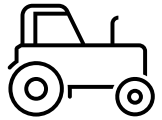
Introduction to Robotics I

01266212
CYBER PHYSICAL SYSTEM DESIGN

SEMESTER 1-2022

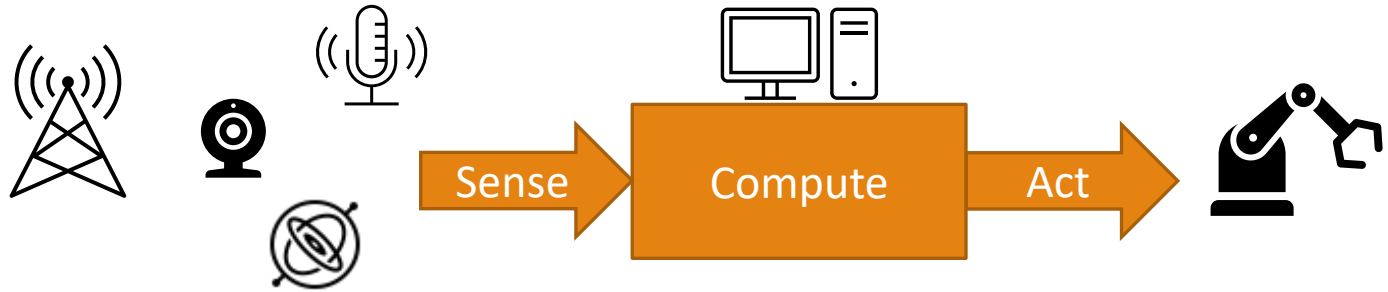
What Is Robot?





A robot is an autonomous machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.

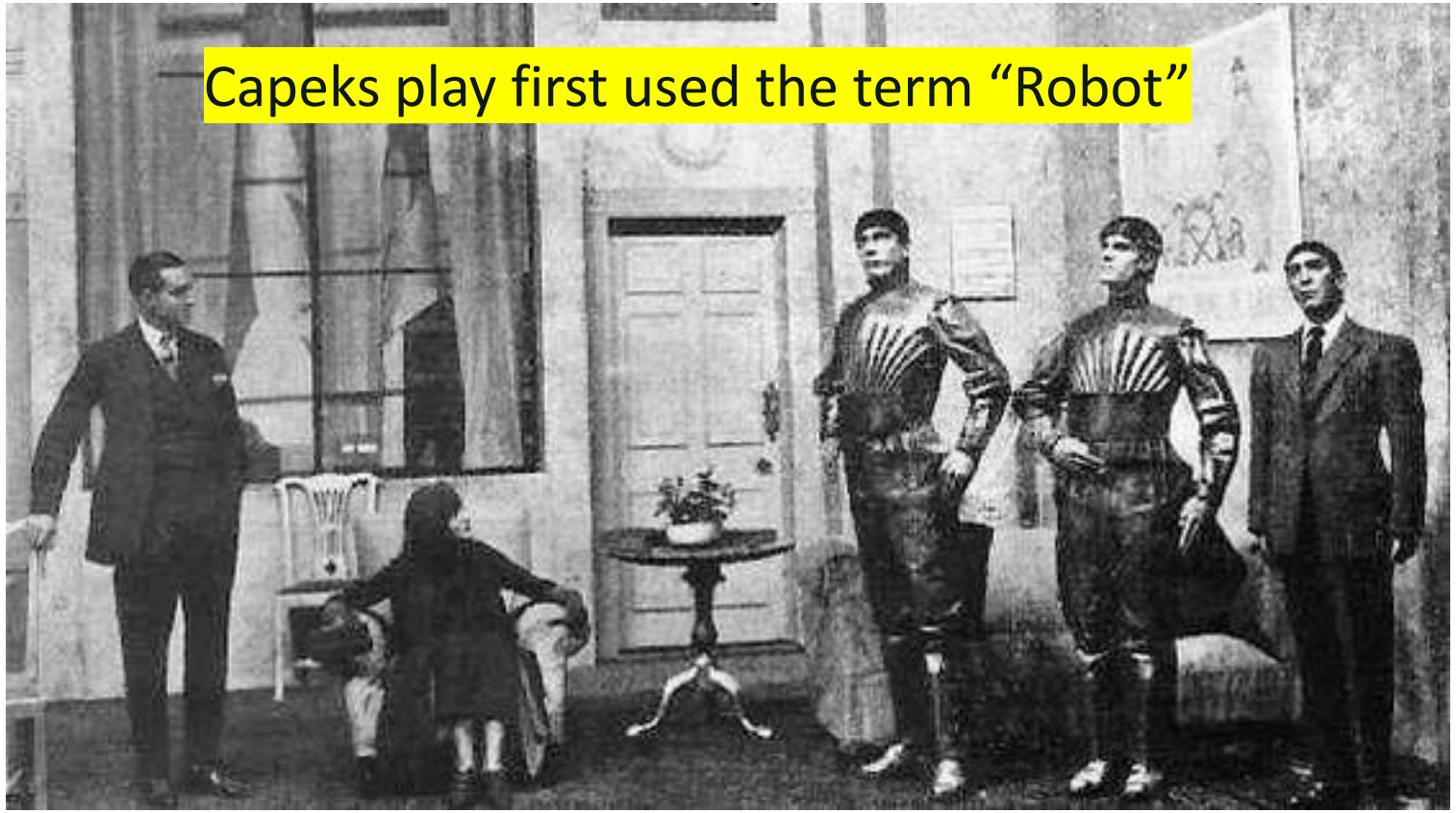
But some are dependent robots that interact with humans to enhance and supplement their already existing actions.



Although robots vary in how they sense, compute, and act, they all operate in a similar way: Their sensors feed measurements to a controller or computer, which processes them and then sends control signals to motors and actuators.

SOURCE: HTTPS://ROBOTS.IEEE.ORG/LEARN/WHAT-IS-A-ROBOT/

History of Robotics



Source: Wikipedia

1921 - The term “robot” was first used in a play called "R.U.R." or “Rossum's Universal Robots” by the Czech writer Karel Capek. Robot in Czech is a word for worker or servant

1941 - Science fiction writer Isaac Asimov first used the word "robotics" to describe the technology of robots and predicted the rise of a powerful robot industry.

History of Robotics

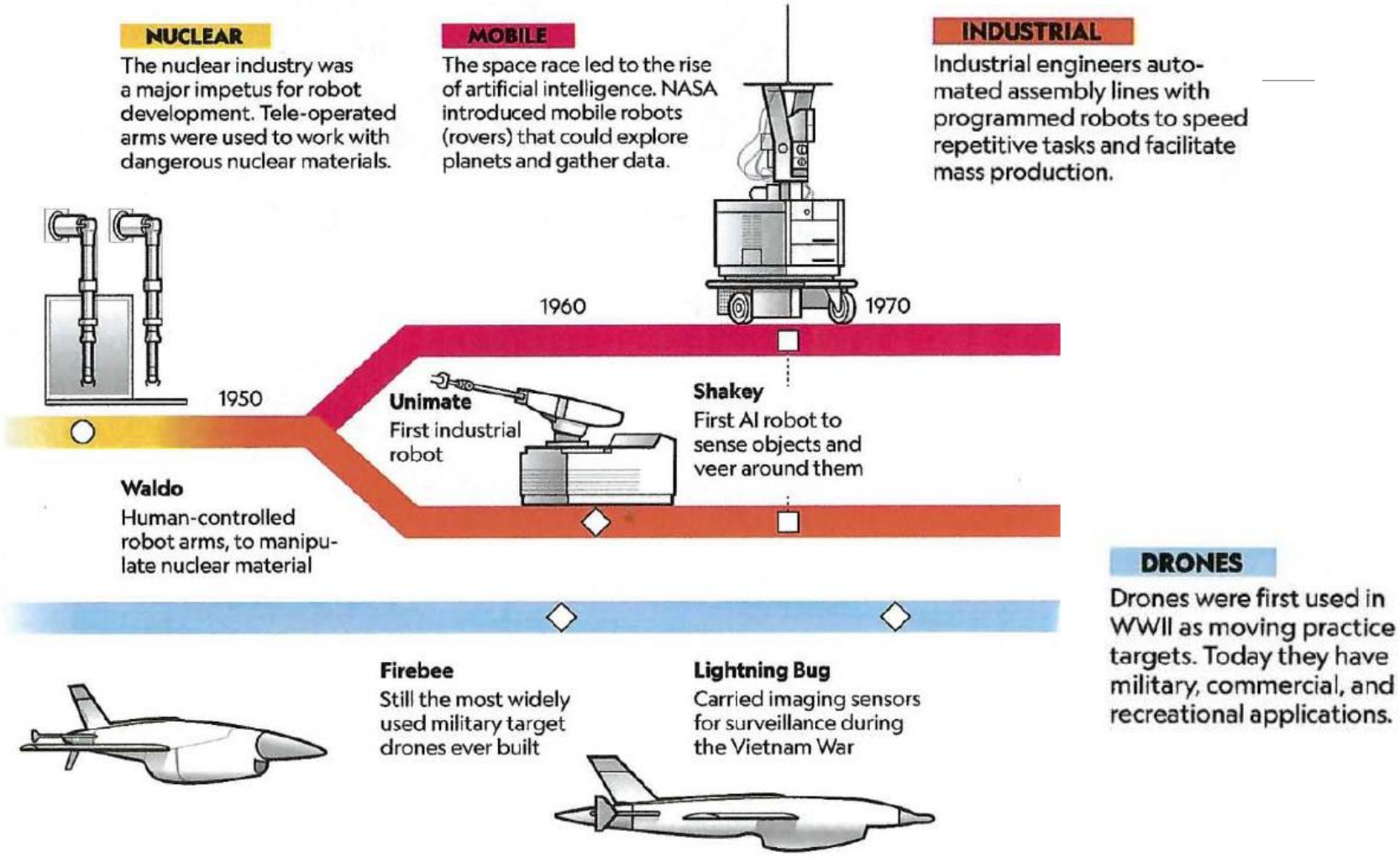
GETTING SMARTER

Some of the earliest robots of the mid-20th century were devices mostly controlled by nearby humans or were simple tools able to perform limited tasks.

Robot autonomy

- ○ Remotely operated
- ◇ Automated
- □ Artificial Intelligence

Year marks date of commercialization or extensive use. Illustrations are not to scale.



Robot History Timeline, 1950s – 1970s

The world first industrial robot: UNIMATE

In 1954 first programmable robot is designed by George Devol, who coins the term Universal Automation. He later shortens this term to Unimation, which become the name of the first robot company in 1962.

The world's first AI-based robot, Shakey

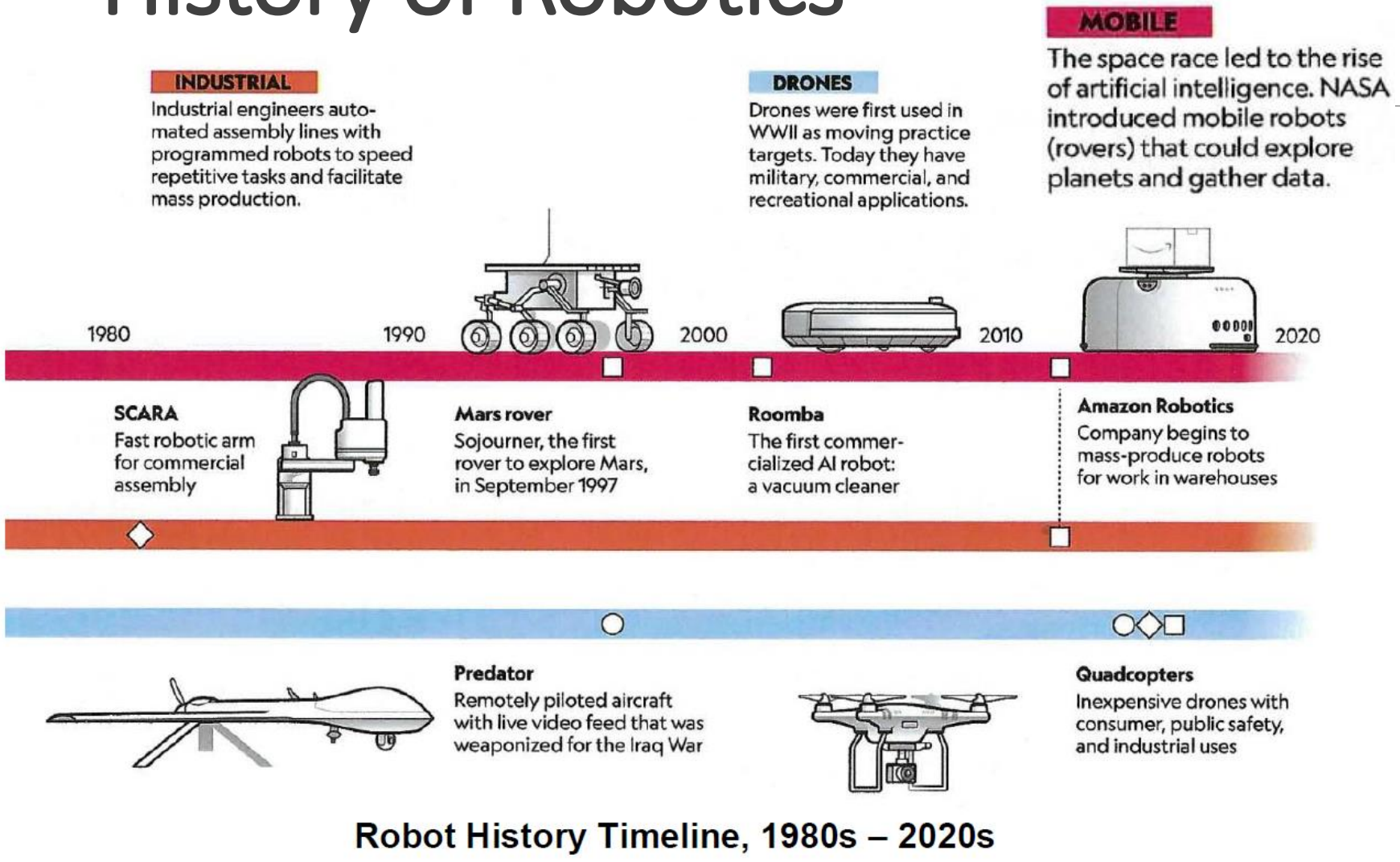
Shakey the world's first autonomous robot to move around, make human-like decisions on the fly and take action. Created by researchers at SRI, Shakey earned legendary status for its combination of robotics and AI into one system.

Shakey was developed at SRI International from 1966 to 1972.

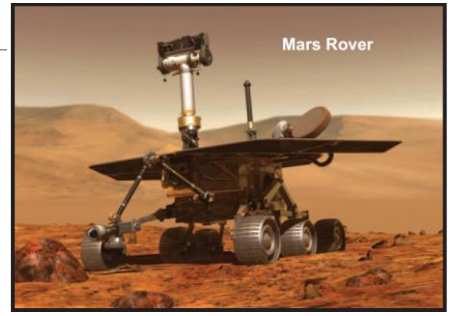
https://www.youtube.com/watch?v=7bsEN8mwUB8

SOURCE: HTTPS://WWW.SRI.COM/CASE-STUDIES/THE-MAN-THE-MYTH-THE-LEGEND-MEET-SHAKEY-THE-ROBOT-THE-WORLDS-FIRST-AI-BASED-ROBOT/

History of Robotics



Robot History Timeline, 1980s – 2020s



Types of Robots

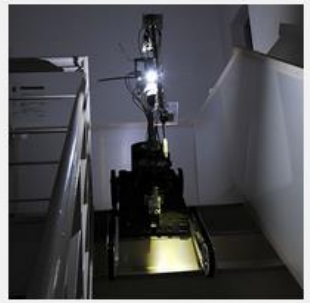


Aerospace: This is a broad category. It includes all sorts of flying robots—the SmartBird robotic seagull and the Raven surveillance drone, for example—but also robots that can operate in space, such as Mars rovers and NASA's Robonaut, the humanoid that flew to the International Space Station and is now back on Earth.



Consumer: Consumer robots are robots you can buy and use just for fun or to help you with tasks and chores. Examples are the robot dog Aibo, the Roomba vacuum, AI-powered robot assistants, and a growing variety of robotic toys and kits.





Disaster Response: These robots perform dangerous jobs like searching for survivors in the aftermath of an emergency. For example, after an earthquake and tsunami struck Japan in 2011, Packbots wer used to inspect damage at the Fukushima Daiichi nuclear power station.



Drones: Also called unmanned aerial vehicles, drones come in different sizes and have different levels of autonomy. Examples include DJI's popular Phantom series and Parrot's Anafi, as well as military systems like Global Hawk, used for long-duration surveillance.

Types of Robots



Education: This broad category is aimed at the next generation of roboticists, for use at home or in classrooms. It includes hands-on programmable sets from Lego, 3D printers with lesson plans, and even teacher robots like EMYS.



Exoskeletons: Robotic exoskeletons can be used for physical rehabilitation and for enabling a paralyzed patient walk again. Some have industrial or military applications, by giving the wearer added mobility, endurance, or capacity to carry heavy loads.



Entertainment: These robots are designed to evoke an emotional response and make us laugh or feel surprise or in awe. Among them are robot comedian RoboThespian, Disney's theme park robots like Navi Shaman, and musically inclined bots like Partner.



Humanoids: This is probably the type of robot that most people think of when they think of a robot. Examples of humanoid robots include Honda's Asimo, which has a mechanical appearance, and also androids like the Geminoid series, which are designed to look like people.

Types of Robots


Industrial: The traditional industrial robot consists of a manipulator arm designed to perform repetitive tasks. An example is the Unimate, the grandfather of all factory robots. This category includes also systems like Amazon's warehouse robots and collaborative factory robots that can operate alongside human workers.



Medical: Medical and health-care robots include systems such as the da Vinci surgical robot and bionic prostheses, as well as robotic exoskeletons. A system that may fit in this category but is not a robot is Watson, the IBM question-answering supercomputer, which has been used in healthcare applications.



Military & Security: Military robots include ground systems like Endeavor Robotics' PackBot, used in Iraq and Afghanistan to scout for improvised explosive devices, and BigDog, designed to assist troops in carrying heavy gear. Security robots include autonomous mobile systems such as Cobalt.



Research: The vast majority of today's robots are born in universities and corporate research labs. Though these robots may be able to do useful things, they're primarily intended to help researchers do, well, research. So although some robots may fit other categories described here, they can also be called research robots.

Types of Robots



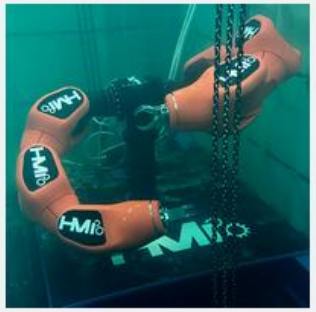
Self-Driving Cars: Many robots can drive themselves around, and an increasing number of them can now drive *you* around. Early autonomous vehicles include the ones built for DARPA's autonomous-vehicle competitions and also Google's pioneering self-driving Toyota Prius, later spun out to form Waymo.

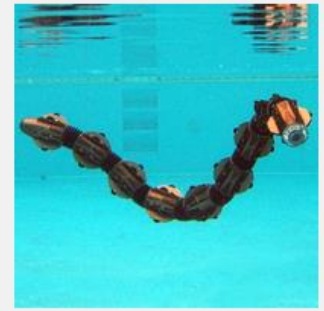


Telepresence: Telepresence robots allow you to be present at a place without actually going there. You log on to a robot avatar via the internet and drive it around, seeing what it sees, and talking with people. Workers can use it to collaborate with colleagues at a distant office, and doctors can use it to check on patients.



Telepresence: Telepresence robots allow you to be present at a place without actually going there. Yo log on to a robot avatar via the internet and drive it around, seeing what it sees, and talking with people. Workers can use it to collaborate with colleagues at a distant office, and doctors can use it to check on patients.





Underwater: The favorite place for these robots is in the water. They consist of deep-sea submersibles like Aquanaut, diving humanoids like Ocean One, and bio-inspired systems like the ACM-R5H snakebot.

Pop-Culture Droids and Humanoid Robots

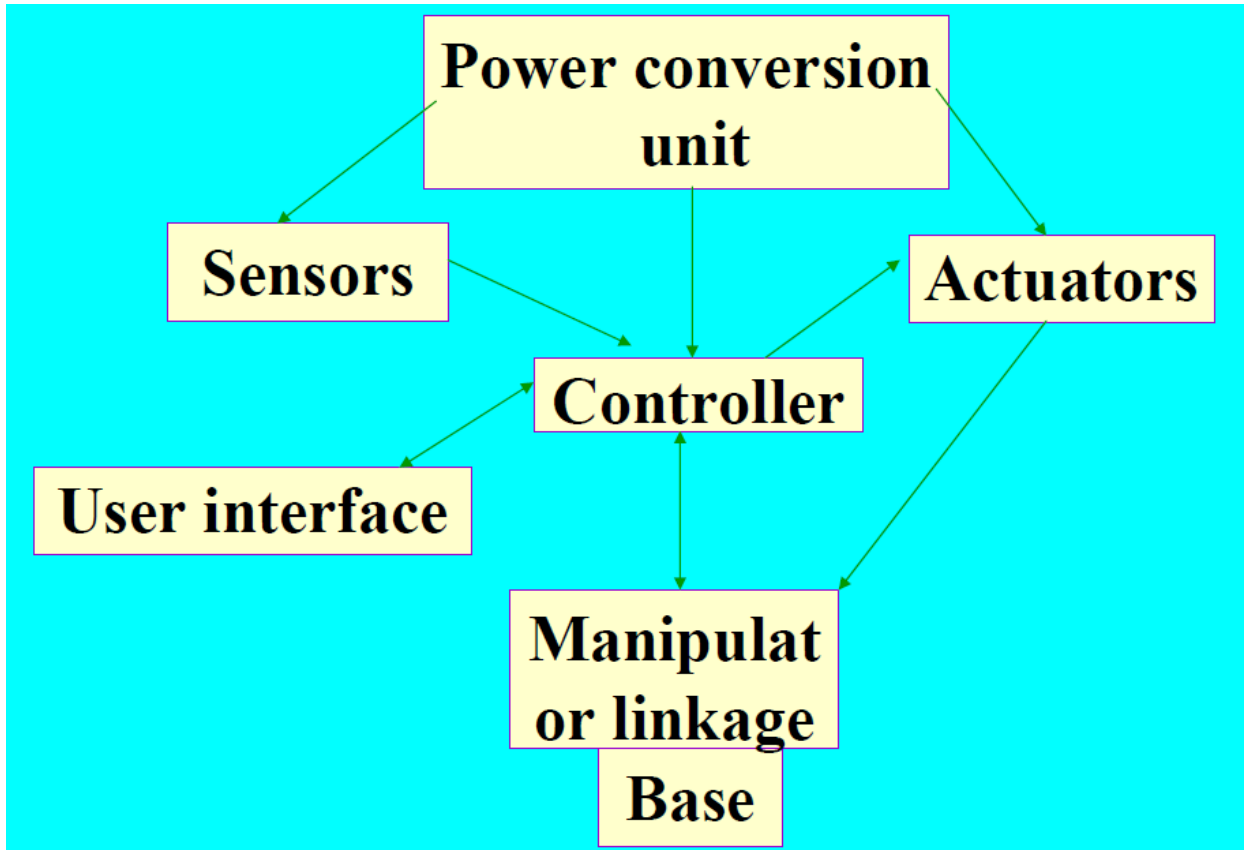
What is Robotics?

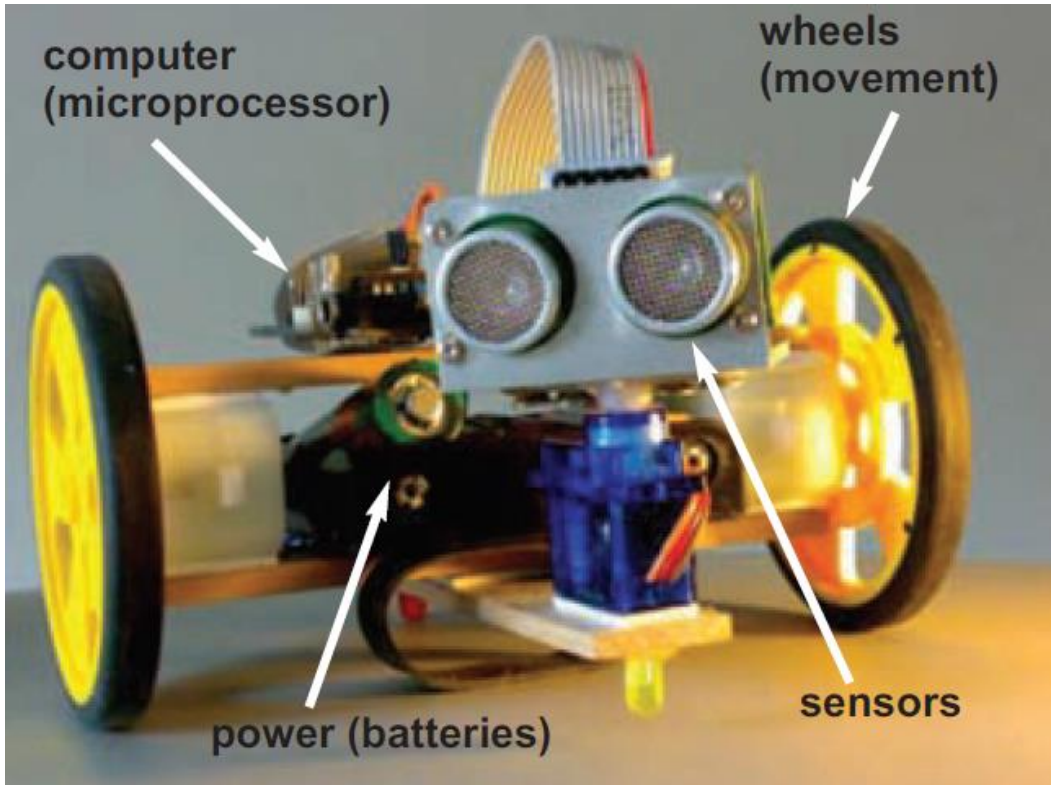
Robotics is a branch of Artificial Intelligence (AI), it is mainly composed of electrical engineering, mechanical engineering and computer science engineering for construction, designing and application of robots.

Robotics is science of building or designing an application of robots. The aim of robotics is to design an efficient robot.

- The robots have
  - ✓ **electrical components** for providing power and control the machinery.
  - ✓ **mechanical construction,** shape, or form designed to accomplish a particular task.
  - ✓ some type of **computer program** that determines what, when and how a robot does something.

Components of Robot





Mobile Robots

Mobile robots are able to move from one location to another location using locomotion. It is an automatic machine that is capable of navigating an uncontrolled environment without any requirement of physical and electromechanical guidance devices. Mobile Robots are of two types:

- **(a) Rolling robots -** Rolling robots require wheels to move around. They can easily and quickly search. But they are only useful in flat areas.
- **(b) Walking robots -** Robots with legs are usually used in condition where the terrain is rocky. Most walking robots have at least 4 legs.

SOURCE HTTPS://WWW.JAVATPOINT.COM/ROBOTICS-TUTORIAL

Mobile Robot and Locomotion

Mobile robots are able to move from one location to another location using locomotion. Locomotion is the method of moving from one place to another. The mechanism that makes a robot capable of moving in its environment is called as robot locomotion.

There are many types of locomotion's:-

•Wheeled

•Legged

•Combination of legged and wheeled locomotion

•Tracked slip/skid

SOURCE HTTPS://WWW.JAVATPOINT.COM/ROBOTICS-TUTORIAL

Legged locomotion

- Requires leg coordination for locomotion and difficult to implement because of stability issues.
- Requires more motors to accomplish a movement.
- Consumes more power while demonstrating jump, hop, walk, trot, climb up or down etc.

source: Wikipedia

Source: https://spectrum.ieee.org/

Source: https://robohub.org/

Wheeled and Slip/Skid Locomotion

- Requires a smaller number of motors for movement so little easy to implement as there are lesser stability issues
- More power efficient as compared to legged locomotion.
- Different types of week
  1. Castor wheel
  2. Standard wheel
  3. Ball or spherical wheel -
  4. Swedish 45 and Swedish 90 wheels

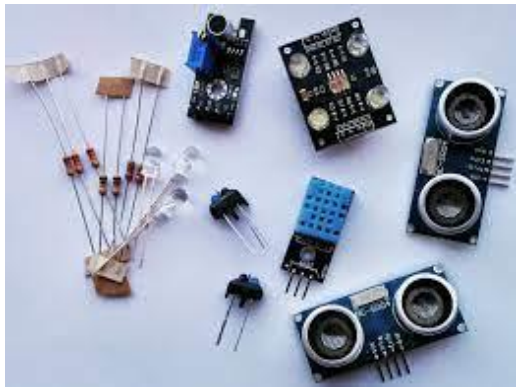
Slip/Skid:

- Use tracks as available in a tank.
- Is steered by moving tracks with different speeds in the same or opposite direction. I
- Offers stability because of large contact area of ground and track.

SOURCE HTTPS://WWW.JAVATPOINT.COM/ROBOTICS-TUTORIAL

Types of Robot Sensors

1. **Light Sensor:** detect light
2. **Proximity Sensor:** detect the presence of nearby object without any physical contact.
3. **Sound Sensor:** to detect sound
4. **Temperature Sensor**
5. **Acceleration Sensor**
6. **Gyroscope or IMU (Inertial Measurement Units)** to measure orientation, velocity and gravitational forces.



Types of Robot Actuators

Motion

- Linear Actuators
- Rotary Actuators

Source of Energy

- Hydraulic Actuators
- Pneumatic Actuators
- Electric Actuators
- Thermal and Magnetic Actuators
- Mechanical Actuators

Humanoid Robot

- Companions and assistants for humans in daily life
- Helpers in man-made and natural disasters (DARPA Robotics Challenge)

[source: Honda]

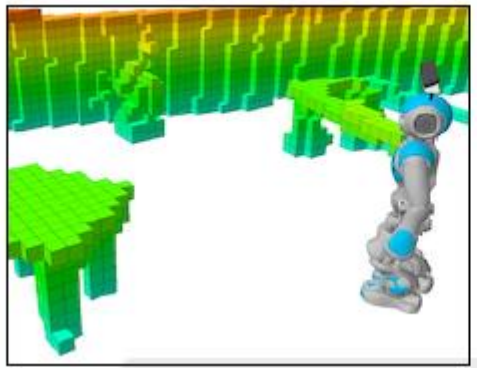
[source: DARPA]

Why Humanoid Robots?

- ✓ Human-like body plan
- ✓ Can act in environments designed for humans
- ✓ Can navigate in multi-story and cluttered scenes
- ✓ Can manipulate objects and use tools of humans
- ✓ Can learn from humans by imitation their motions

[image: Honda]

e: O.v.Stryk]



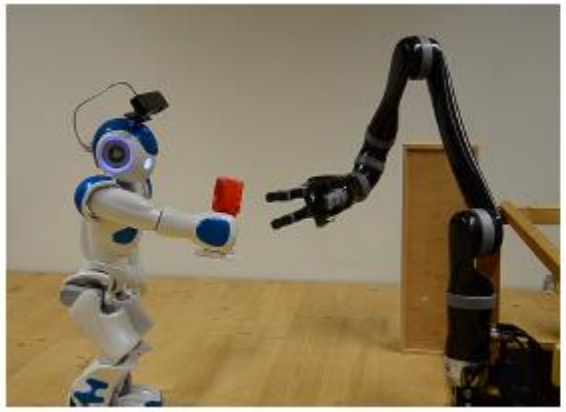
What Makes it Difficult?

Noisy sensor data

Extraction of relevant information

Inaccurate motion execution

High-dimensional state space




Programming for Robotics

Robots process sensor data, perform cognition and plan actions using computer programs that are executed on a processor.

Robot programming is the development of a control scheme for how a machine interacts with its environment and achieves its goals.

Robot Operating System (ROS) is a set of software libraries and tools that helps you build robot applications. You can also write your own programs for ROS e.g. in C/C++ or Python.

Programming Languages in Robotics

- C/C++ Choosing to learn C and C++ is the best start for an aspiring roboticists as it is a general purpose programming language that contains imperative, object-oriented and generic programming features. ...
- Python. ...
- Java. ...
- C#/.NET. ...
- MATLAB

Robotics Simulator

HTTPS://WWW.CYBERBOTICS.COM/DOC/GUIDE/INDEX

Robotic Simulator

A **robotics simulator** is a simulator used to create application for a physical robot without depending on the actual machine, thus saving cost and time. In some case, these applications can be transferred onto the physical robot (or rebuilt) without modifications.

Software	Developers	Development status	License	3D rendering engine	Physics engine	3D modeller	Platforms supported
Gazebo	Open Source Robotics Foundation(OSRF)	Active	Apache 2.0	OGRE	ODE/Bullet /Simbody/DART	Internal	Linux, macOS, Windows
RoboDK	RoboDK	Active	Proprietary	OpenGL	Gravity plugin	Internal	Linux, macOS, Windows, Android, iOS, Debian
SimSpark	O. Obst et al. (+26)	Active	GNU GPL (v2)	Internal	ODE	None	Linux, macOS, Windows
Webots	Cyberbotics Ltd.	Active	Apache 2.0	Internal (WREN)	Fork of ODE	Internal	Linux, macOS, Windows
OpenRAVE	OpenRAVE Community	Active	GNU LGPL	Coin3D/OpenSceneGraph	ODE/Bullet	Internal	Linux, macOS, Windows

SOURCE: WIKIPEDIA

Robotic Simulator

Software	Main programming language	Formats support	Extensibility	External APIs	Robotics middleware support	Primary user interface	Headless simulation
Gazebo	C++	SDF[1]/URDF[2], OBJ, STL, Collada	Plugins (C++)	C++	ROS, Player, Sockets (protobuf messages)	GUI	Yes
RoboDK	Python	SLDPRT, SLDASM, STEP, OBJ, STL, 3DS, Collada, VRML, URDF, Rhinoceros_3D, ...	API[3], Plug-In Interface[4]	Python, C/C++, C#, Matlab, ...	Socket	GUI	Unknown
SimSpark	C++, Ruby	Ruby Scene Graphs	Mods (C++)	Network (sexpr)	Sockets (sexpr)	GUI, Sockets	Unknown
Webots	C++	WBT, VRML, X3D, 3DS, Blender, BVH, Collada, FBX, STL, OBJ, URDF	API, PROTOs, Plugins (C/C++)	C, C++, Python, Java, Matlab, ROS	Sockets, ROS, NaoQI	GUI	Yes[5]
OpenRAVE	C++, Python	XML, VRML, OBJ, Collada	Plugins (C++), API	C/C++, Python, Matlab	Sockets, ROS, YARP	GUI, Sockets	Yes

Webots Simulator

Webots is an open-source three-dimensional mobile robot simulator.

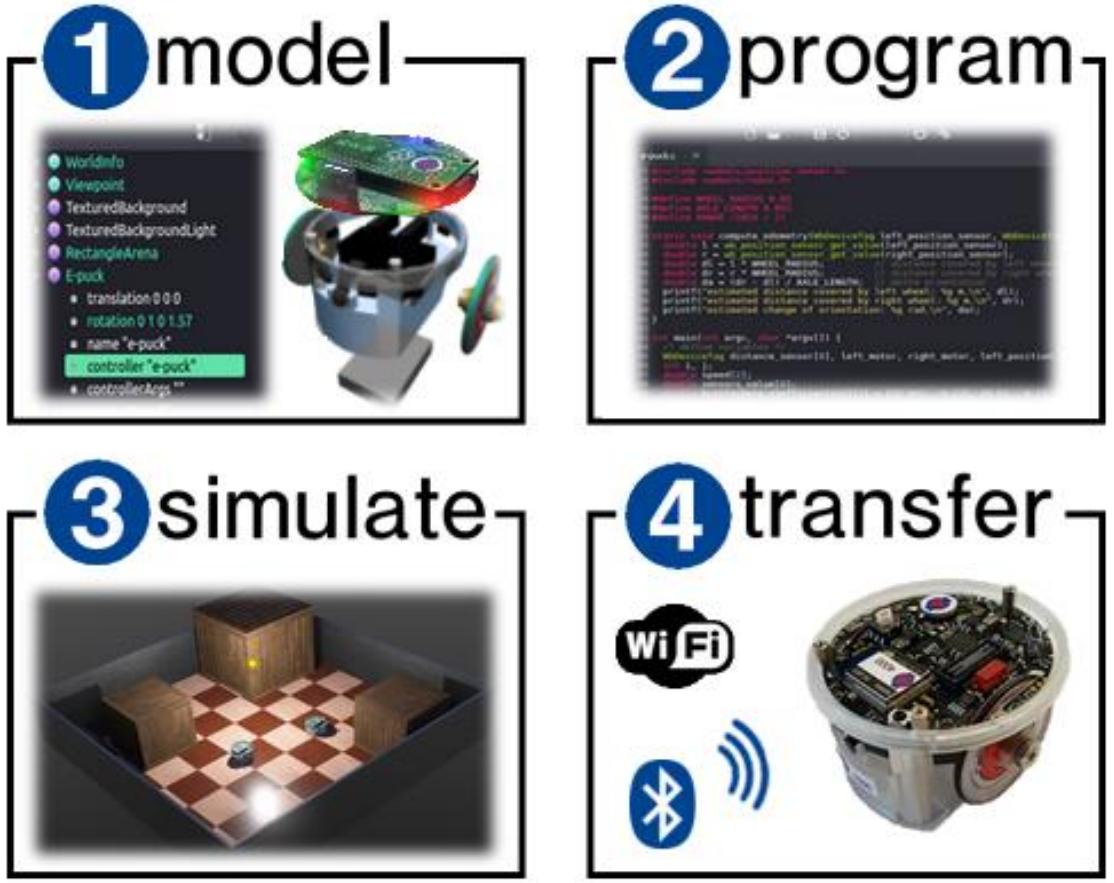
It was originally developed as a research tool to investigate various control algorithms in mobile robotics. Since December 2018, Webots is released as an open source software under the Apache 2.0 license.

It is expected to have minimal knowledge in mobile robotics, in C, C++, Java, Python or MATLAB programming, and in VRML97 (Virtual Reality Modeling Language).

The latest version is R2022b.

https://www.cyberbotics.com

Webots Simulator



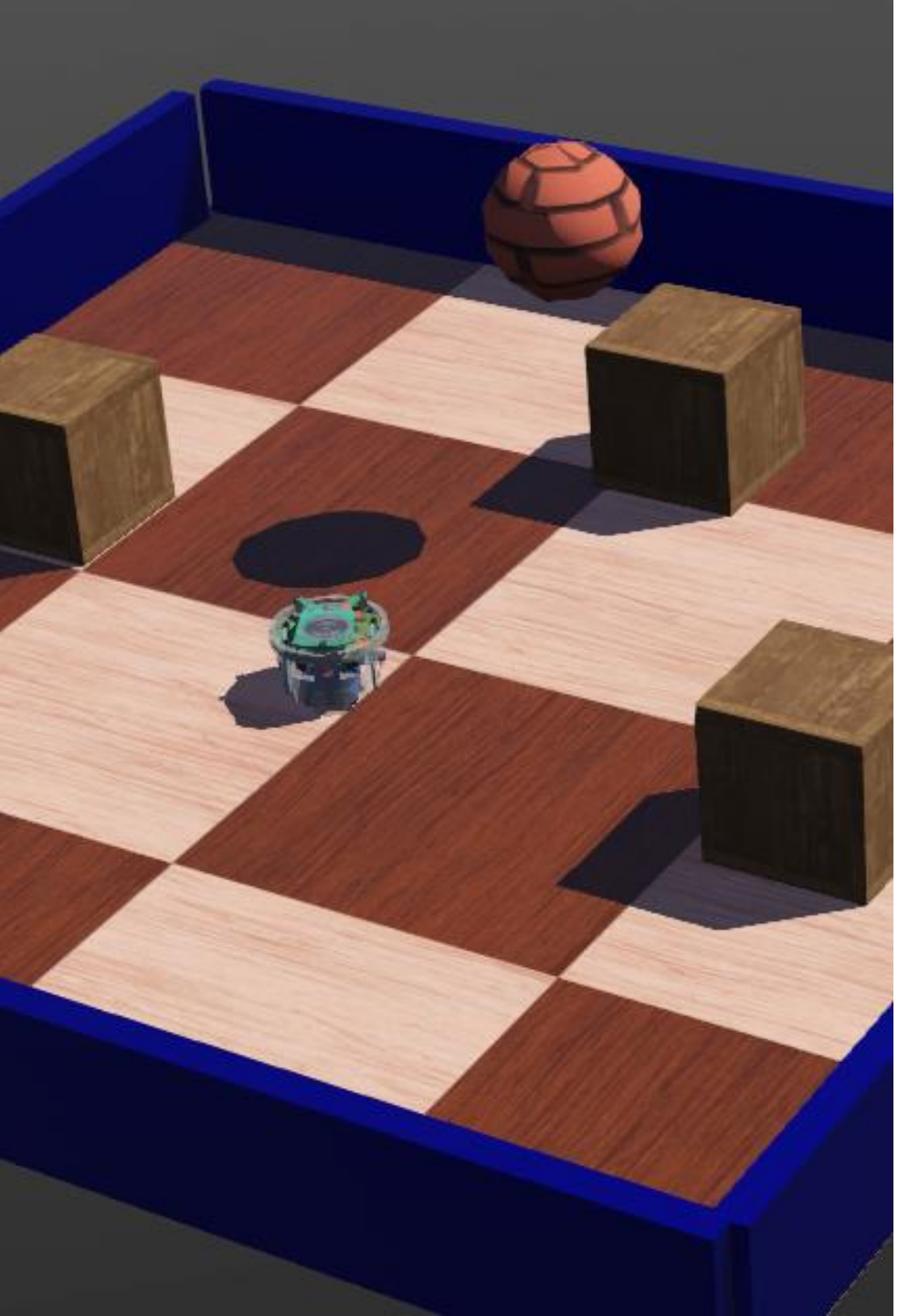
What Is Webots?

- It is a professional mobile robot simulation software package.
- It offers a rapid prototyping environment, that allows the user to create 3D virtual worlds with physics properties such as mass, joints, friction coefficients, etc.
- The user can add simple passive objects or active objects called mobile robots(wheeled robots, legged robots, or flying robots).
- Moreover, they may be equipped with a number of sensor and actuator devices, such as distance sensors, drive wheels, cameras, motors, touch sensors, emitters, receivers, etc.
- Finally, the user can program each robot individually to exhibit the desired behavior.
- Webots contains a large number of robot models and controller program examples to help users get started.

Webots Simulation

A Webots simulation is composed of following items:

1. A Webots *world* file (.wbt) that defines one or several robots and their environment. The .wbt file does sometimes depend on external PROTO files (.proto) and textures.
2. One or several controller programs for the above robots (in C/C++/Java/ Python/MATLAB).
3. An optional physics plugin that can be used to modify Webots regular physics behavior (in C/C++).



What Is a World?

- A world, in Webots, is a 3D description of the properties of robots and of their environment.
- It contains a description of every object: position, orientation, geometry, appearance (like color or brightness), physical properties, type of object, etc.
- It defines the initial state of a simulation.
- Worlds are organized as hierarchical structures where objects can contain other objects.

For example, a robot can contain two wheels, a distance sensor and a joint which itself contains a camera, etc.

- Worlds are saved in ".wbt" files. The ".wbt" files are stored in the "worlds" subdirectory of each Webots project.

The file format is derived from the **VRML97** language, and is human readable.

A Simulation World

A **World** is a file containing information like where the objects are, what they look like, how they interact with each other, what is the color of the sky, how is defined the gravity, friction, masses of the objects, etc.

The world files must be stored directly in a directory called worlds.

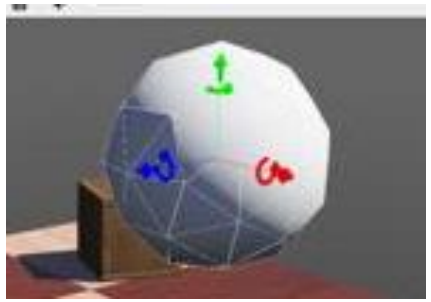
In a world, the different objects are called **Nodes** and are organized hierarchically in a **Scene Tree**.

A node may contain sub-nodes.

".wbt" file

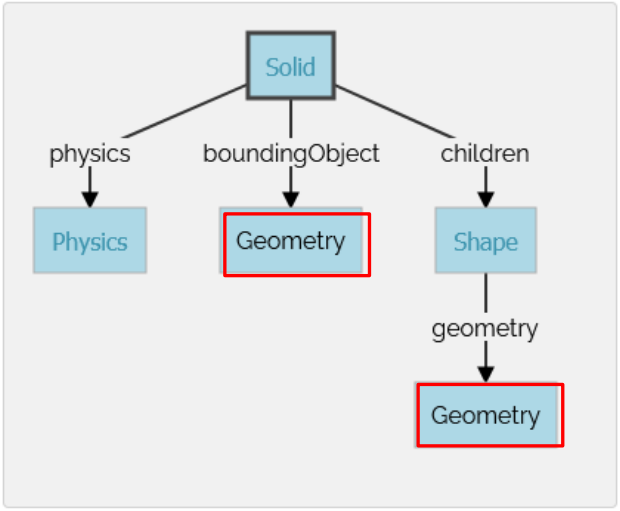
```
#VRML_SIM V8.5 utf8

WorldInfo {
  info [
    "Description"
    "Author: first name last name <e-ma
    "Date: DD MMM YYYY"
  ]
}
Viewpoint {
  orientation 1 0 0 -0.8
  position 0.25 0.708035 0.894691
}
Background {
  skyColor [
    0.4 0.7 1
  ]
}
PointLight {
  ambientIntensity 0.54
  intensity 0.5
  location 0 1 0
}
```

The Solid Node



The collision bounds are defined in its boundingObject field.

The physics field defines if the object belongs to the dynamical or to the static environment.



The graphical representation of the Solid node is defined by the Shape nodes populating its children list.

A Solid node represents a **rigid body**, that is a body in which deformation can be neglected.

The distance between any two given points of a rigid body remains constant in time regardless of external forces exerted on it.

The physics engine of Webots is designed for simulating rigid bodies only.

Inside a solid node, there are different sub-nodes corresponding to the characteristics of the rigid body.

All these sub-nodes are optional, but the physics field needs the boundingObject to be defined.

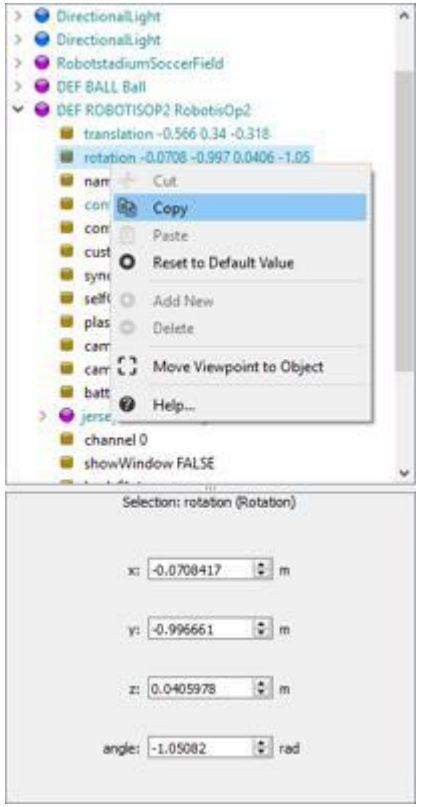
The Scene Tree

To access the Scene Tree Window you can either choose Scene Tree in the Tools menu, or press the Show Scene Tree button in the main toolbar.

The scene tree contains the information that describes a simulated world, including robots and environment, and its graphical representation.

The scene tree contains the following information and the nodes.

- WorldInfo: contains global parameters of the simulation.
- Viewpoint: defines the main viewpoint camera parameters.
- TexturedBackground: defines the background of the scene (you should see mountains far away if you rotate a little bit the viewpoint)
- TexturedBackroundLight: defines the light associated with the above background.
- RectangleArena: define the only object you see so far in this scene.



four_wheeled_collision_avoidance.py

```python
from controller import Robot

TIME_STEP = 64
robot = Robot()
ds = []
dsNames = ['ds_right', 'ds_left']
for i in range(2):
    ds.append(robot.getDevice(dsNames[i]))
    ds[i].enable(TIME_STEP)
wheels = []
wheelsNames = ['wheel1', 'wheel2', 'wheel3', 'wheel4'
for i in range(4):
    wheels.append(robot.getDevice(wheelsNames[i]))
    wheels[i].setPosition(float('inf'))
    wheels[i].setVelocity(0.0)
avoidObstacleCounter = 0
while robot.step(TIME_STEP) != -1:
    leftSpeed = 1.0
    rightSpeed = 1.0
    if avoidObstacleCounter > 0:
        avoidObstacleCounter -= 1
        leftSpeed = 1.0
        rightSpeed = -1.0
    else:  # read sensors
        for i in range(2):
            if ds[i].getValue() < 950.0:
                avoidObstacleCounter = 100
    wheels[0].setVelocity(leftSpeed)
    wheels[1].setVelocity(rightSpeed)
    wheels[2].setVelocity(leftSpeed)
    wheels[3].setVelocity(rightSpeed)
```

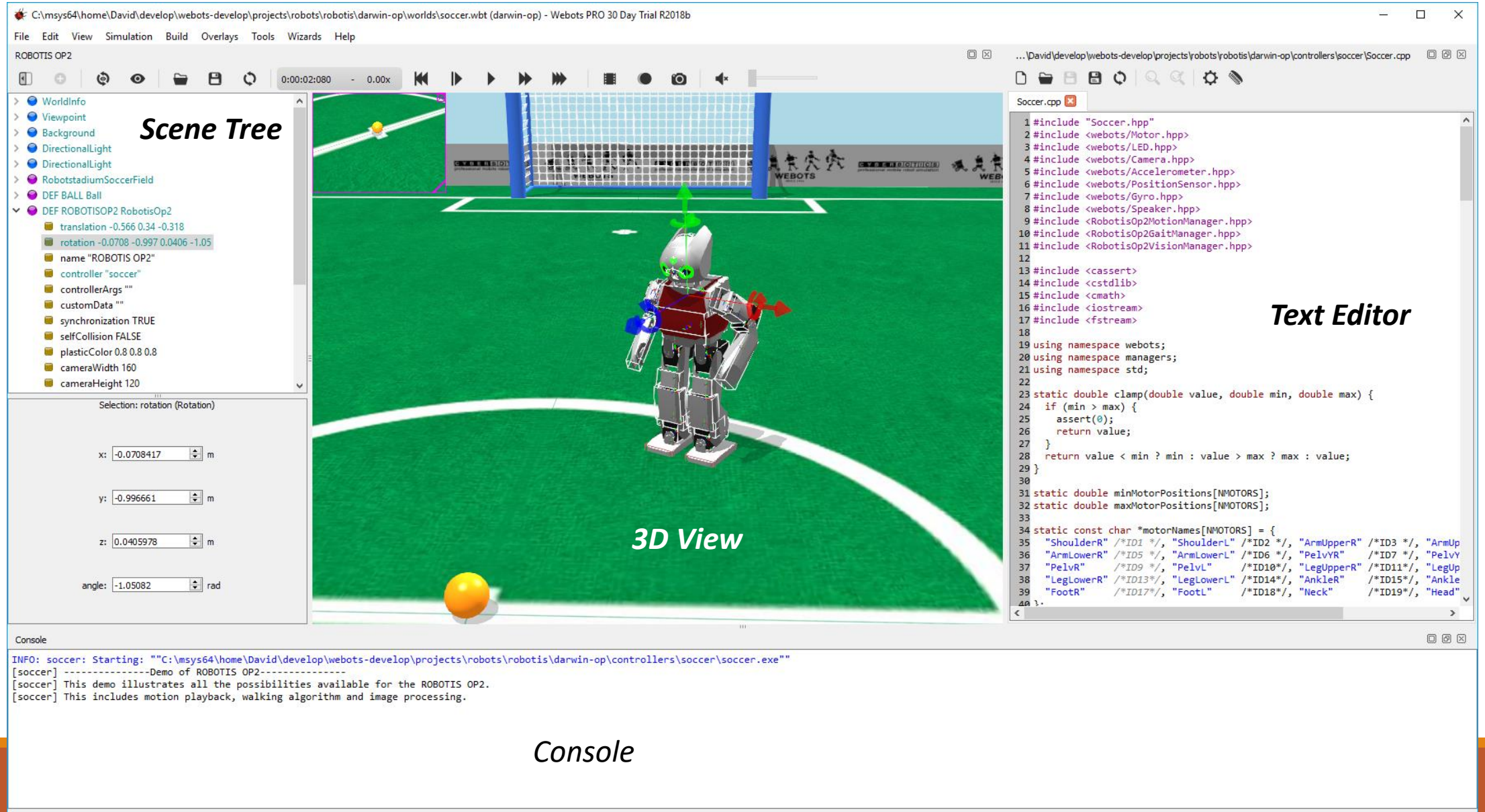
What Is a Controller?

- A controller is a computer program that controls a robot specified in a world file.
- Controllers can be written in any of the programming languages supported by Webots: C, C++, Java, Python or MATLAB.
- When a simulation starts, Webots launches the specified controllers, each as a separate process, and it associates the controller processes with the simulated robots.

Note that several robots can use the same controller code, however a distinct process will be launched for each robot.

The source files and binary files of each controller are stored together in a controller directory. A controller directory is placed in the "controllers" subdirectory of each Webots project.

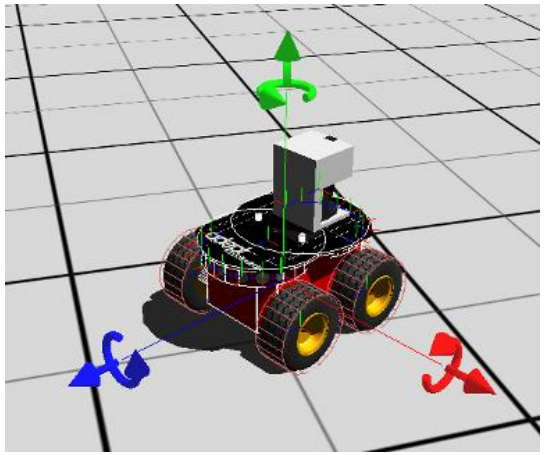
The User Interface



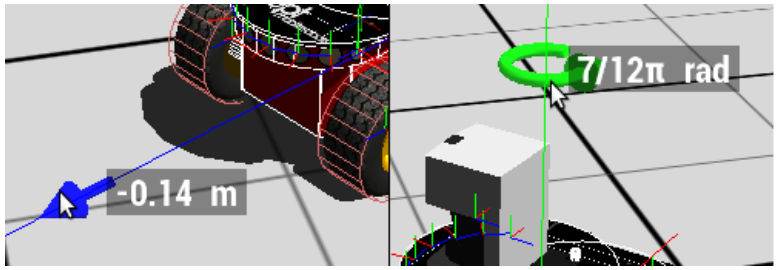
Console

The 3D Window

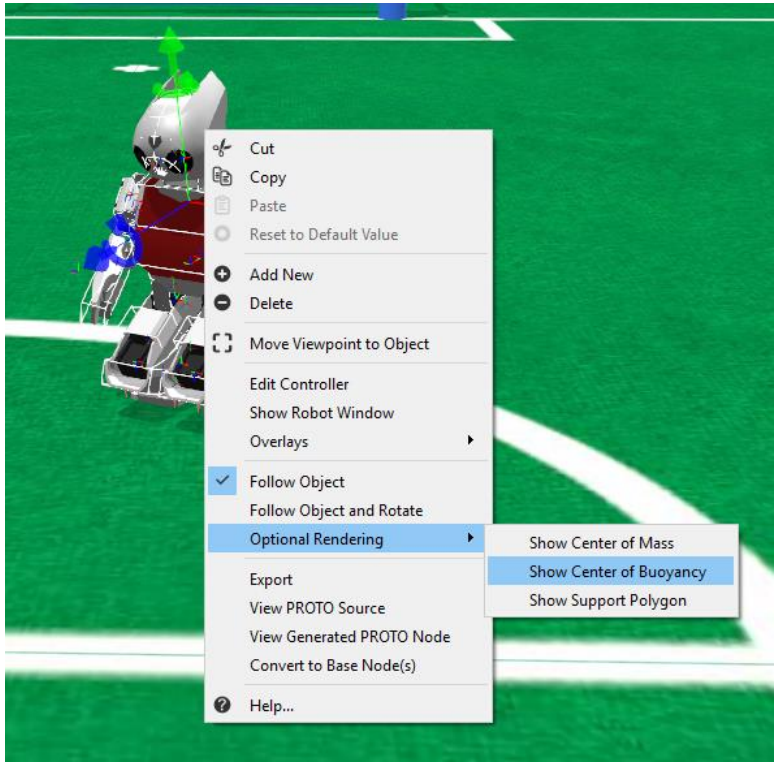
A single mouse left click allows you to select a Solid object.



Axis-aligned handles to move solid objects

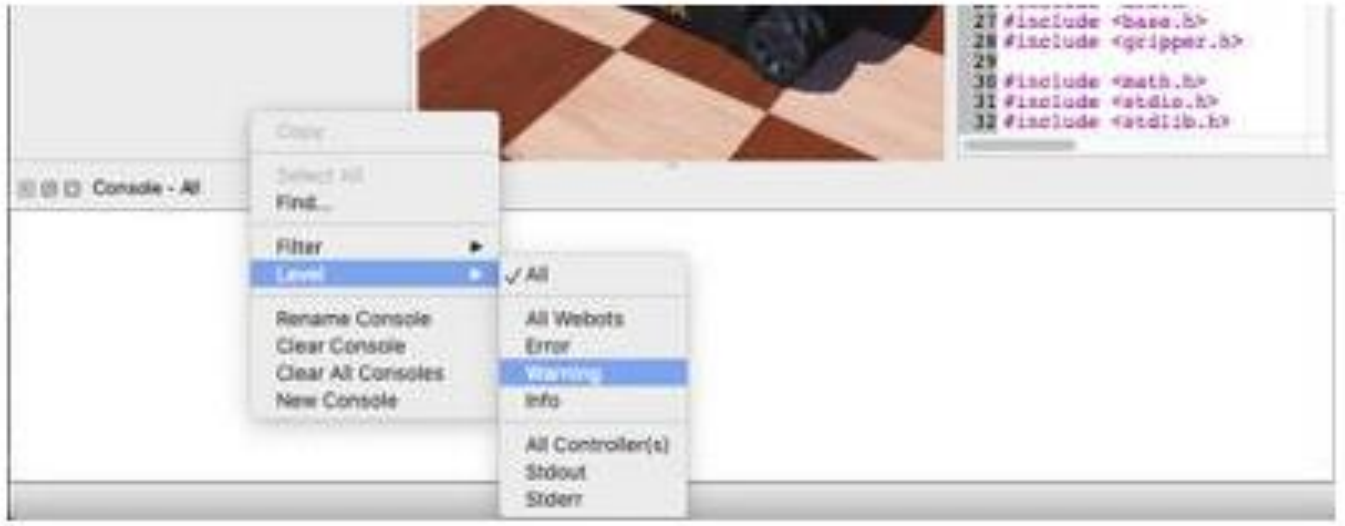


Labels displaying relative translation and rotation when moving objects with handles



The Console

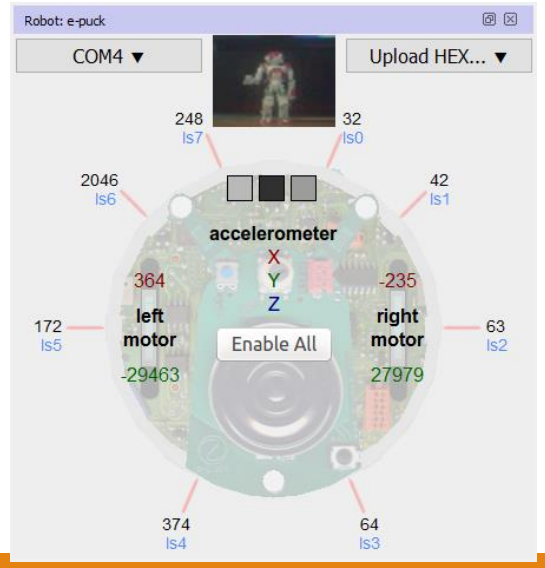
The Webots console displays the logs coming from Webots (such as parsing warnings or errors when opening a world, compilation outputs, ODE warnings, etc.) and the outputs of the controllers.



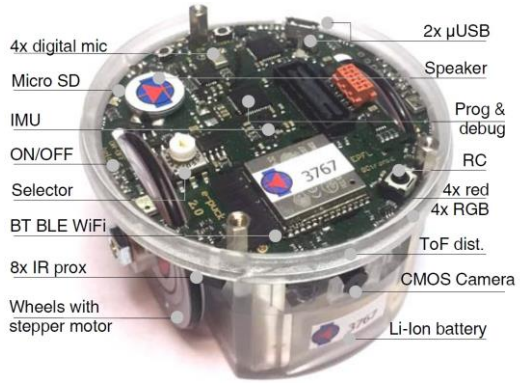
Console Window

E-puck robot

E-puck is a miniature mobile robot originally developed at EPFL for teaching purposes by the designers of the successful Khepera robot. The hardware and software of e-puck is fully open source, providing low level access to every electronic device and offering unlimited extension possibilities.
The model includes support for the differential wheel motors (encoders are also simulated, as position sensors), the infra-red sensors for proximity and light measurements, the Accelerometer, the Camera, the 8 surrounding LEDs, the body and front LEDs, bluetooth communication (modeled using Emitter / Receiver devices) and ground sensors extension.



https://youtu.be/tn9hHYVfqvQ



http://www.e-puck.org/

Lab: Tasks

Click on each task in below and follow the instructions from the webpage.

Show your outcome of each task to TAs for grading.

1. Your First Simulation in Webots (30 minutes)
2. Modification of the Environment (30 minutes)
3. Robot Controllers (30 minutes)

For Homework Assignment (No Grading)

- Read the user guide
  https://www.cyberbotics.com/doc/guide/index
- Do the following tutorial

https://www.cyberbotics.com/doc/guide/tutorial-3-appearance

https://www.cyberbotics.com/doc/guide/tutorial-5-compound-solid-and-physics-attributes

Reference

Introduction to Robotics (http://www.lacapnm.com/Cadets/STEM/Robotics/Introduction.pdf)

https://www.javatpoint.com/robotics-tutorial

https://www.britannica.com/technology/robot-technology

https://cyberbotics.com/doc/guide/introduction-to-webots