Webots Tutorials

TUT #1

"World file" → "Nodes" (organized in a "Scene Tree")

.vbt — derived from → VRML97 language

"fields" (customizable properties)

Scene tree view
- WorldInfo
- Viewpoint
- Textured Background
- TexturedBackgroundLight
- RectangleArena

3D View

basicTimeStep → duration (in ms) of a physics step

Mouse
{Left drag}: rotate camera
{Right drag}: pan
{Middle drag}: zoom/rotate camera

- Tip: Pause, Reset, Modify simulation before saving (to avoid errors)

- "PROTO" node: Reuse "prototype"

{Ctrl + Alt + Left drag}: Apply force (Mouse)

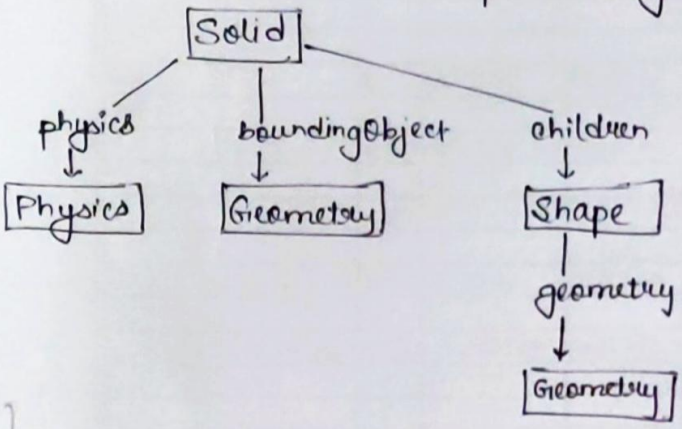
- "controller" field of Robot → epuck_go_forward

epuck_go_forward.py

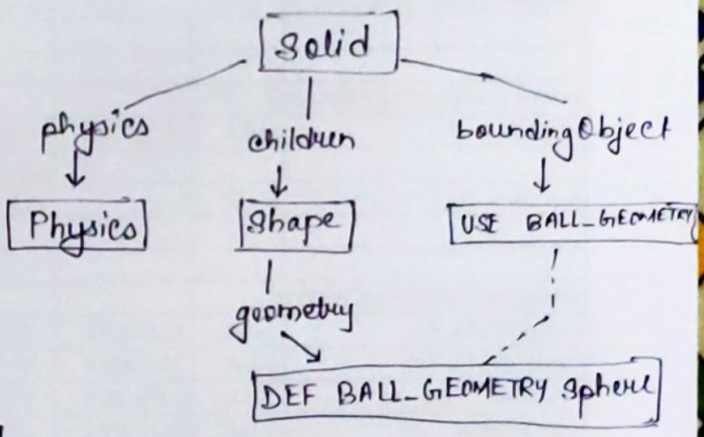
```
from controller import Robot
TIMESTEP = ~~628~~ 64
MAX_SPEED = 6.28

robot = Robot()
leftMotor = robot.getMotor('left wheel motor')
rightMotor = robot.getMotor('right wheel motor')

leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))

leftMotor.setVelocity(0.1*MAX_SPEED)
rightMotor.setVelocity(0.1*MAX_SPEED)

while robot.step(TIMESTEP) != -1:
    pass
```

TUT #2 : Modification of the Environment

- Modifying the Floor
- "Solid" Node
  ⇓
  rigid body. The physics engine of webots is capable of simulating rigid bodies only.
  ∴ An important step, when designing simulations, is to break up the various entities into separate rigid bodies.



- Created a Ball
- Sphere node → radius, subdivision
- DEF-USE Mechanism → (To reduce node redundancy)



- Added 4 walls definining shape only once.

TUT #3 : Appearance

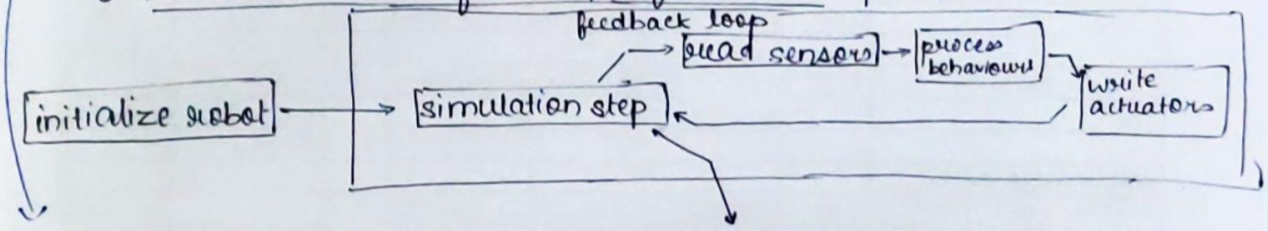
- Lights
  1. Directional Light
  2. SpotLight
  3. PointLight

- PBR : Physically Based Rendering
- Modify Appearance of the Walls [PBRAppearance]
  (Blue colour) ↳ BaseColour
- Add Texture to the Ball
  (Red Brick Wall Texture)
  via UV Mapping

TUT #4 : More About Controllers

epuck_avoid_collision.py : go forwards until an obstacle is detected by the front distance sensors, then turn towards the obstacle-free direction

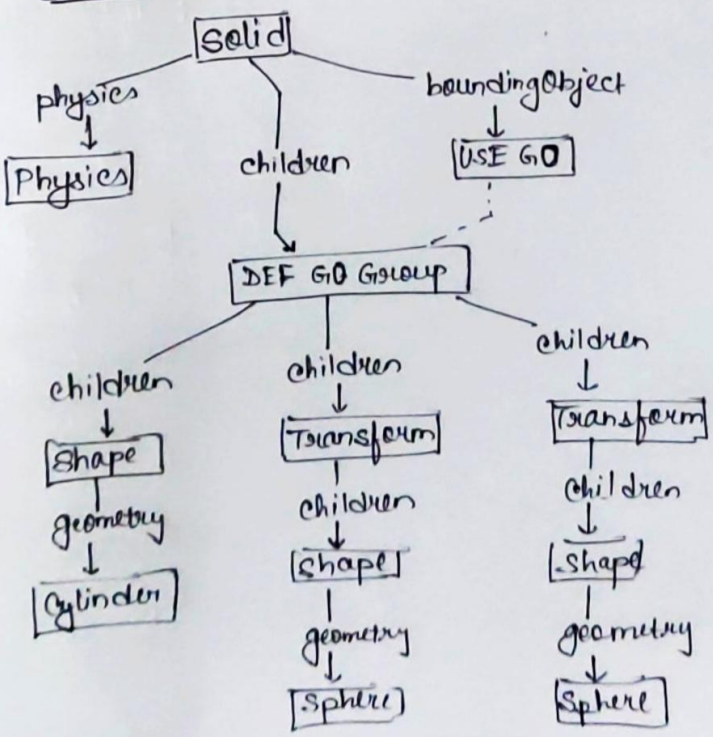
UML State Machine of a simple feedback loop




```
from controller import Robot
robot = Robot()
timestep = int(robot.getBasicTimeStep())
MAX_SPEED = 6.28

ps = []
psNames = [
    'ps0', 'ps1', 'ps2', 'ps3',
    'ps4', 'ps5', 'ps6', 'ps7'
]
for i in range(8):
    ps.append(robot.getDistanceSensor(psNames[i]))
    ps[i].enable(timestep)
leftMotor = robot.getMotor('left wheel motor')
rightMotor = robot.getMotor('right wheel motor')
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
```

```
leftMotor.setVelocity(0.0)
rightMotor.setVelocity(0.0)
while robot.step(timestep) != -1:
    psValues = []
    for i in range(8):
        psValues.append(ps[i].getValue())
    right_obstacle = psValues[0] > 80.0 or psValues[1] > 80.0 or
        psValues[2] > 80.0
    # left_obstacle = 
    left_speed = 0.5*MAX_SPEED
    right_speed = 0.5*MAX_SPEED
    if left_obstacle:
        left_speed += 0.5*MAX_SPEED
        right_speed -= 0.5*MAX_SPEED
    elif right_obstacle:
        #
        #
    leftMotor.setVelocity(left_speed)
    rightMotor.setVelocity(right_speed)
```

TUT #5 : Compound Solid and Physics Attributes



Physics Attributes: density or mass [either of them must be set to -1]

centerOfMass

- Choosing bounding objects
- contacts : contactMaterial

contactProperties in WorldInfo node

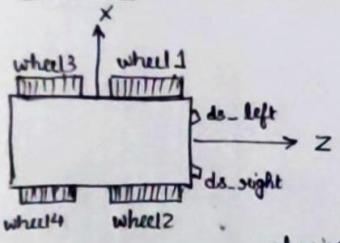
eg. coulombFriction

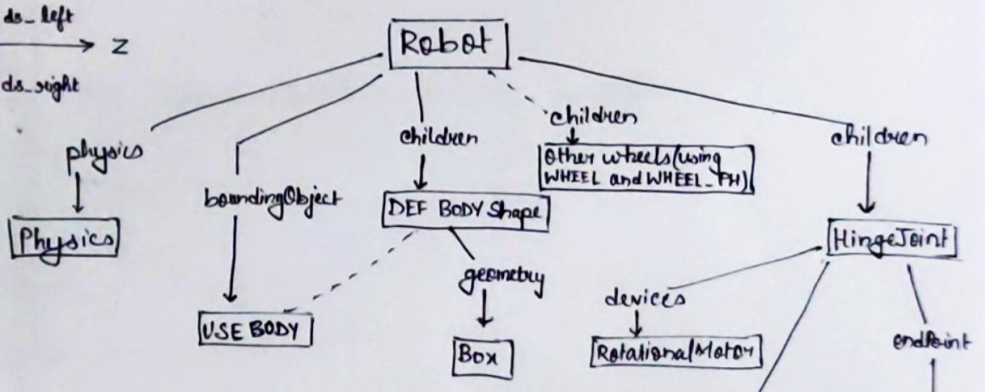
- basicTimeStep. ERP, CFM ← Used by ODE (Physics Engine)

{Error Reducing Parameter} {Constraint Force Mixing}

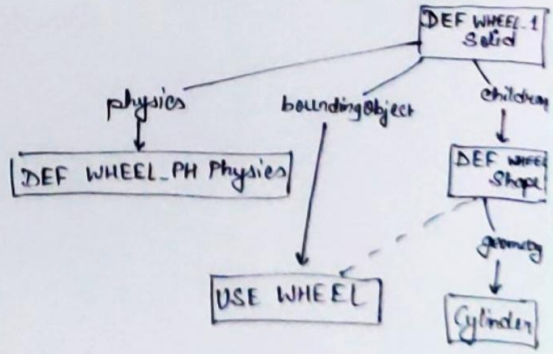
- Other physics parameters: linear damping on objects, inertia matrix, physicsDisableTime

TUT #6: 4-Wheels Robot





- HingeJoint
  - HingeJointParameters
    - anchor
    - axis
  - device
    - RotationalMotor
  - endPoint
    - Solid



- Sensors

"lookupTable"

TUT #7: First PROTO

```
PROTO FourWheelsRobot [                    → in protos/ directory
  field SFVec3f translation 0 0 0
  field SFRotation rotation 0 1 0 0
  field SFFloat bodyMass 1
]
{
  Robot {
    translation IS translation
    rotation IS rotation
    children [
      # ...
    ]
    boundingObject USE BODY
    physics Physics {
      density -1
      mass IS bodyMass
    }
    controller "four_wheels_collision_avoidance"
  }
}
```

TUT #8: Using ROS

- $ apt install ~~webot~~ ros-melodic-webots-ros
- Add to .bashrc or .zshrc:
  export WEBOTS_HOME=/usr/local/webots
- roslaunch webots_ros e-puck_line.launch