RACE TO THE WITCH MOUNTAIN

V-REP & ROS Tutorial

Create a new scene

- Click on File
- Click on New Scene
- Save that scene (Ctrl + S)
- Name it whatever you want! (maybe "race_to_the_witch_mountain")

Add Our Model

- Where can you see all the MODELS in ?
    - FTV?
    - Nop.. The Model Browser

- Go to Mobile Models

- Drag and drop dr20.ttm robot onto your scene

Adding Elements to the scene

- Add a Cylinder of 1m Diameter (x) and 1m Hight (z)

- Place it at position
  - X : 1m
  - Y : 1m
  - Z : Don’t Touch it

Adding Elements to the scene

- Add a Cube of 0.5m Side (x,y,z)

- Place it at position

  - X : -1m
  - Y : -1m
  - Z : Don’t Touch it

Adding Elements to the scene

- Add 4 Cylinders of 0.25m Diameter (x) and 0.5m Hight (z)

- Place it at positions
  - (1, 0)
  - (0, 1)
  - (0, -0.5)
  - (-0.5, 0)
  - Z : Don't Touch

Let's LUA...

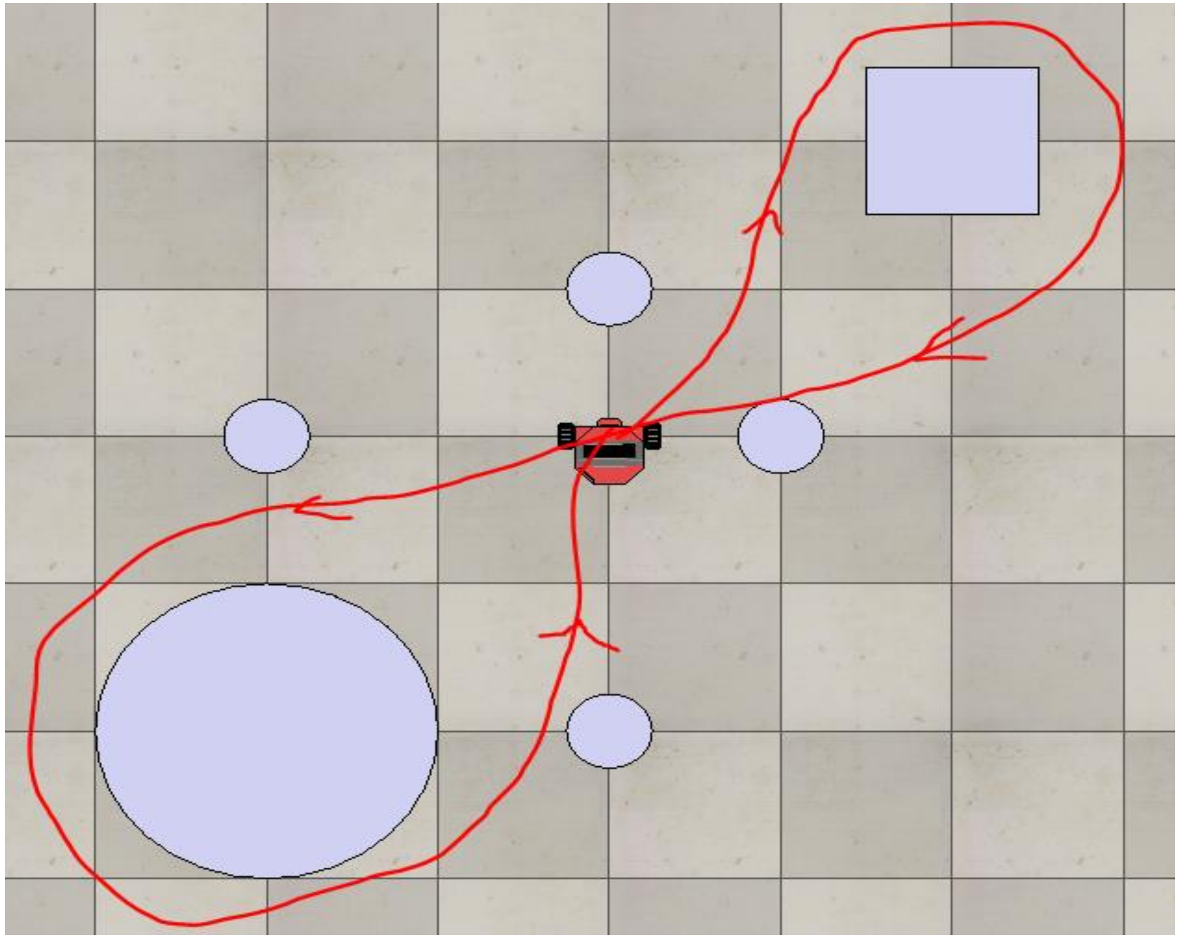
```
function sysCall_init()
    leftJoint=sim.getObjectHandle("dr20_leftWheelJoint_")
    rightJoint=sim.getObjectHandle("dr20_rightWheelJoint_")

    -- ROS Subscription to topic
   /turtle1/cmd_vel     subscriber=simROS.subscribe('/turtle1/cmd_vel','
   geometry_msgs/Twist','subscriber_callback')
end
```

Let's LUA...

```lua
function subscriber_callback(msg)
    -- This is the subscriber callback function
    print(msg)
    velocityFactor = 5
    linearVelocity = msg.linear.x * velocityFactor
    angularVelocity = msg.angular.z * velocityFactor
    leftWheelVelocity = linearVelocity + ((linearVelocity + 1) * (-1 * angularVelocity))
    rightWheelVelocity = linearVelocity + ((linearVelocity + 1) * angularVelocity)
    -- Set the velocities
    sim.setJointTargetVelocity(leftJoint, leftWheelVelocity)
    sim.setJointTargetVelocity(rightJoint, rightWheelVelocity)
end
```

Challenge 1: LET THE RACE BEGIN



- Start the Simulator
- Start the turtlesim teleop ros node....

- ?

... turtlesim turtle_teleop_key

Challenge 2...

- Let's RACE with the robot's eyes!!

Let's (re)LUA....

- Add below lines to sysCall_init function

```
camera=sim.getObjectHandle("Vision_sensor")
pub=simROS.advertise('/d20_image', 'sensor_msgs/Image') -- You created a publisher object
simROS.publisherTreatUInt8ArrayAsString(pub)
```

Let's (re)LUA....

```
function sysCall_init()
    leftJoint=sim.getObjectHandle("dr20_leftWheelJoint_")
    rightJoint=sim.getObjectHandle("dr20_rightWheelJoint_")
    camera=sim.getObjectHandle("Vision_sensor")

    -- ROS Subscription to topic /turtle1/cmd_vel
    subscriber=simROS.subscribe('/turtle1/cmd_vel','geometry_msgs/Twist','subscriber_callback')

    -- ROS Publisher to publish Image
    pub=simROS.advertise('/d20_image', 'sensor_msgs/Image') -- You created a publisher object
    simROS.publisherTreatUInt8ArrayAsString(pub) -- treat uint8 arrays as strings (much faster,
    tables/arrays are kind of slow in Lua)
end
```

```
function sysCall_sensing()
    -- Publish the image of the active vision sensor:
    local data,w,h=sim.getVisionSensorCharImage(camera)
    d={}
    d['header']={seq=0,stamp=simROS.getTime(), frame_id="Robot_Image"}
    d['height']=w
    d['width']=h
    d['encoding']='rgb8'
    d['is_bigendian']=1
    d['step']= w*3
    d['data']=data
    --print(w,h)
    simROS.publish(pub,d)
end
```

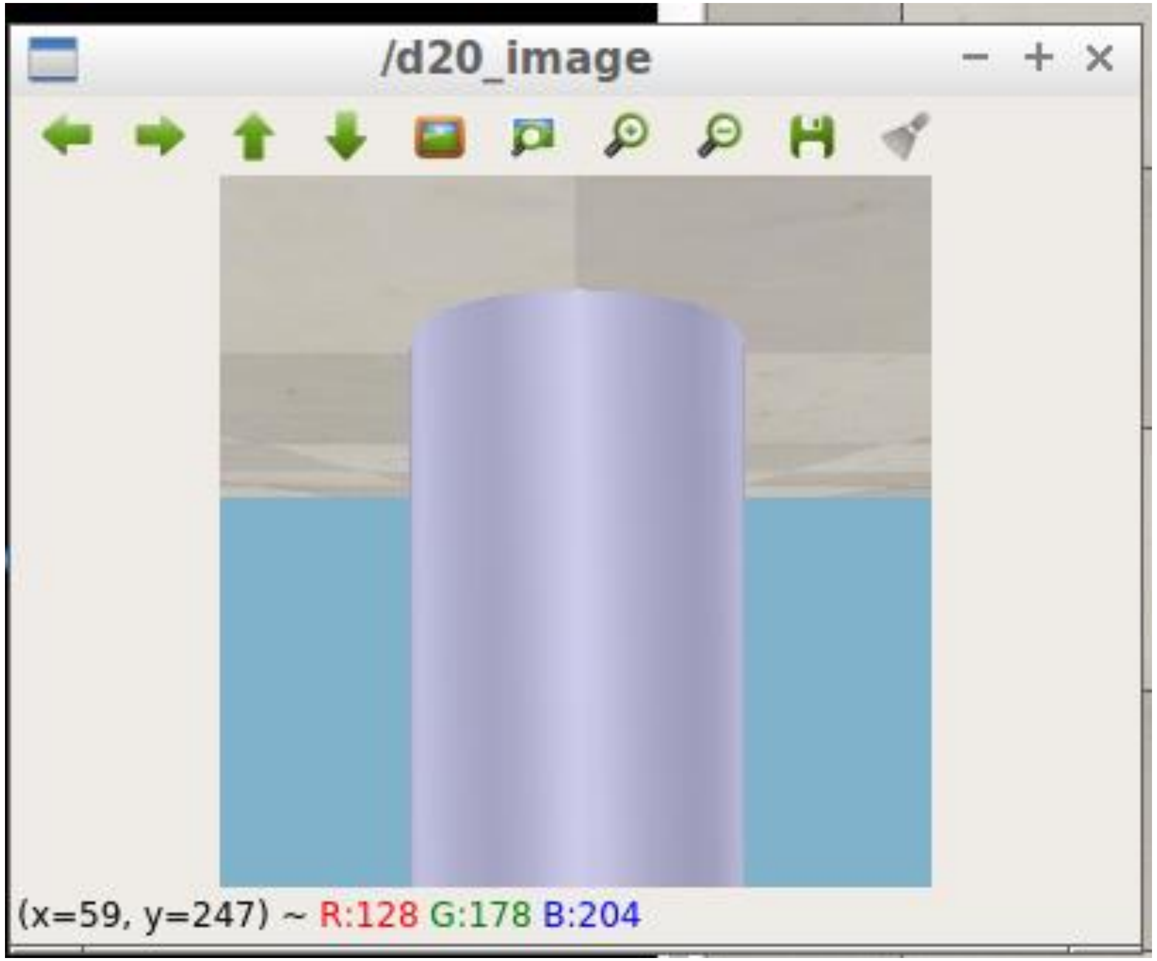
Install this

- sudo apt-get install ros-kinetic-image-view

Challenge 2: Let's begin

- Start the simulator
- Run command:
  - rosrun image_view image_view image:=/d20_image

Challenge 2: Let's begin



(x=59, y=247) ~ R:128 G:178 B:204

OMG the image is inverted !!!!!!

Ok.. Lets fix this in V-REP

Invert the image

- Double click on the Vision_sensor node
- Click on "Show filters dialog" (you are not looking properly :P)
- In Add Filter: choose "Flip work image vertically"
- Use the arrows to place it in the middle

Challenge 2: Let's begin

- Start the simulator

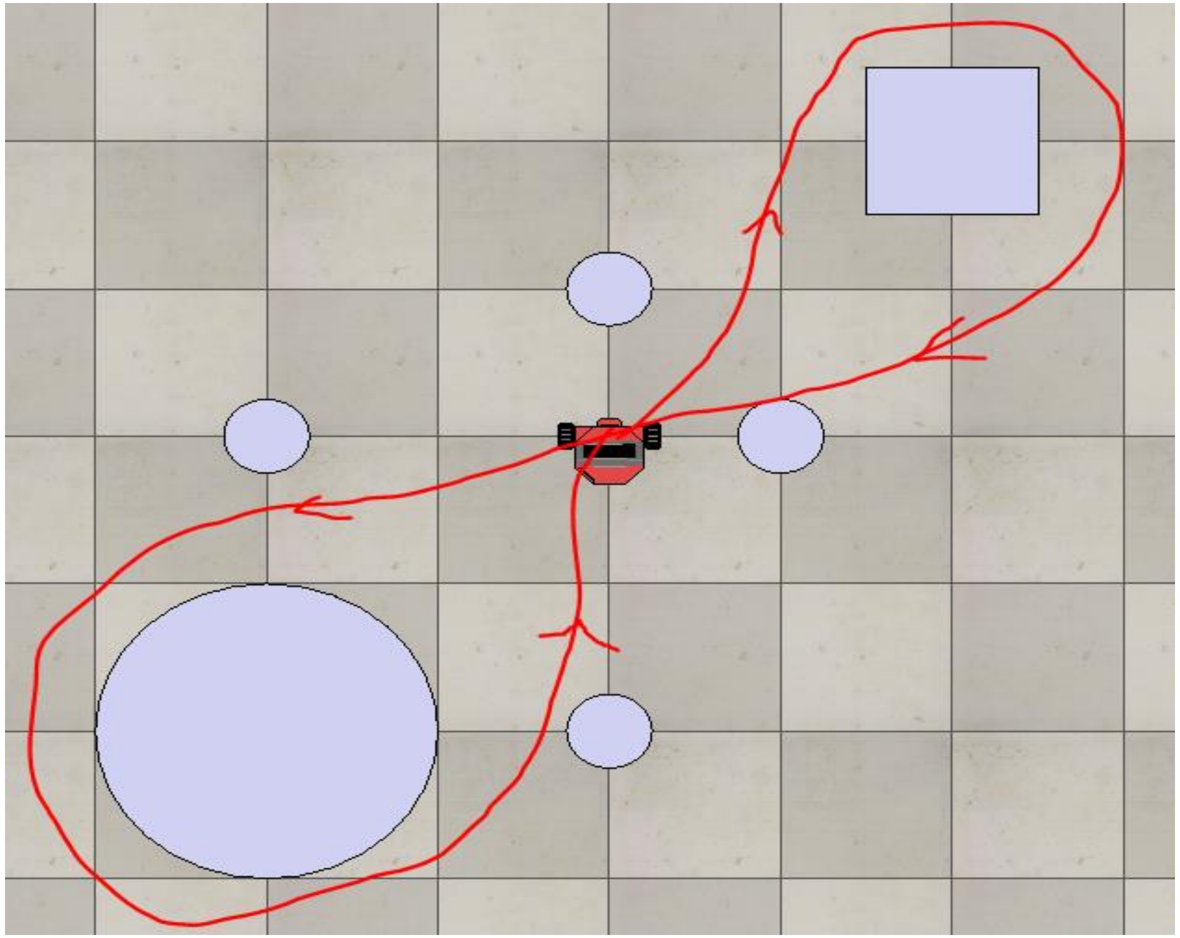
- Run command:

  - rosrun turtlesim turtle_teleop_key

  - rosrun image_view image_view image:=/d20_image

  - Maximise the view

Challenge 2: Compete



Thank You