

Autonomous Boats and Submarines

Water SIM + WORLD Files

```
# Environment
world worlds/pool.wld
# Load custom robot
robot robots/mako.robi
# Robot pose (x, y, phi), exec.
Mako 7000 4000 0 mako-drive.x
```

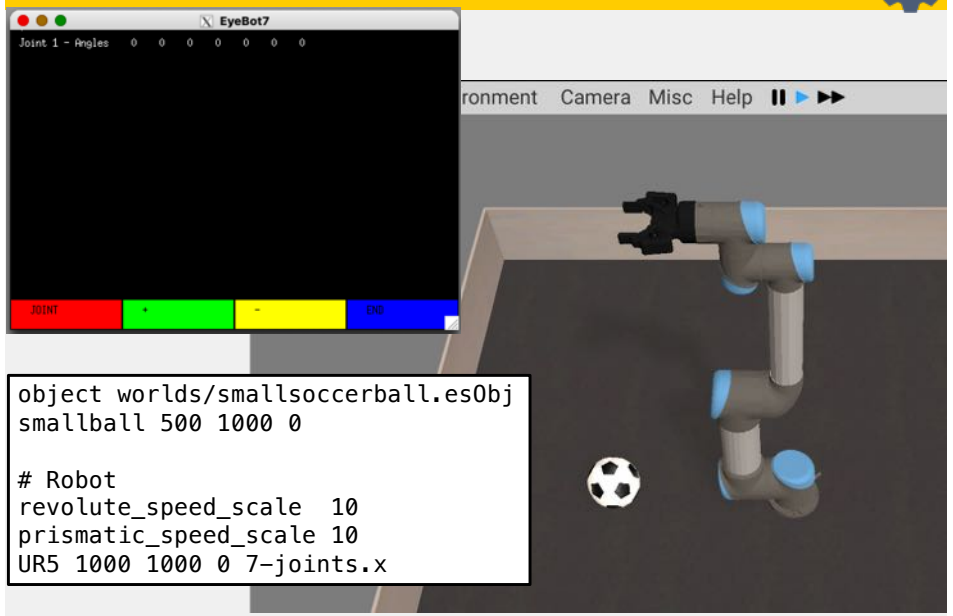
```
floor_texture rough-blue.jpg
terrain 14000 8000 2000
        olympic-pool.png
water_level 1900
```

Water Program

```
// PSD IDs.
const int PSD_DOWN = 6;

// Thruster IDs.
const int LEFT  = 1;
const int FRONT = 2;
const int RIGHT = 3;
const int BACK  = 4;
…
void down()
{   LCDSetPrintf(0,0,"DOWN       ");
    MOTORDrive(FRONT, -SPEED);
    MOTORDrive(BACK, -SPEED);
}

void forward()
{   LCDSetPrintf(0,0,"FORWARD ");
    MOTORDrive(LEFT, SPEED);
    MOTORDrive(RIGHT, SPEED);
}
```

7. Manipulators

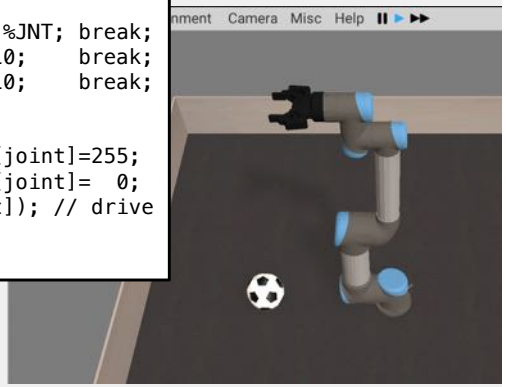
UR5 Simulation




```
object worlds/smallsoccerball.esObj
smallball 500 1000 0

# Robot
revolute_speed_scale  10
prismatic_speed_scale 10
UR5 1000 1000 0 7-joints.x
```

UR5 Program

```
#include "eyebot.h"
#define JNT 6

int main ()
{ int angles[JNT] = {0,0,0,0,0,0};
  int joint = 0, done  = false;

  LCDMenu("JOINT","+","-","END");
  while (!done)
  { switch (KEYGet())
    { case KEY1: joint = (joint+1)%JNT; break;
      case KEY2: angles[joint] +=10;    break;
      case KEY3: angles[joint] -=10;    break;
      case KEY4: done = true;
    }
    if (angles[joint]>255) angles[joint]=255;
    if (angles[joint]<  0) angles[joint]=  0;
    SERVOSet(joint+1, angles[joint]); // drive
  }
}
```



Thanks for joining!

Thomas Bräunl, UWA
http://RobLab.org
http://REVproject.com
Thomas.Braunl@UWA.edu.au