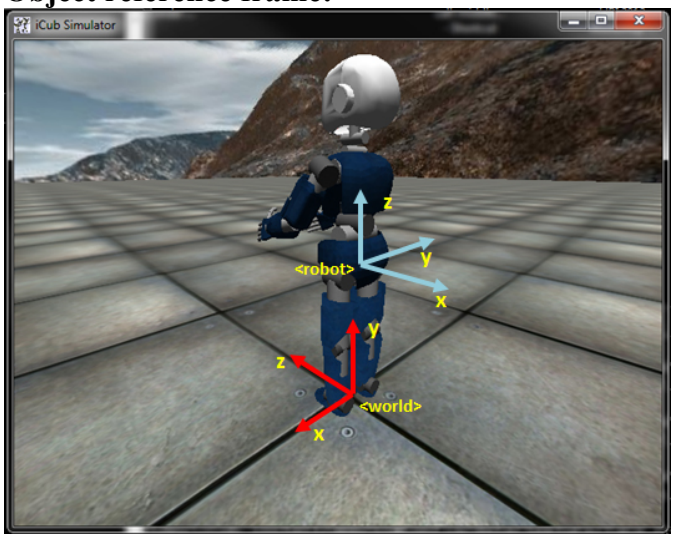
SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

The iCub Simulator and objects

2. Object in the iCub Simulator:

There are currently different objects in the world **ball, cube, and box**.

Object reference frame:



Objects are placed in the <world> reference frame, placed on the floor, between the legs of the robot as in the picture (y axis pointing above, x towards the left of the robot and z towards the front). Notice that the robot kinematics (Cartesian interface) uses the <robot> reference frame.
The rototranslational matrix describing the homogeneous transformation from the <robot> reference frame to the <world> reference frame of the simulator is:
T = [0 -1 0 0; 0 0 1 0.5976; -1 0 0 -0.026; 0 0 0 1] (given in a matlab-like format, ordered by rows)
Note that this rototranslation applies as long as the 'elevation' flag in the simulator configuration file is set to off. Otherwise, the robot is lifted, changing the position of the root reference frame w.r.t. the world reference frame.

Exercise 2.1 Create objects in the iCub Simulator

It is possible to create **boxes, spheres and cylinders** in the world. It is also possible to specify if the actual object can collide with the robot. This is an extra boolean parameter while creating the object. By default the collision is on and can be removed only if specified by the user.

Objects can be created in the world port:

```
yarp rpc /icubSim/world
```

The format on how to create an object is structured as follows:

world mk box (three params for size) (three params for pos) (three params for colour) ---- This creates a box affected by gravity
world mk sph (radius)(three params for pos) (three params for colour) ------- This creates a sphere affected by gravity
world mk cyl (radius length) (three params for pos) (three params for colour) ------- This creates a cylinder affected by gravity
world mk sbox (three params for size) (three params for pos) (three params for colour) ----- This creates a static box
world mk ssph (radius) (three params for pos) (three params for colour) ------- This creates a static sphere
world mk scyl (radius length) (three params for pos) (three params for colour) ------- This creates a static cylinder

```
world mk box 0.03 0.03 0.03 0.3 0.2 1 1 0 0
world mk sph 0.04 0.0 1.0 0.5 1 0 1
world mk cyl 0.1 0.2 0.0 0.9 1.0 0 0 1
```

OPTIONAL: removing collision with the robot - by default the flag is set to "true".
eg:

```
world mk box 0.03 0.03 0.03 0.3 0.2 1 1 0 0 FALSE
```

The rpc port is a special port:

RPC stands for "Remote Procedure Call", and in YARP is used to refer to communication that consists of a message and a reply to that message. Historically, YARP has stressed streaming communication (without replies) because in our experience it leads to more robust and scalable control logic. However, RPC definitely has its place and the world port in the simulator is on of this RPC ports.

The benefit of this port is that we can manipulate objects after the creation, e.g. get and set their position after they have been created:

Exercise 2.2 Get/set object position:

To get and set positions for these newly created objects:

world get box (num) or world set box (num) x y z

eg:

```
world get box 1    or    world set box 1 2 2 2
world get sph 1    or    world set sph 1 2 2 2
world get cyl 1    or    world set cyl 1 2 2 2
```

Exercise 2.3 Get/Set object rotation:

If you need to rotate the boxes or the cylinders just use the following function:

world rot (object)(num) rotx roty rotz

(where: rotx = rotation in degrees on the x axis roty = rotation in degrees on the y axis rotz = rotation in degrees on the z axis)

To query the rotation of an object, omit the angles:

world rot (object)(num)

```
world rot box 1    or    world rot box 1 2 2 2
world rot sph 1    or    world rot sph 1 2 2 2
world rot cyl 1    or    world rot cyl 1 2 2 2
```

Exercise 2.4 Change the color of objects

If you need to change the color of the created objects just use the following function:

world col (object)(num) R G B

```
world col box 1 2 2 2
```

Exercise 2.5 Deleting objects in the simulator:

```
world del all
```

(this will delete all objects in the world)

SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

The iCub Simulator setup

Exercise 2.6 Setup/activate parts in the iCub Simulator:

Download from vision Lab2_src into your local account.
Unzip the folder, start a Cmd.
Change into the directory and start the iCub_sim from there:

```
cd Lab2_src\InitialisationiCubSimulator\
iCub_sim
```

The file inside the folder you just downloaded looks like this:

```
/// initialization file for the icub simulation parts
[SETUP]
elevation off
startHomePos on

[PARTS]
legs on
torso on
left_arm on
left_hand on
right_arm on
right_hand on
head on
fixed_hip on

[COLLISIONS]
self_collisions off
covers_collisions off

[SENSORS]
pressure off
whole_body_skin_emul off

[VISION]
cam on

[RENDER]
objects off
screen off

head_cover on
legs_covers on
torso_covers on
left_arm_covers on
right_arm_covers on
```

by changing the status of one of this variables you can change the simulation, try e.g. to turn off the head_cover.

Exercise 2.7 Show a video on the screen

Make sure your screen is switched on in the iCub_sim you are using.

```
yarpdev --device test_grabber --name /test/video --mode ball
yarp connect /test/video /icubSim/texture/screen
```

Exercise 2.8 Show a webcam feed on the screen

Make sure your screen is switched on in the iCub_sim you are using.

```
yarpdev --device opencv_grabber --name /test/video
yarp connect /test/video /icubSim/texture/screen
```

Read more about the test_grabber here:
http://www.yarp.it/group__grabber__basic.html

Read about how to create your own yarp port in c++:
http://www.yarp.it/note_ports.html

Try to compile the example code by using the yarp cmake command to generate a CMakeList.txt.
Read about C++ compilation:
http://www.yarp.it/using_cmake.html