

EyeSim Mobile Robot Simulation System

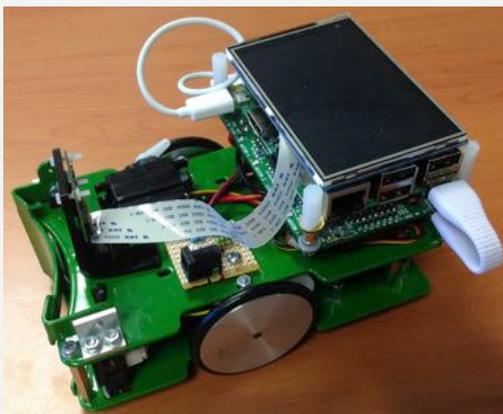


Professor Thomas Bräunl
The University of Western Australia

EyeSim Simulation System



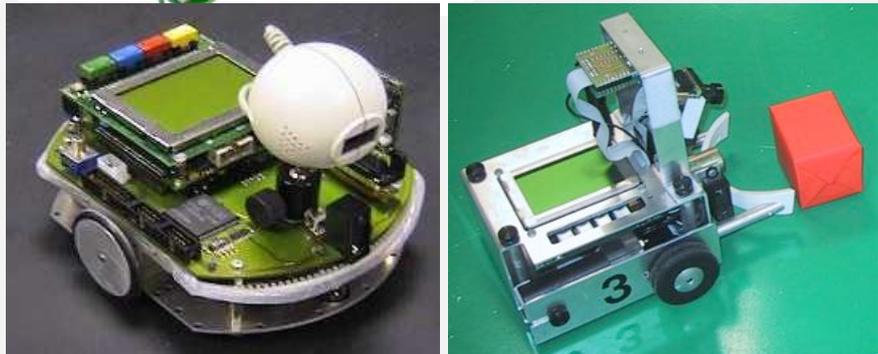
- Mobile robot research without real robots
- Rapid prototyping:
Being able to test a design before it has been manufactured
- Algorithm optimization
- Educational purposes



EyeBot Generations



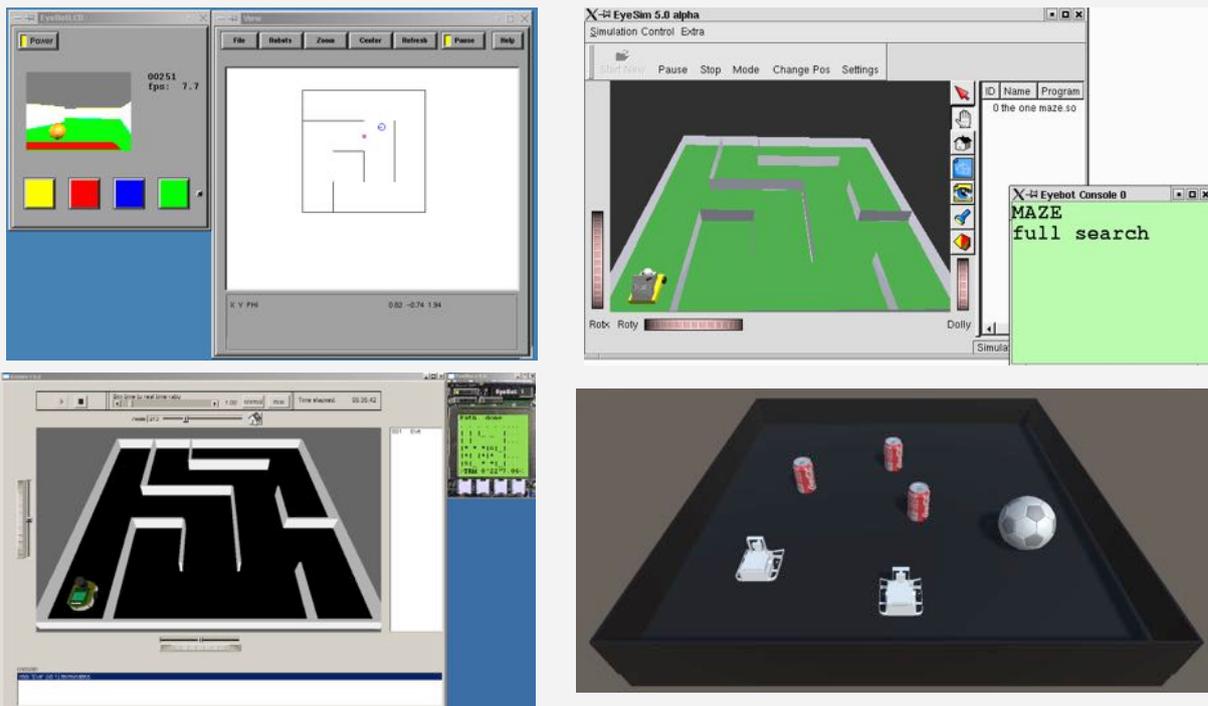
Designs:
 Jörg Henne, Frank Sautter,
 Joshua Pettit, Richard Meager,
 Thomas Bräunl



Bräunl 2018

3

EyeSim Generations



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

Bräunl 2018

4

EyeBot-7 – RoBIOS API



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 *linSpeed, 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 VWDriveRemain(void); // Return remaining drive distance in [mm]
int VWDriveDone(void); // Non-blocking check whether drive is finished (1) or not
int VWDriveWait(void); // Suspend current thread until drive operation has finished
int VWStalled(void); // Returns number of stalled motor [1..2], 3 if both stalled
```

<http://robotics.ee.uwa.edu.au/eyebot7/Robios7.html>

Random Drive V7



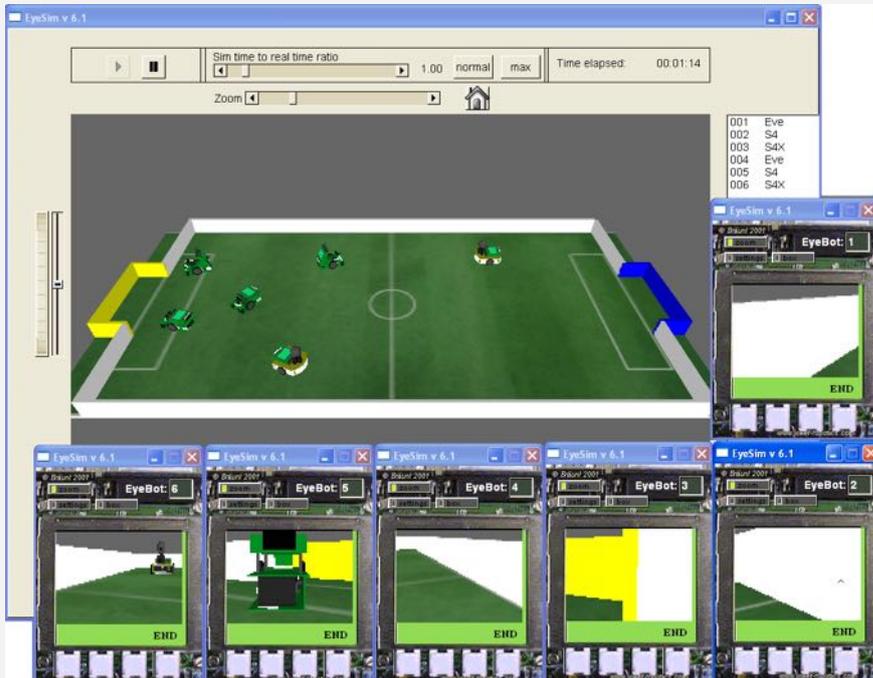
```
#include "eyebot.h"
#define SAFE 300

int main ()
{ int dir, l, f, r;

  while(KEYRead() != KEY4)
  { l = PSDGet(1); // left
    f = PSDGet(2); // front
    r = PSDGet(3); // right

    if (l>SAFE && f>SAFE && r>SAFE) VWStraight( 100, 200);
    else
    { VWStraight(-25, 50); // back up
      VWWait();
      dir = (((float)rand()/((float)RAND_MAX - 0.5))*180);
      VWTurn(dir, 45); // turn random angle
      VWWait();
    }
    OSWait(100);
  } // while
  return 0;
}
```

EyeSim



- Bräunl, Koestler, 2003
- Bräunl, Povey, Frewin, 2017
- Multi-robot simulation
- Sensor modeling incl. camera for each robot
- EyeBot/RoBIOS compatible

Bräunl 2018

7

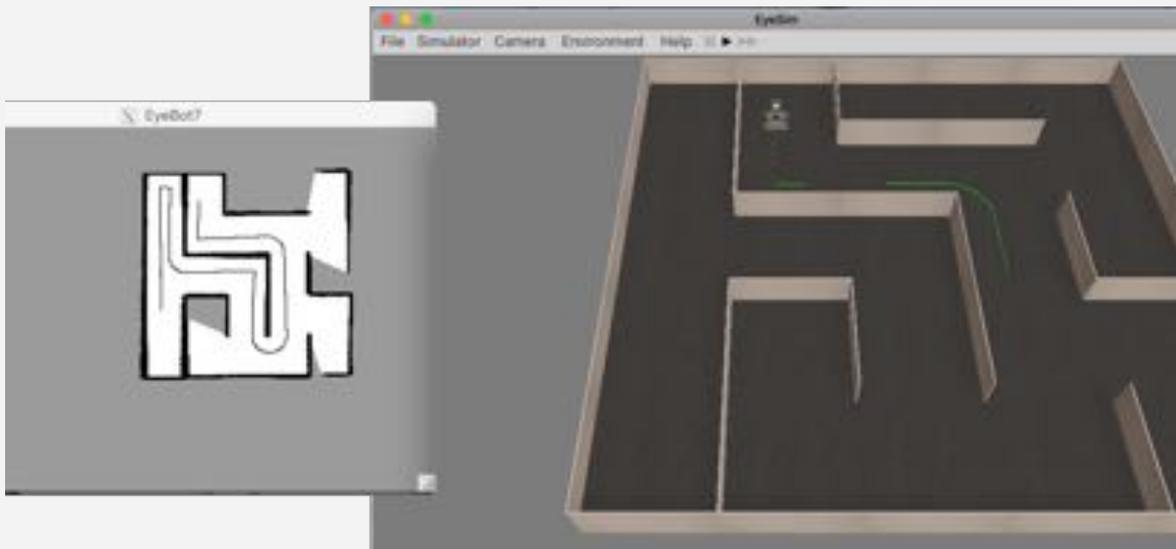
EyeSim



Bräunl 2018

8

EyeSim



Simulated Lidar (rotating laser scanner)
Maze Exploration

EyeSim-VR



System extension to work with VR Systems:

- Oculus Rift
- HTC Vive



Experience

- Viewer can see robot's interacting up close
- Viewer can interact inside the scene (seen as an obstacle by the robots)
- Viewer can ride on one of the robots

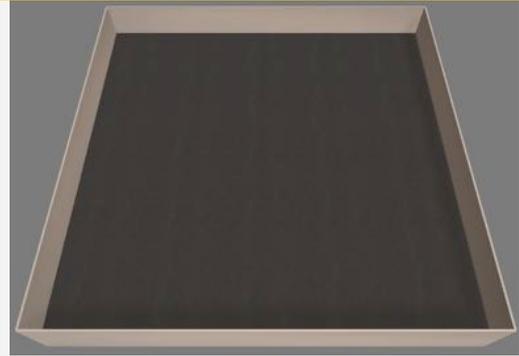


EyeSim Start



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`

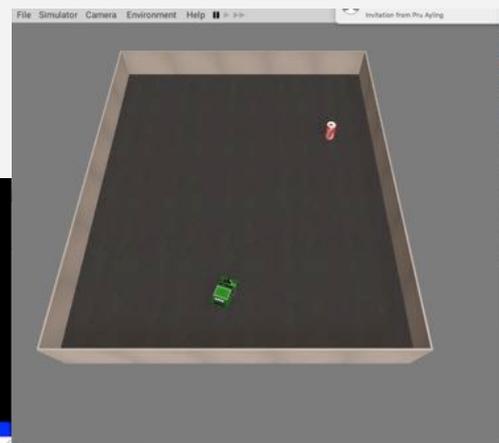
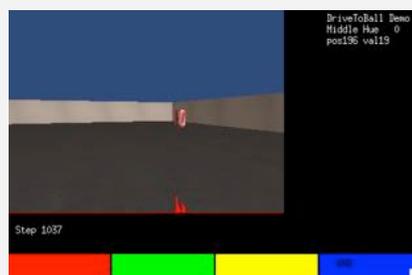
EyeSim SIM File



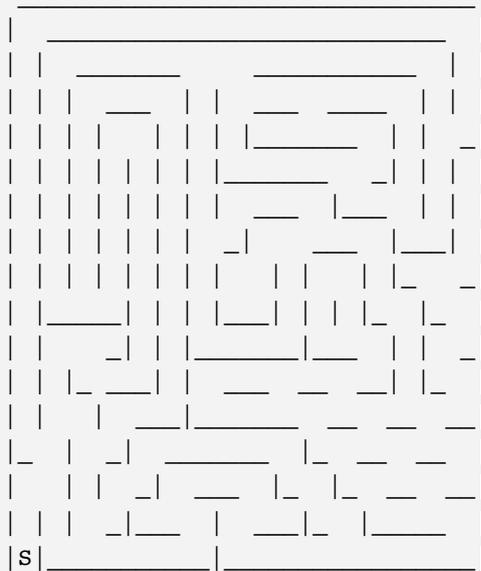
```
# World description file (maze or world)
world box.maz
```

```
# Objects placed in driving area
can 1500 1500 0
```

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

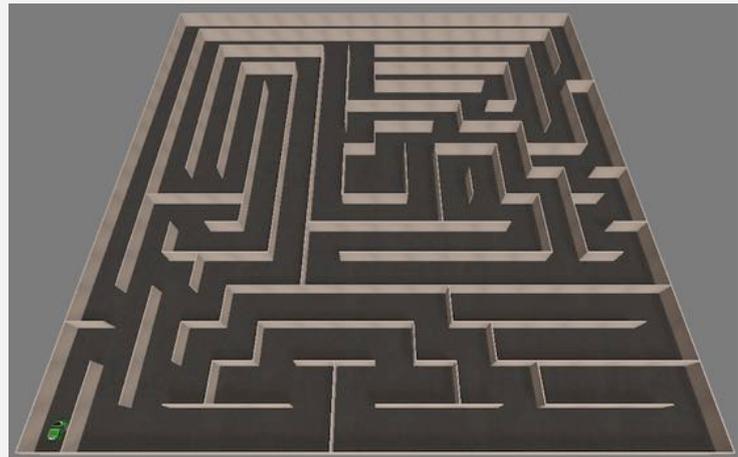


EyeSim MAZE File



Character Graphics

- Starting pose S,s (U,u, D,d, L, l, R,r) with wall below for capital characters
- Optional size parameter



```
# Environment
world london-1986.maz
```

```
# Robot description file using "S"
S4 S maze.x
```

EyeSim WORLD File



```
floor_texture ../textures/soccer.png
width 3100
height 1530
```

```
; wall-start (x,y), wall-end (x,y), thickness, height, color (R,G,B)
180 0 180 515 18 100 255 255 255
180 1015 180 1530 18 100 255 255 255
0 497 198 497 18 100 255 255 0
0 1015 198 1015 18 100 255 255 0
0 515 0 1015 18 100 255 255 0
180 0 2938 0 18 100 255 255 255
180 1530 2938 1530 18 100 255 255 255
2938 0 2938 533 18 100 255 255 255
2938 1015 2938 1530 18 100 255 255 255
2938 1015 3100 1015 18 100 0 0 255
2938 515 3100 515 18 100 0 0 255
3082 515 3082 1015 18 100 0 0 255
```



EyeSim Robots



SIM File

```
# robi description file
Acker 600 600 0 acker-drive.x

# robi description file
Omni 600 600 0 omni-drive.x

# robi description file from robi file
robi ../../robots/S5/S5.robi
S5      600 600 0 acker-drive.x
```

ROBI File

```
name S4

# robot diameter in mm
diameter 186
# max linear velocity in mm/s
speed    600
# max rotational velocity in deg/s
turn     300
# Graphics file name
model    S5.obj
...
```

OBJECT File (generated)

```
# File produced by Open Asset Import Library
(http://www.assimp.sf.net)
# (assimp v3.0.1262)

mtllib S4.ms3d.obj.mtl

# 24024 vertex positions
v 0.0719748 0.0433378 0.0479923
v 0.0719748 0.0431497 0.0479923
v 0.0719748 0.0433378 0.0360212
v 0.0719748 0.0418327 0.0479923
...
```

EyeSim ROBI File



```
# the name of the robi, the used string is the rest of the line
# behind the keyword without the surrounding white space
name S4

# robot diameter in mm
diameter 186

# max linear velocity in mm/s
speed    600

# max rotational velocity in deg/s
turn     300

# the file name of the OpenInventor model used for this robi
model    S5.obj

# axis is the distance between the center of the robi and the center of
# the robis axis, e.g. a value of 0 means the axis is in the center of
# the robi, if no value is given, it is set to 0.
#axis 00 60

# psd sensor definition: (psd names are followed by id-number from "hdt_sem.h")
# "psd", name, id, relative position to robi center(x,y,z) in mm, angle in x-y plane in deg
psd PSD_FRONT 1 60 20 30 0
psd PSD_LEFT 2 56 45 30 90
psd PSD_RIGHT 3 56 -45 30 -90
```

EyeSim ROBI File



...

```
# color camera sensor definition:
# "camera", relative position to the robi center (x,y,z), default
# pan-tilt-angle (pan, tilt), maximum image width and height in pixel
camera 62 0 60 0 0 482 162

# wheel diameter [mm], max. rotational velocity [deg/s], encoder ticks/rev., wheel-base
distance [mm]
wheel 54 3600 1100 90

# motors and encoders for low level drive routines
# Diff.-drive: left motor, left encoder, right motor, right encoder
drive DIFFERENTIAL_DRIVE MOTOR_LEFT QUAD_LEFT MOTOR_RIGHT QUAD_RIGHT
```

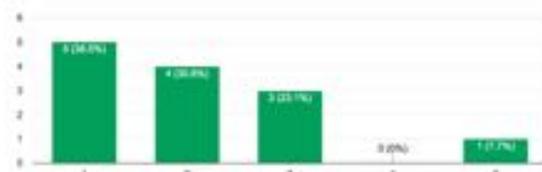
User/Student Feedback



Survey - Results

When you first used EyeSim, how easy did you find creating a scene with robots and environments?

13 responses



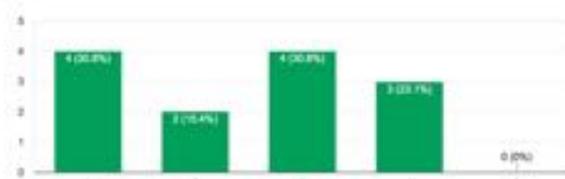
How difficult did you find compiling and running an EyeSim program?

13 responses



How easy did you find it to change the viewpoint (scale, pan, rotate)?

13 responses



How would you rate the ability to program and test a robot using the simulation system outside of lab hours and unrestricted in time.

13 responses



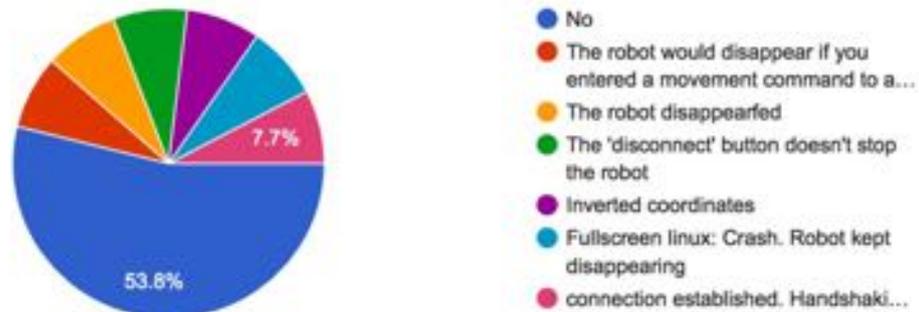
User/Student Feedback



Survey - Feedback

Were there any times while using EyeSim when you tried to perform a certain action and nothing happened? If Yes, explain it in 'Other'

13 responses



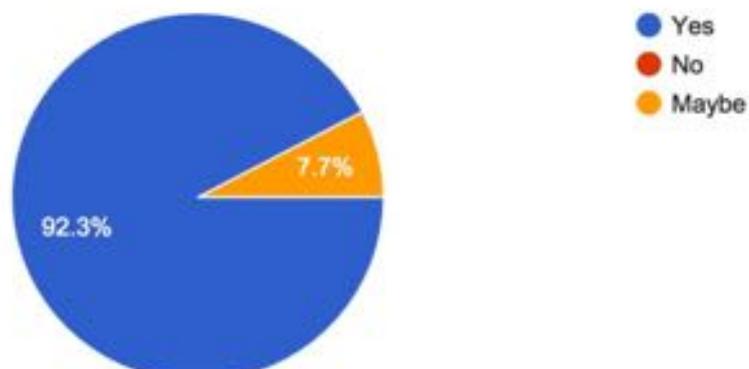
User/Student Feedback



Survey - Result

Did you see a benefit in using EyeSim for understanding robotics?

13 responses



EyeSim

