SoftBank Robotics

pepper

Technical Training

RobotLAB Choregraphe Training

An Introduction to Choregraphe and Pepper

Meet,

Get ready to explore your new robot!

Our agenda:

- Getting started with Pepper, Introduction to the software, Pepper's Software Architecture: Services, Boxes, Application Design, NAOqi Access Methods, Programming Basics, Build an Application, Sensing, Navigation, Object Detection, Face Detection

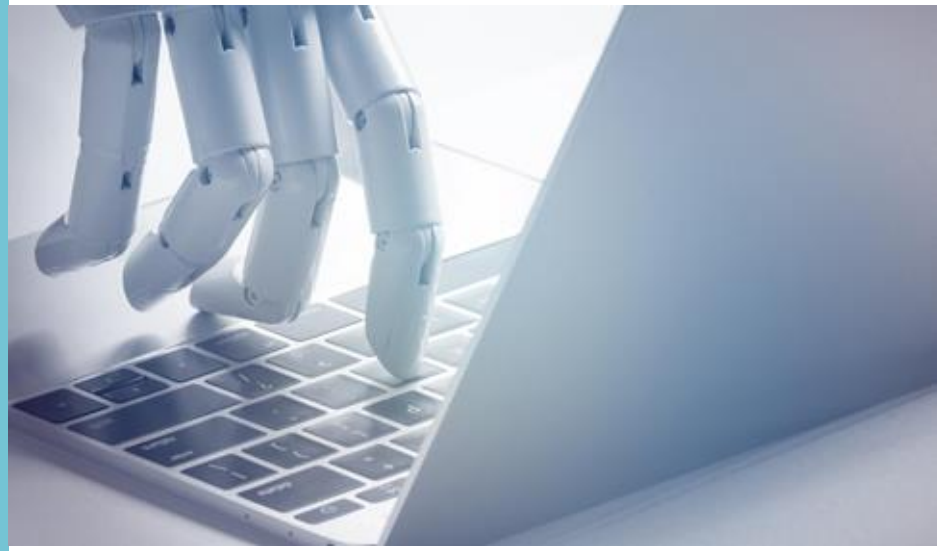
Learning Objectives | Choregraphe

So you want to bring your robot to life...?

GOAL: Master building Pepper applications using Choregraphe!

- Connect to your robot
- Learn all types of "*boxes*"
- Understand box flow logic
- Application management for your robot

Table of Contents | Choregraphe



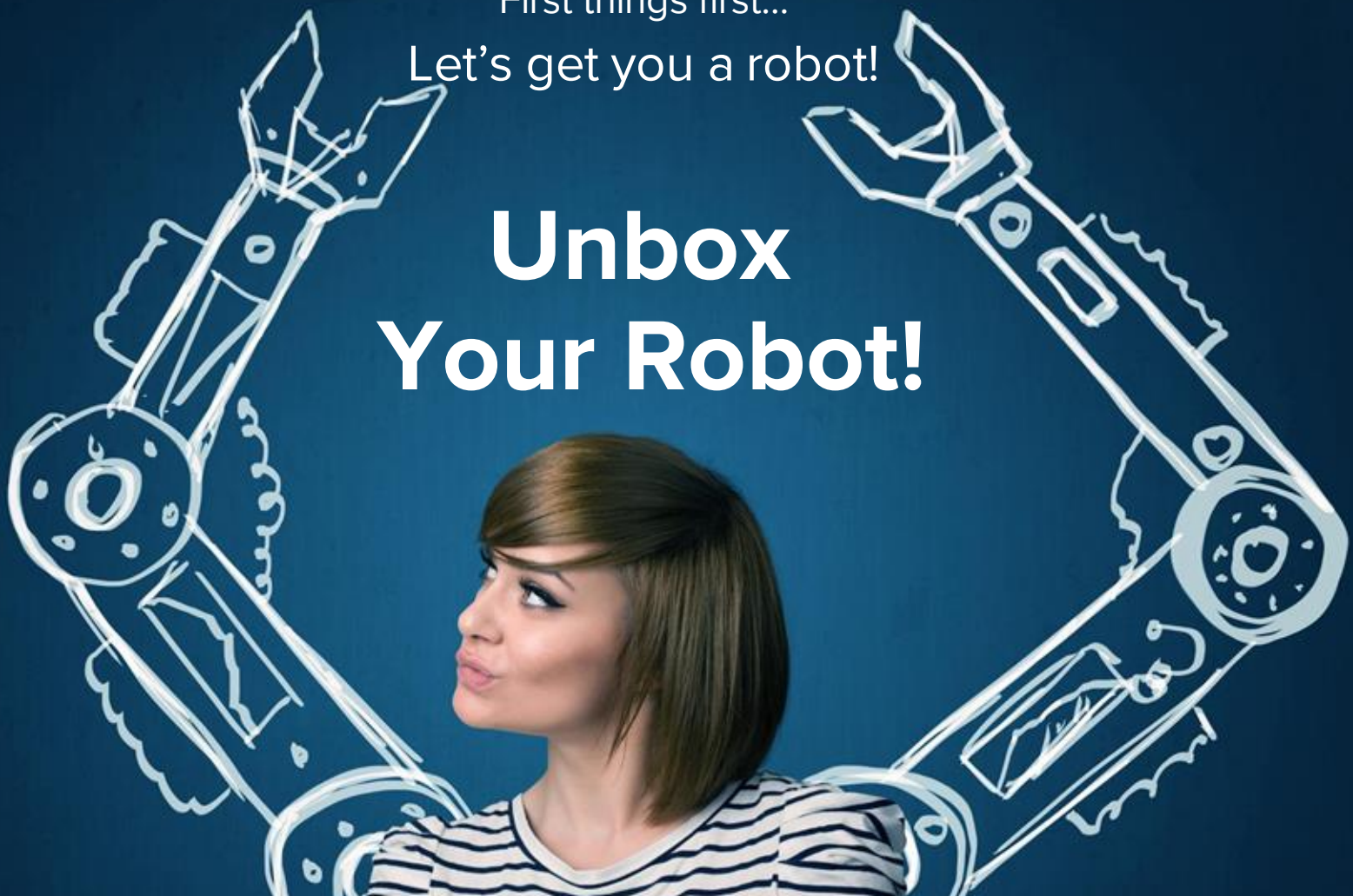
Overview - Topic I

Basic Orientation

First things first...

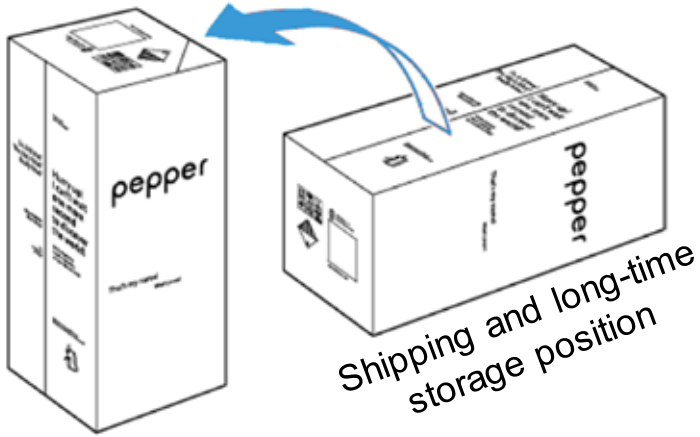
Let’s get you a robot!

Unbox Your Robot!



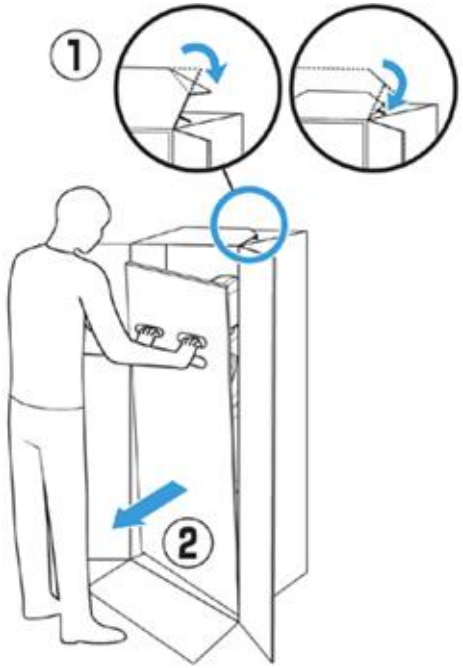
Opening the Box | Unbox your robot

1) Stand up the box



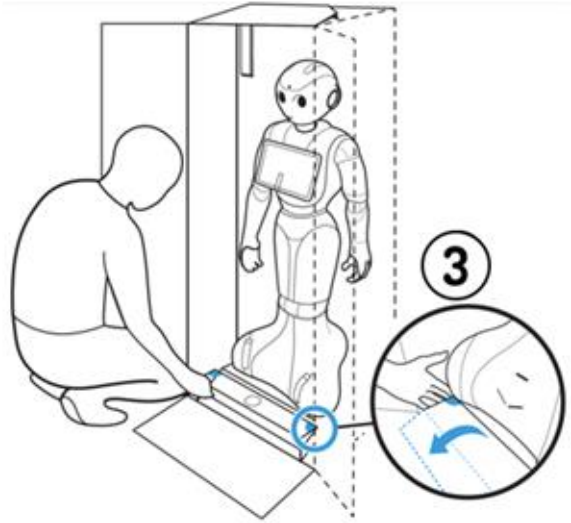
short-term storage position

2) Open & remove the cover



Pre-Removal Steps | Unbox your robot

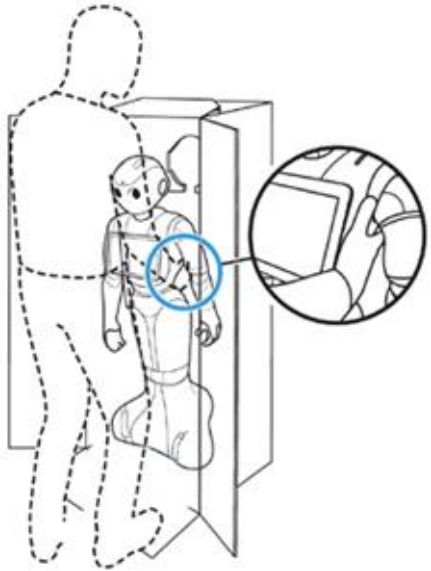
3) Flip the ramp open



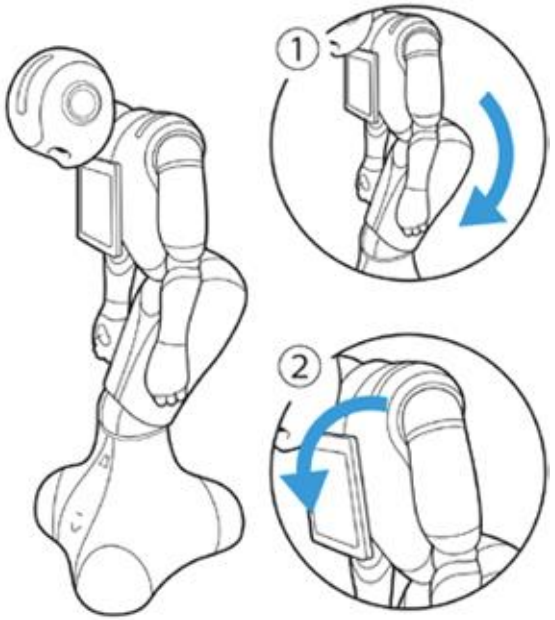
4) Pull head and arms out

Remove Pepper | Unbox your robot

5) Place your hands under Pepper's arms, then hold and pull the robot out of its box

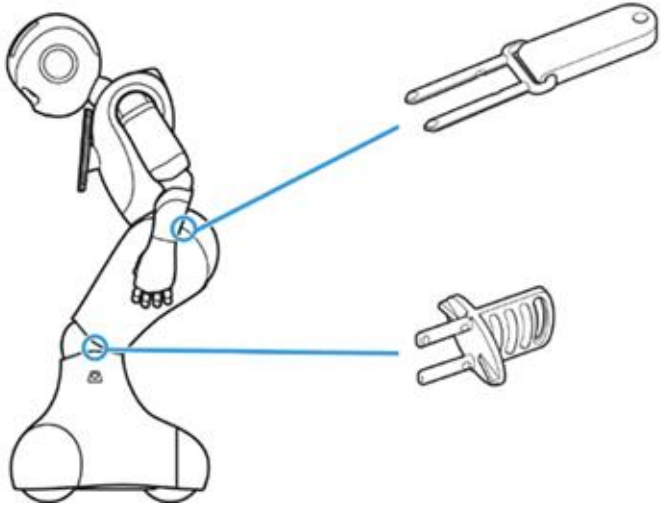


6) Place Pepper in the REST position

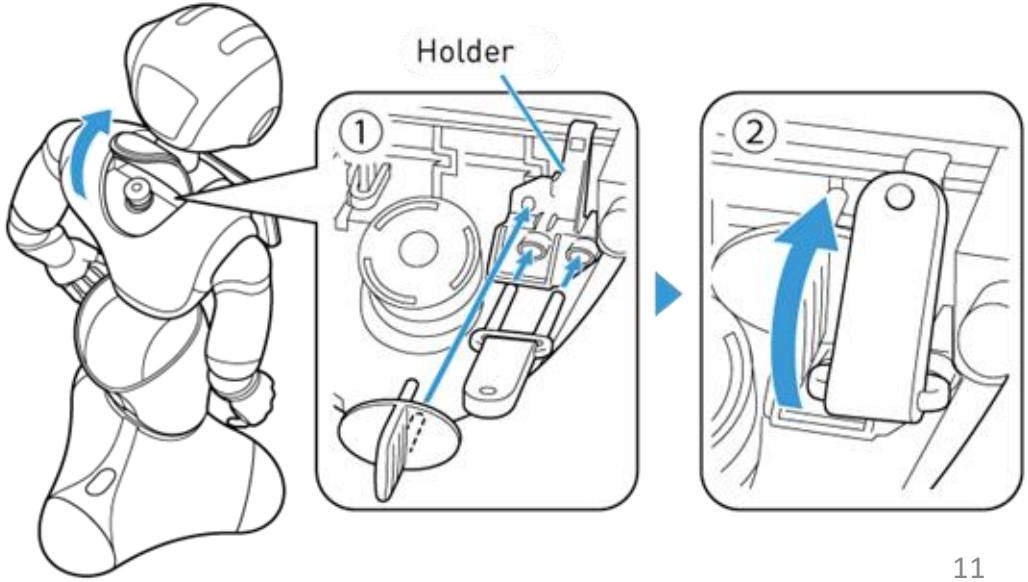


Remove the Pins | Unbox your robot

7) Remove the 2 pins

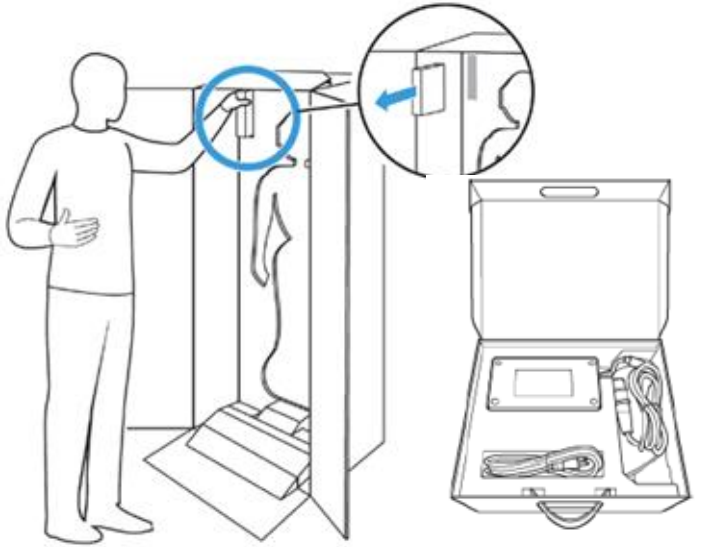


8) Open the soft cover behind the neck and store the pins



9) Unlock the emergency stop

10) Get the charger

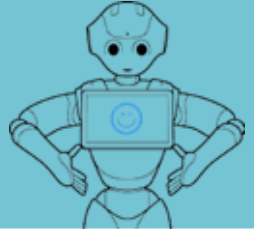


Accessory Storage | Unbox your robot

You should have left:

- The charger box**

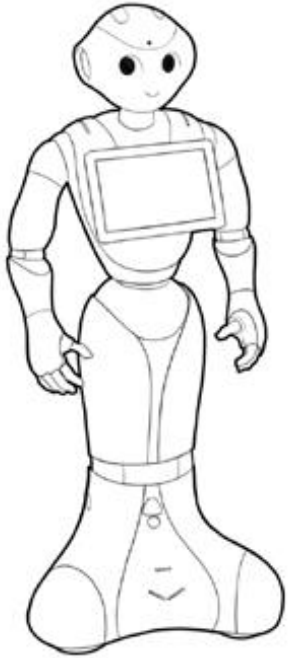
**Store this in the master box; you will need it whenever you store or ship your robot.



Postures

Standing: working posture

- Standing,
  arms along the body
- Pepper is awake
  and ready to use



Rest: safe posture

- Head down
- Knee and hip bent

Used when:

- Motors are off
- Rest mode
- Pepper is off

CONGRATULATIONS!

Basic Orientation - Topic I

Basic Interaction

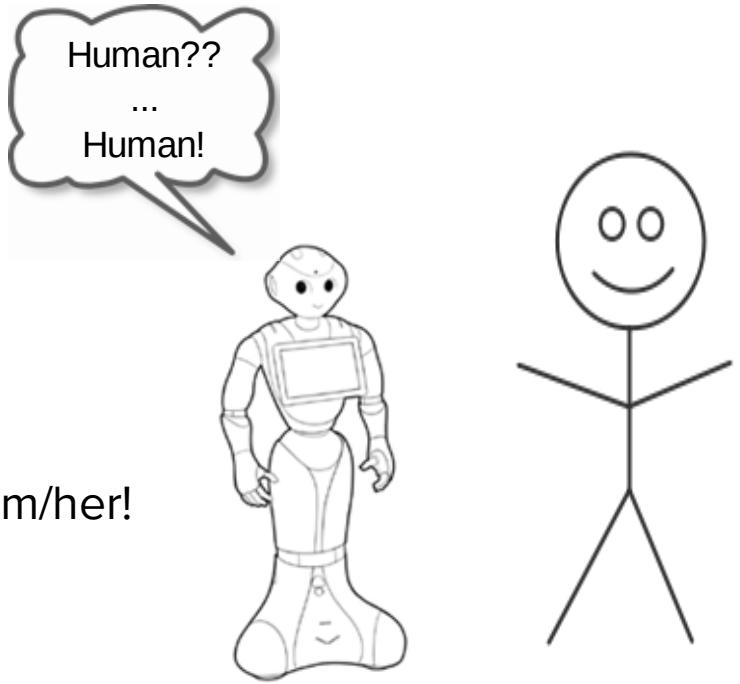
Pre-installed Software | Basic Interaction

Custom, pre-installed-for-many-use-cases software:

- “**Basic Awareness**”
  - App that automatically starts looking for a human
- “**The Dialog**”
  - App that starts automatically when the robot sees a human within its range

Basic awareness

- When your robot starts, it stands up and starts looking for people
- Pepper is now able to react to basic stimuli:
  - Sounds
  - Movements
  - Tactile contacts
  - Human Presence
- The goal is to find a human and interact with him/her!



The Dialog

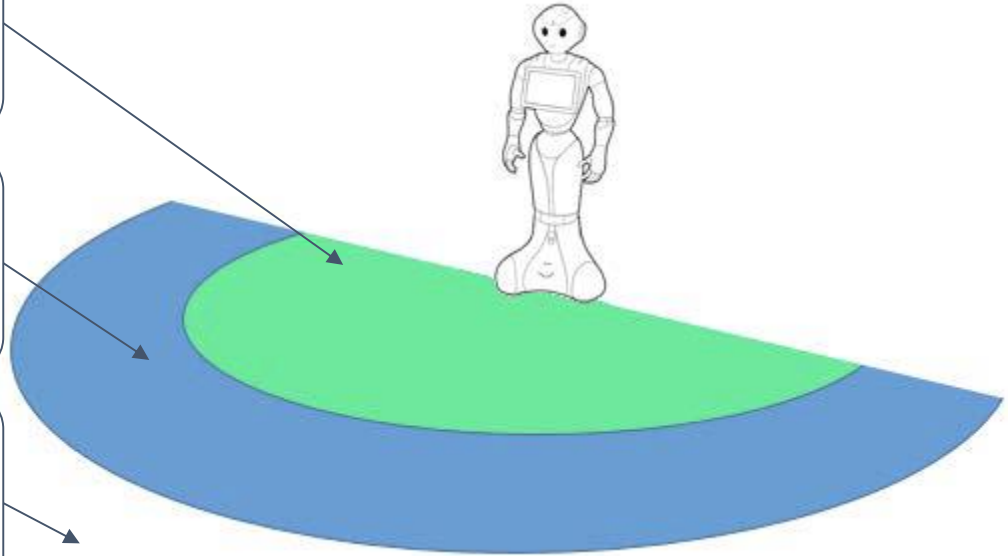
- Starts automatically when the robot sees a human
- Human must be close (in Zone 1 or 2)
- This app activates some “dialog topics” that you can talk about with the robot
- Basic channel: this is the set of “dialog topics”

Interaction zones:

ZONE 1
You are close enough to Pepper to have a conversation

ZONE 2
You are too far for a conversation, but you can hear Pepper calling you over!

ZONE 3
You are very far, Pepper sees you but you cannot hear each other.



Pepper's Eyes | Talking to Pepper

The eye color reveals Pepper's processing state:

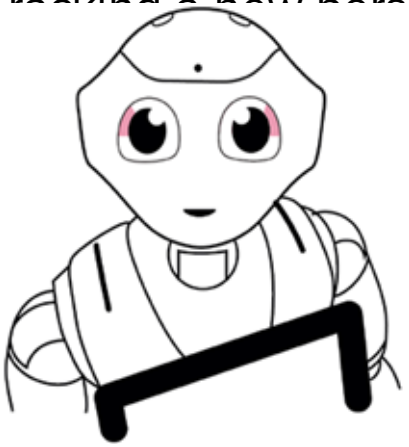
Pink:

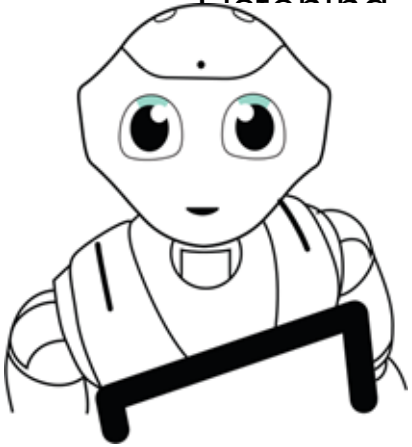
Blue (spinning):

Green:

Tracking a new person

Listening





What can I ask Pepper?

What's your name?

How are you?

What can you do?

What is your IP address?

What time is it?

Raise your arms.

How tall are you?

How much do you weigh?

How old are you?

Are you a boy or a girl?

What color are you?

Why is your name Pepper?

Do you have a family?

Animate The Robot

Bring Your Robot To Life

Animation Library | Animate The Robot

In the animation library, there are 200+ movements available.

Use them in a dialog:

^run	-	starts and blocks until the movement is finished
^start	-	starts and continues while the movement is playing
^wait	-	blocks until the movement is finished (use it after a ^start)
^stop	-	stops a movement

```
u:(hello) ^run(animations/Stand/Gestures/Hey_1) Hey Jonas!
```

Create A Movement | Animate The Robot

If you need more animations, you need to create them!

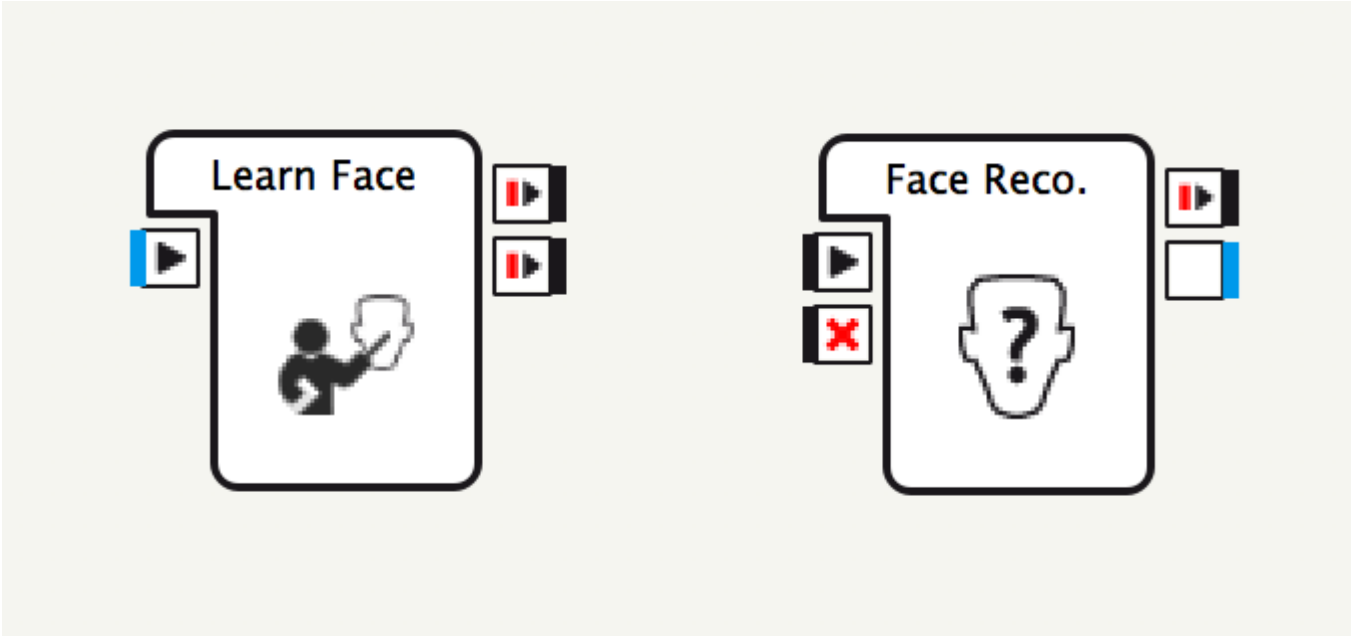
1) Create a new box “timeline”
2)  Disable autonomous life - Pepper goes into Rest mode
3)  Wake up the robot - Only “wakes up” the motors, not AL
4)  Activate “animation mode”
5) Touch the hand to move the arm; store the position by tapping the head

How Pepper handles **obstructions**:

- When an obstacle is detected, Pepper will crop the movements
- Most obstacles are detected by the lasers (on the floor)

➔ A wire is an example of a difficult-to-detect obstacle and can therefore represent an issue

Overview | Face Recognition

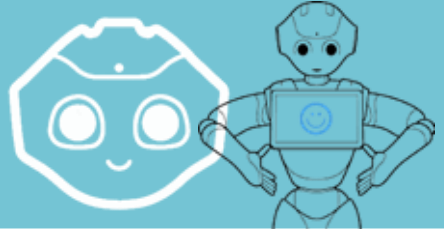


~ Challenge ~ | Face Recognition

Challenge: Have Pepper learn your face and then call you by name when it sees you.

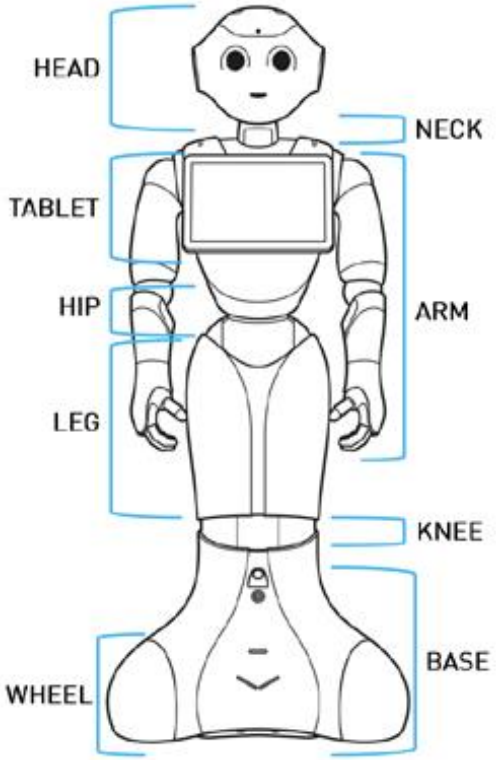
Hardware

Pepper's Anatomy

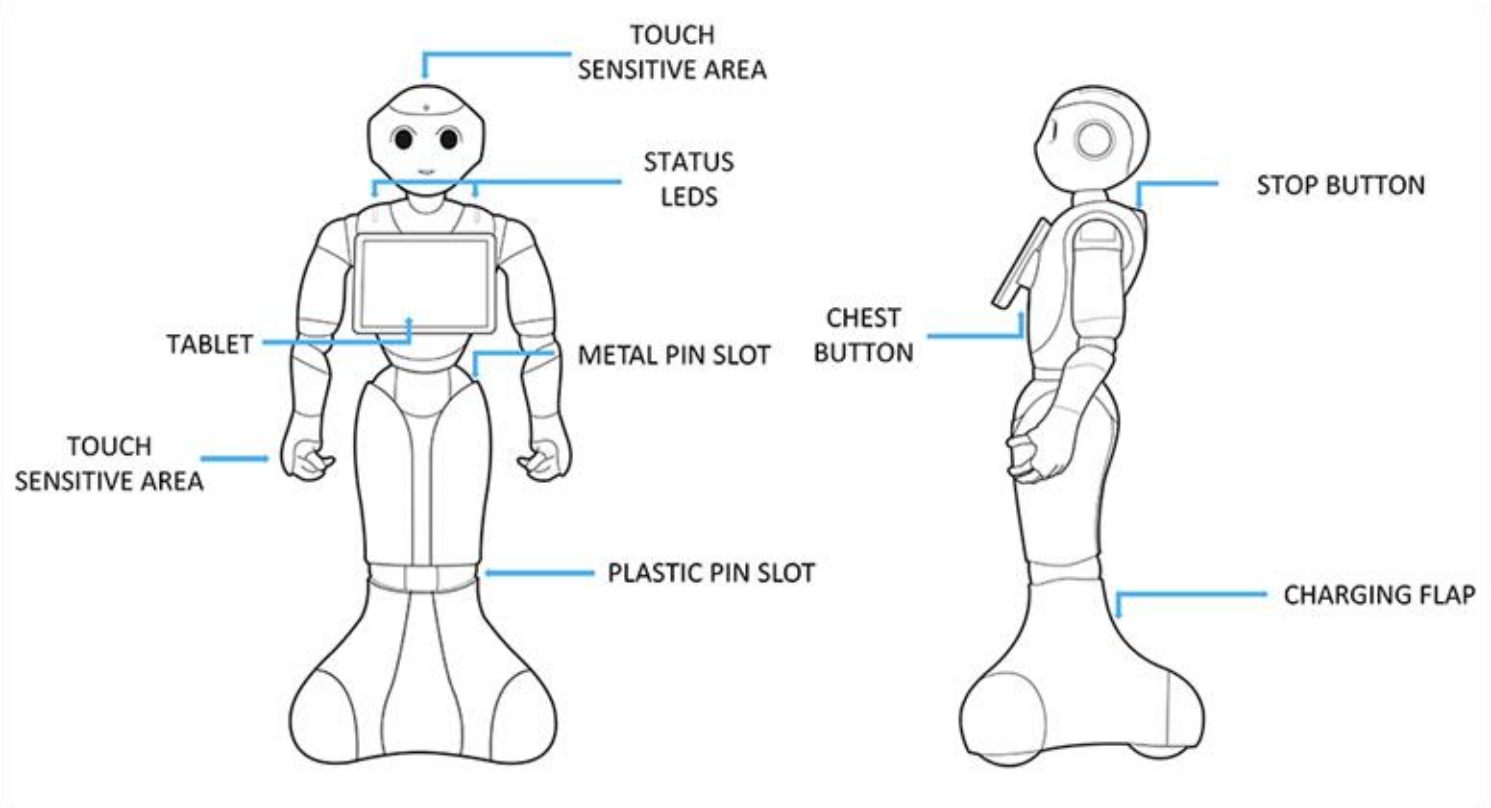


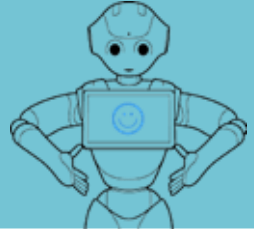
Key Sections | Pepper's Anatomy

The body is divided into several parts:

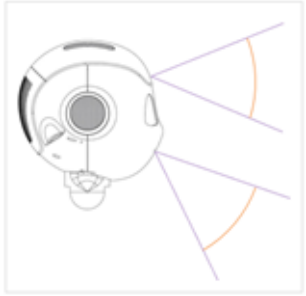


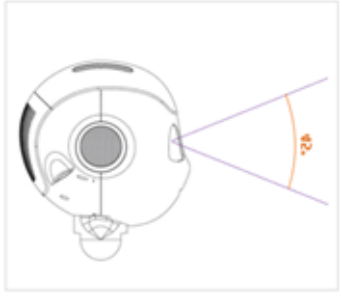
Areas of Interest | Pepper's Anatomy

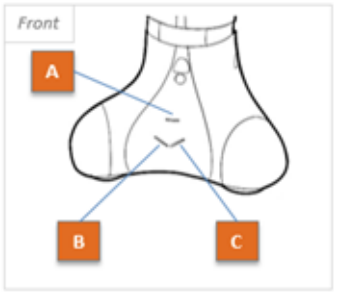


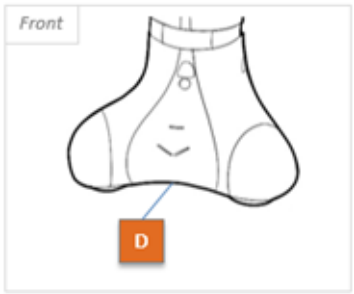


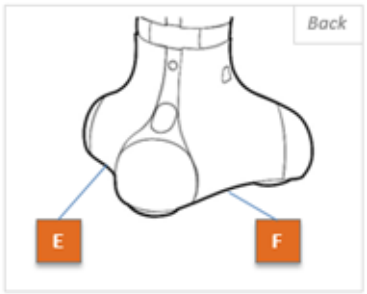
Sensors | Pepper's Anatomy

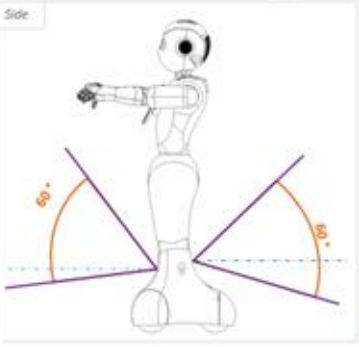


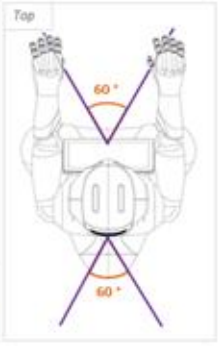






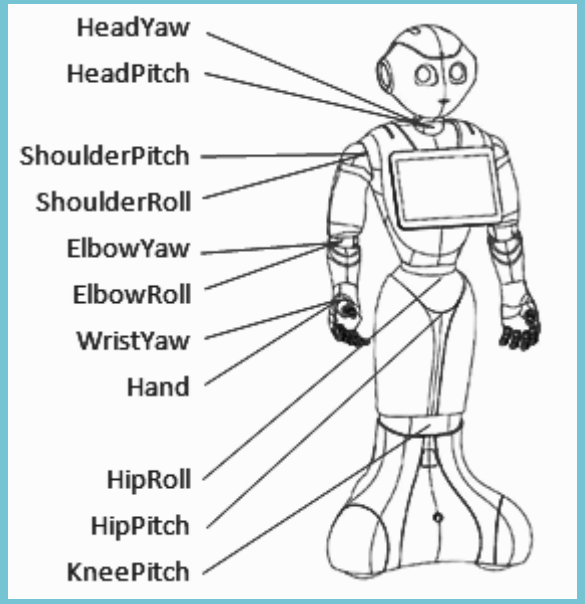




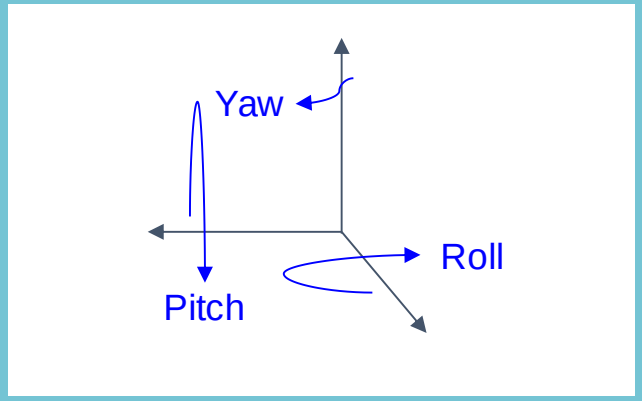


Motors (Actuators) | Pepper's Anatomy

Actuators



Motors are named after the joint and their direction

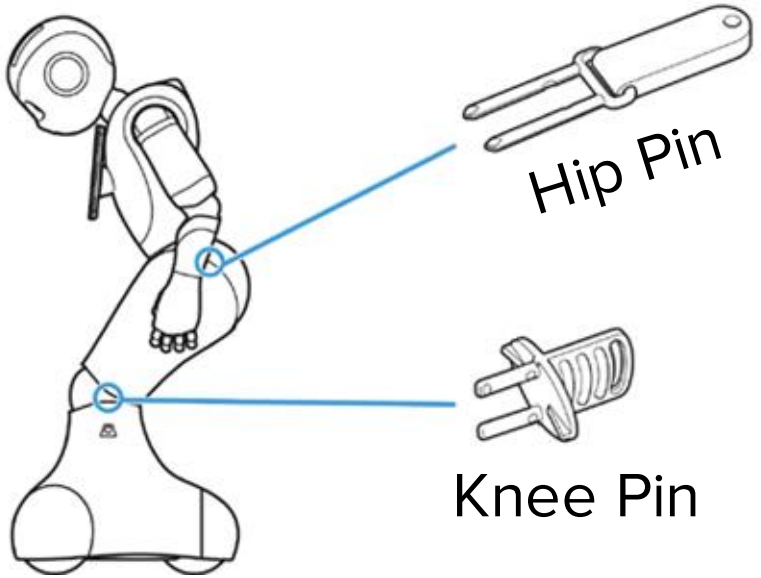


Brakes and Pins

The hip and knees have brakes to prevent Pepper from falling over.

Use the 2 pins to release the brakes:

- When you put Pepper in his box
- For manually setting Pepper's posture
- To move or carry Pepper



Charging the Battery | Pepper's Anatomy

To charge Pepper:

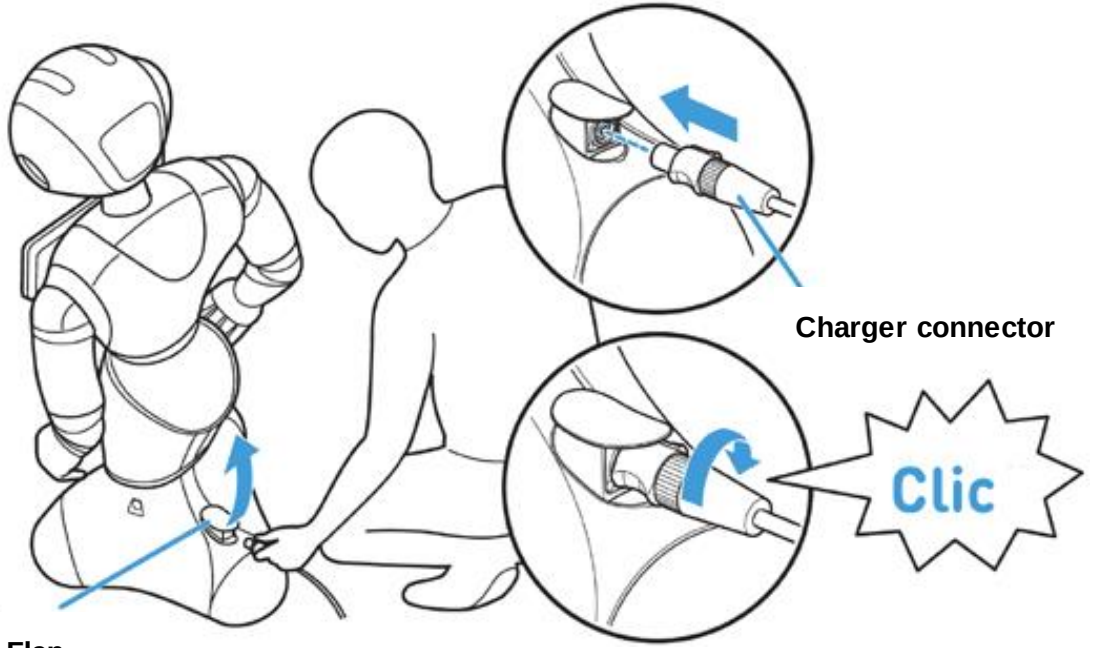
1. Open the charging flap
2. Insert charger connector
3. Turn connector to the right until it clicks

Charging Flap = Mobility Security

When the charging flap is opened, the wheels' motors are deactivated.
=> Open if you don't want Pepper to move around but still want to use him

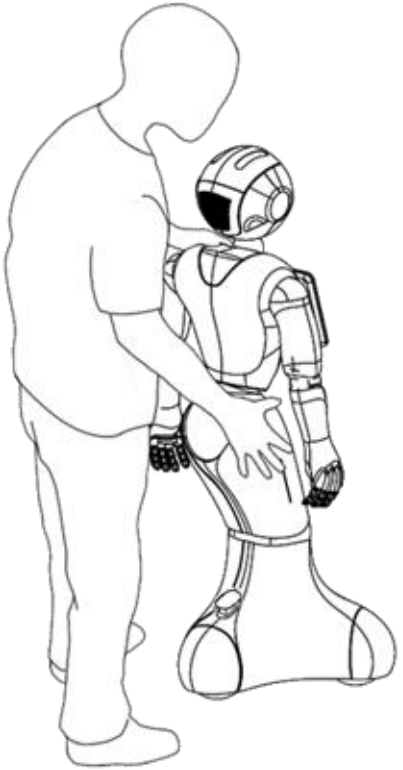
Charging duration:

- 80% in 3h30
- 100% in 8h



Moving your Robot | Pepper's Anatomy

1) Go to the Rest position
2) Make sure the charging flap is opened
3) Hold the robot
    a) One hand on the shoulder for steering**
    b) One hand on the hip for pushing
4) Move it carefully



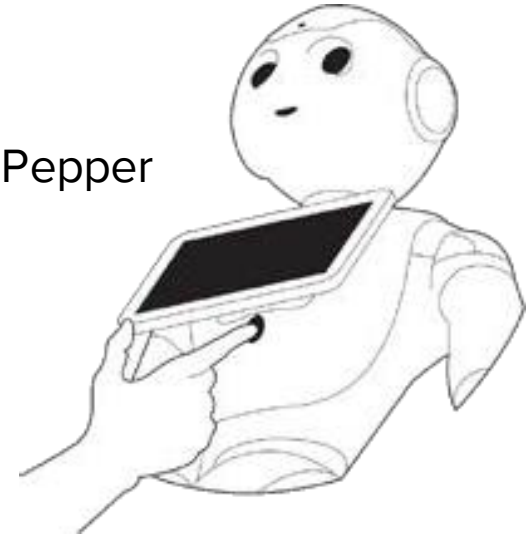
**Never push with the shoulder or Pepper will tilt forward!

Chest Button | Pepper's Anatomy

The chest button has multiple uses:

- When Pepper is **OFF**:
  - Press once: start Pepper
  - Press and hold: check the microcontrollers & start Pepper

- When Pepper is **ON**:
  - Press once: get status and notifications
  - Press twice: Rest / Wake up
  - Press and hold 3s: turn Pepper OFF
  - Press and hold 8s: force switch OFF

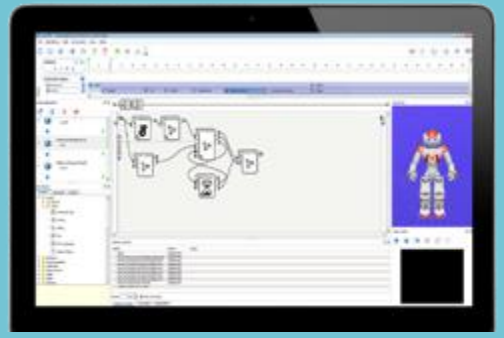


Robot has two connection options:

- Wi-Fi (both Head & Tablet)
- Ethernet

Use the hip pin to open the back of the head and reveal the Ethernet.

What is Choregraphe? | Choregraphe



Choregraphe

Easy visual prototyping tool

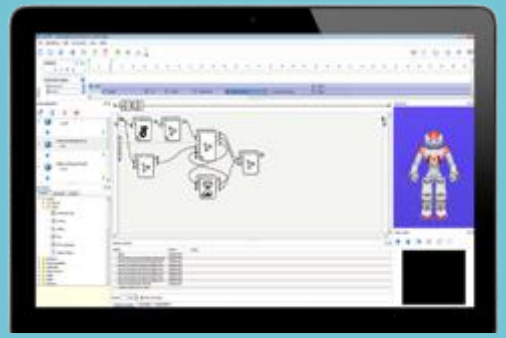
Monitor

Watch internal robot sensor data, logs

Software Development Kit

Comprehensive API for C++ or Python

Choregraphe vs SDK | Choregraphe



Choregraphe

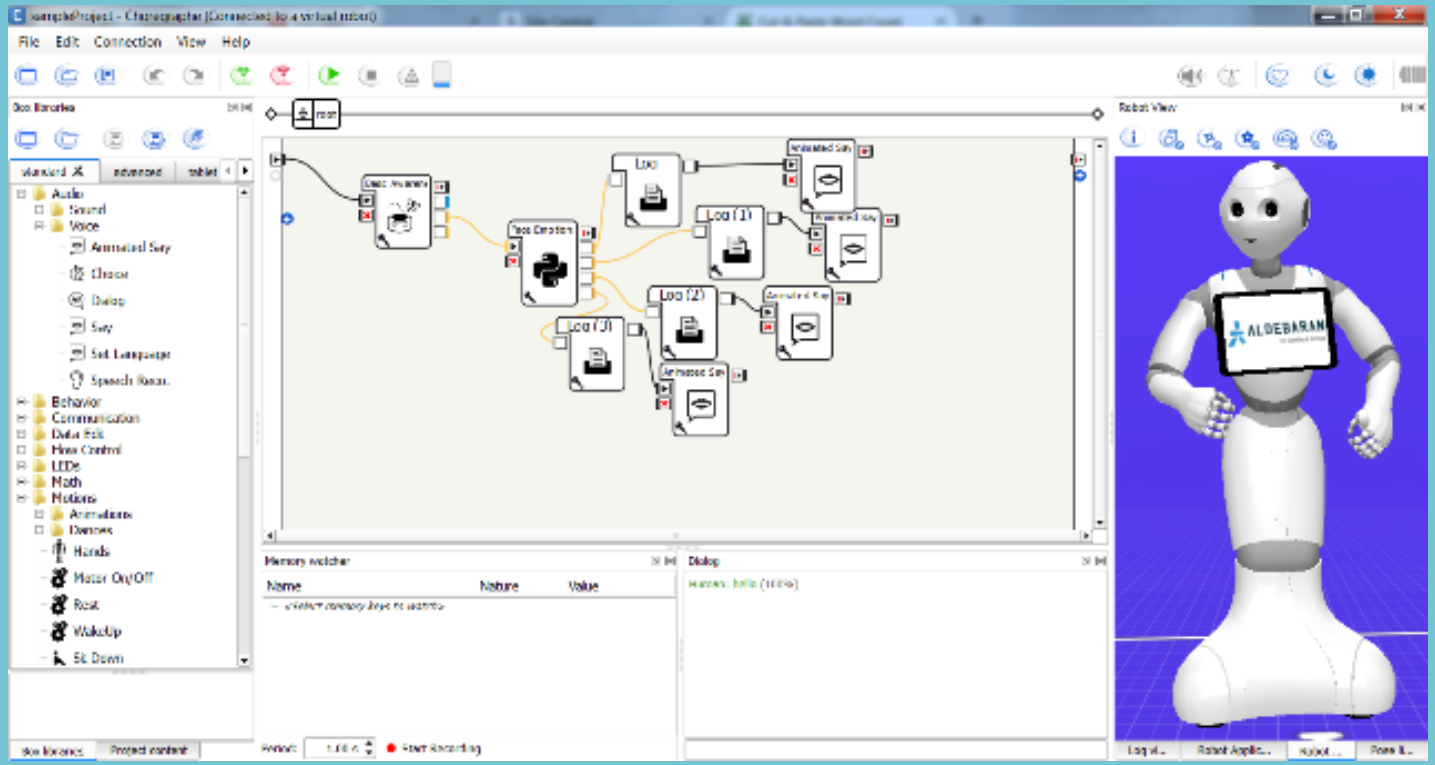
Easy visual prototyping tool

Vs

Software Development Kit

Comprehensive API for C++ or Python

Simple Apps = ✓✓ | Choregraphe

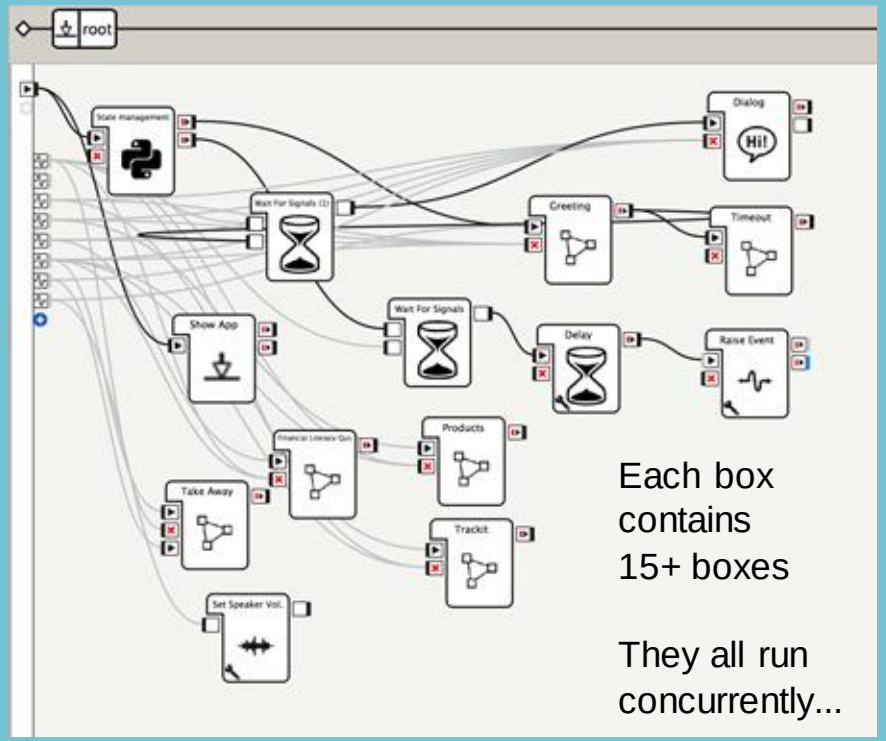


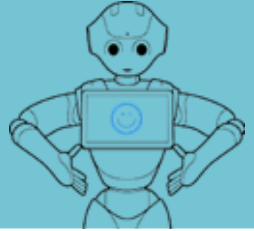
Production = Chaos! | Choregraphe

A “production” app is usually 2000+ lines of code… this can easily reach 50+ boxes, and 1000+ wires…

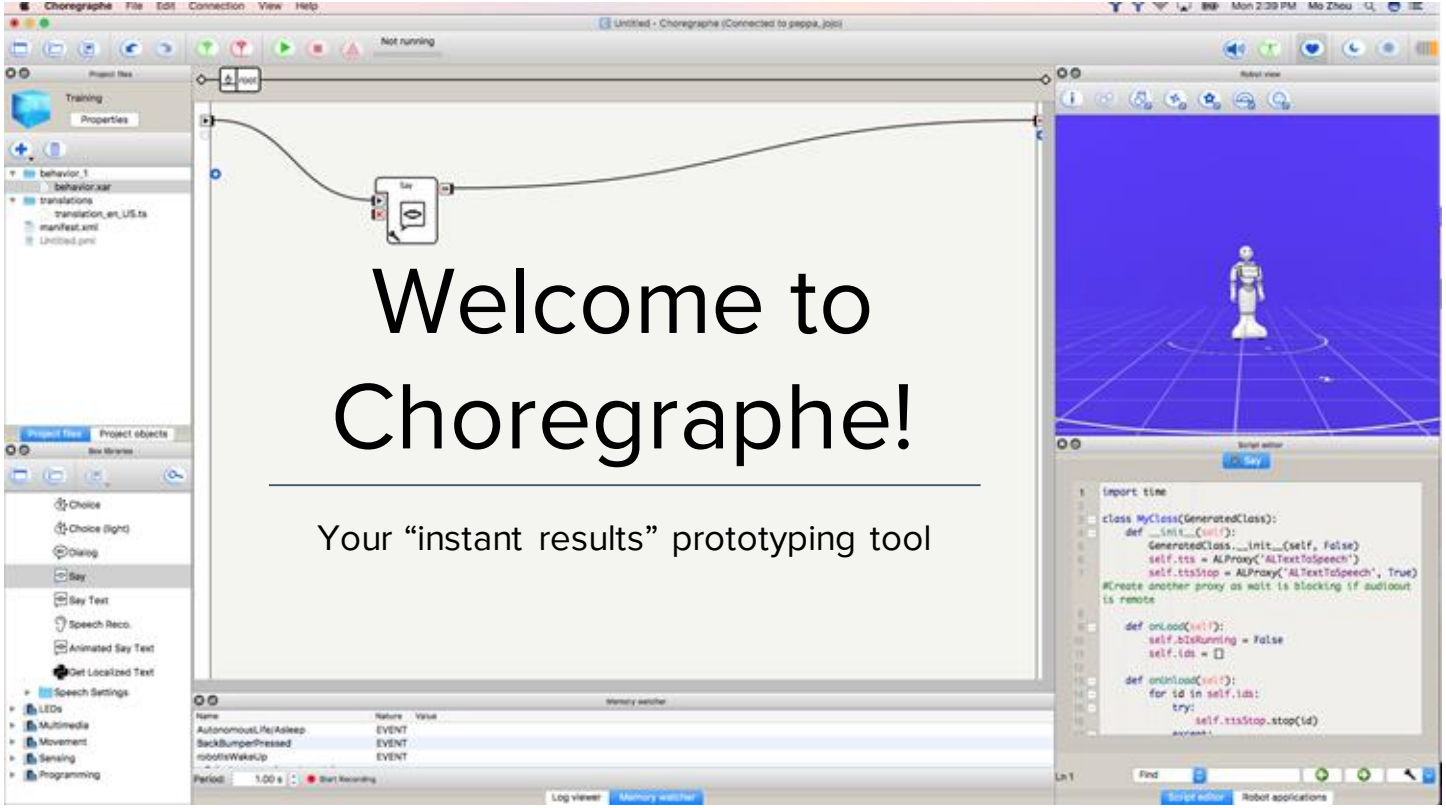
As boxes, it is:

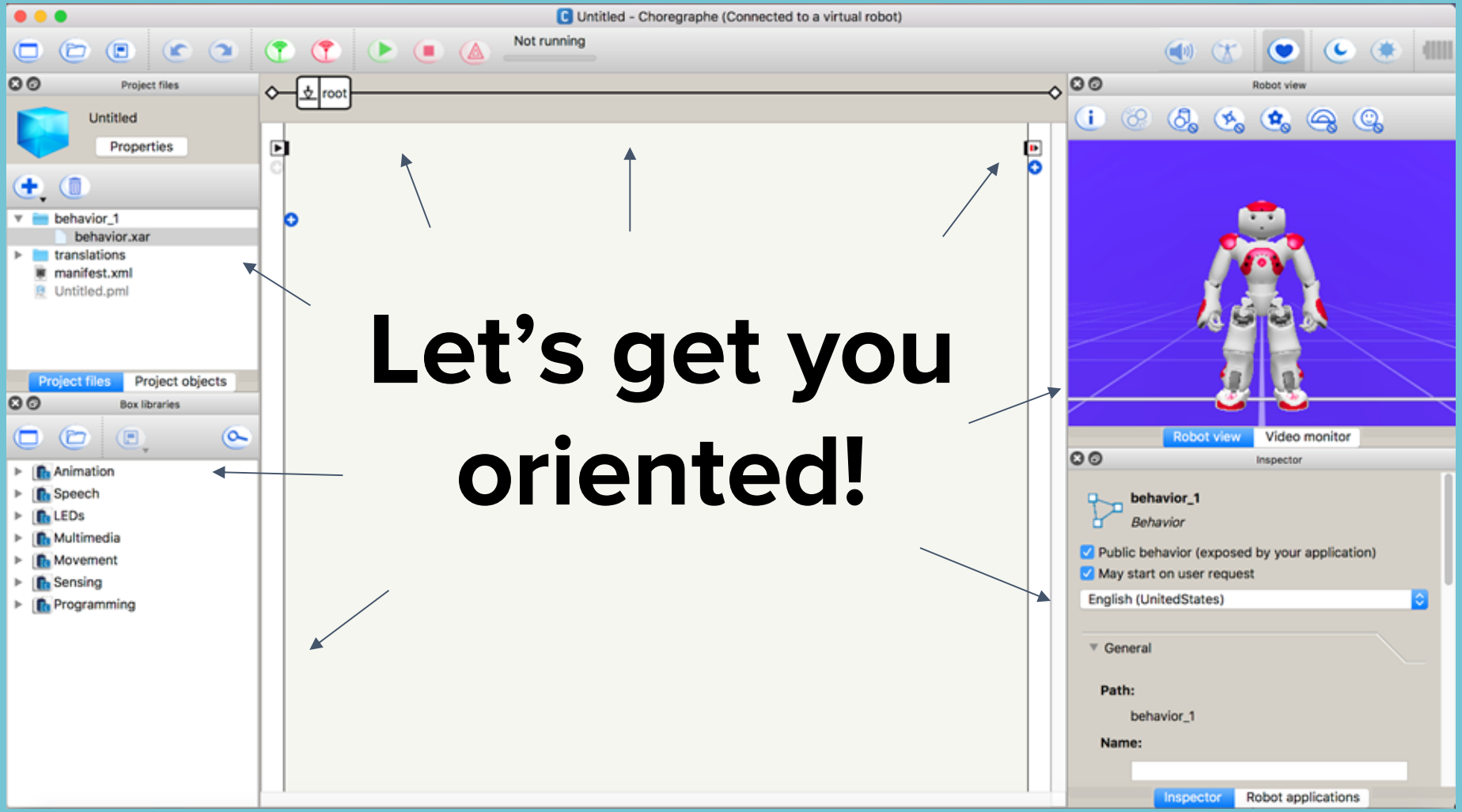
- HARD to write, share, version
- VERY HARD to organise, debug
- IMPOSSIBLE to understand, update

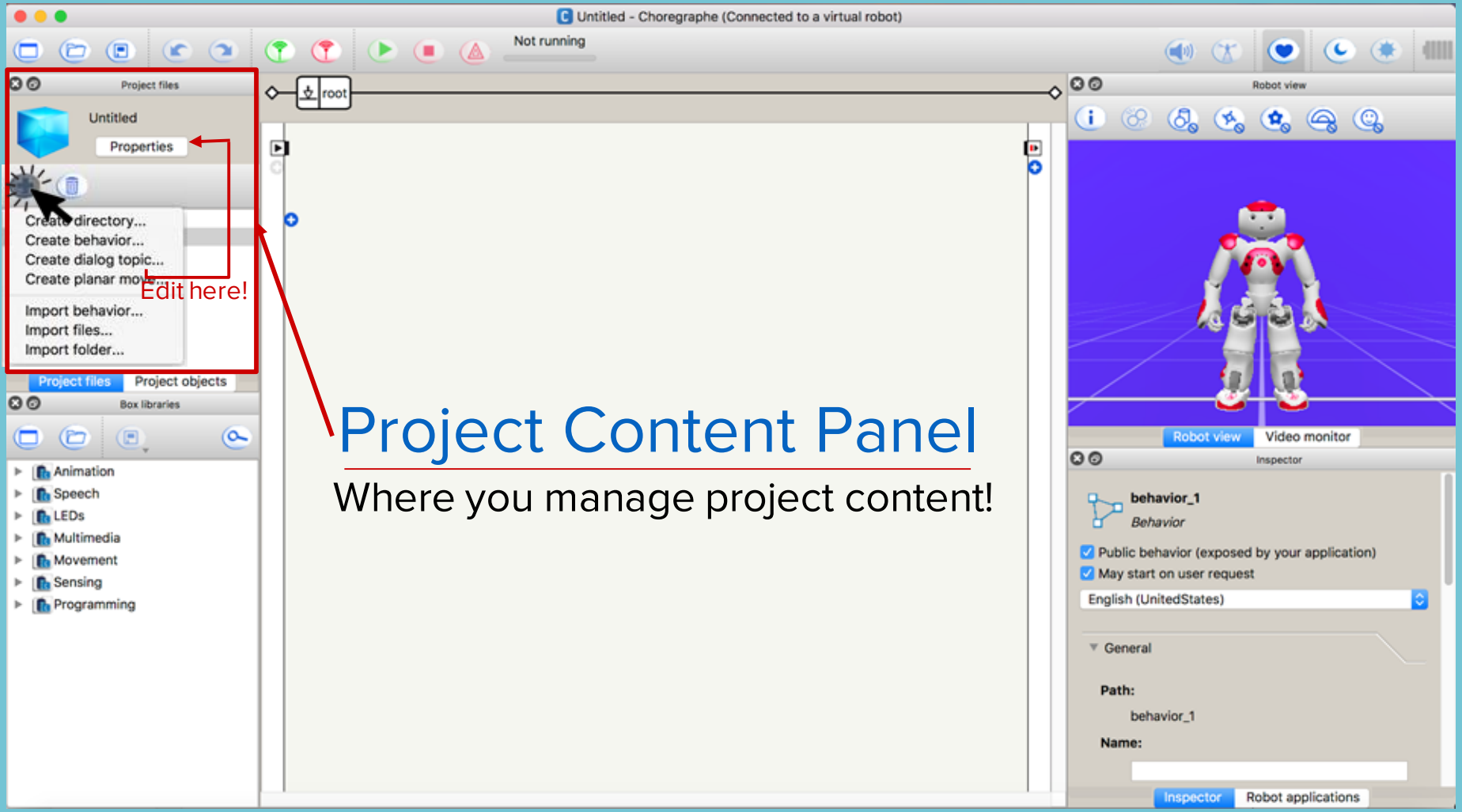


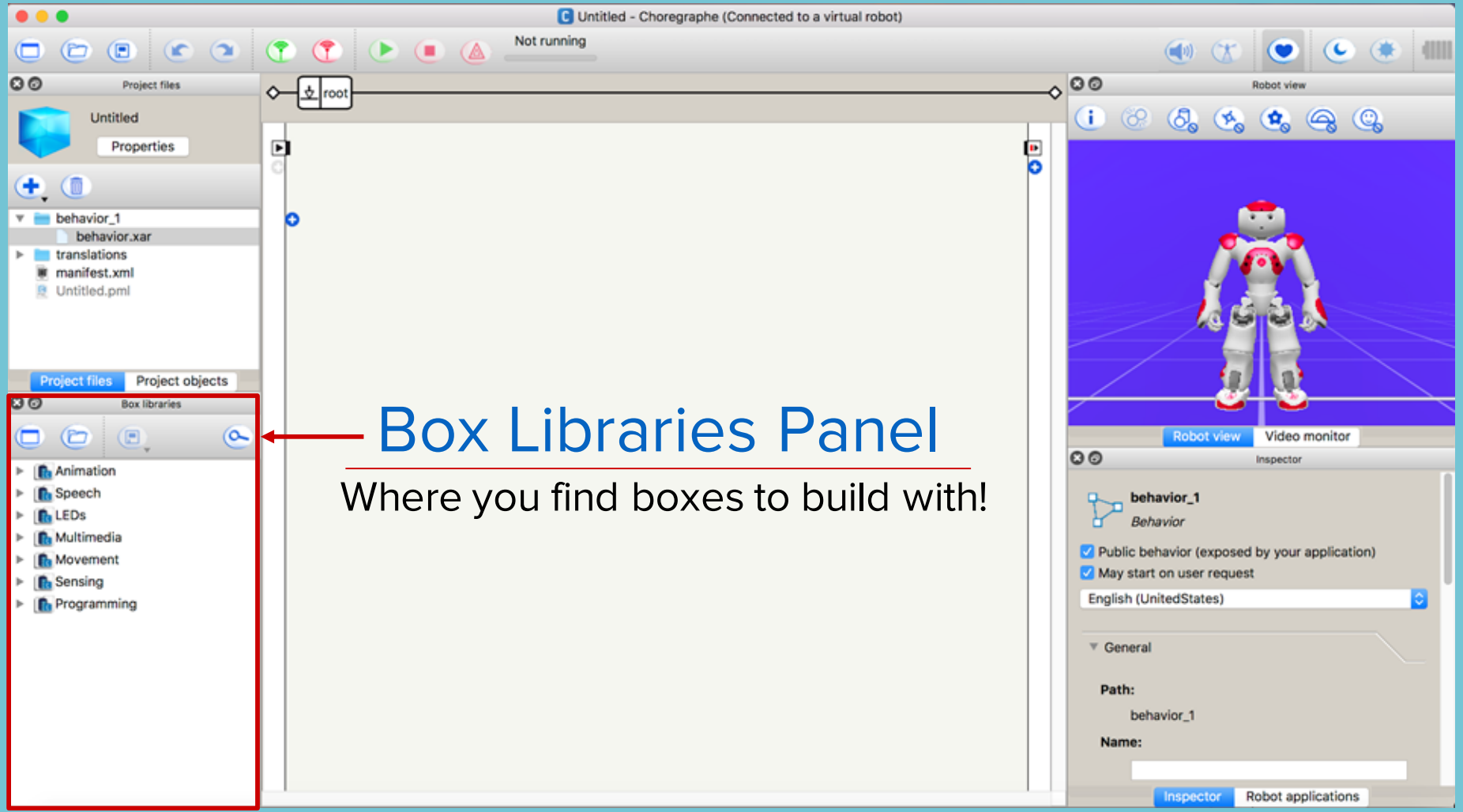


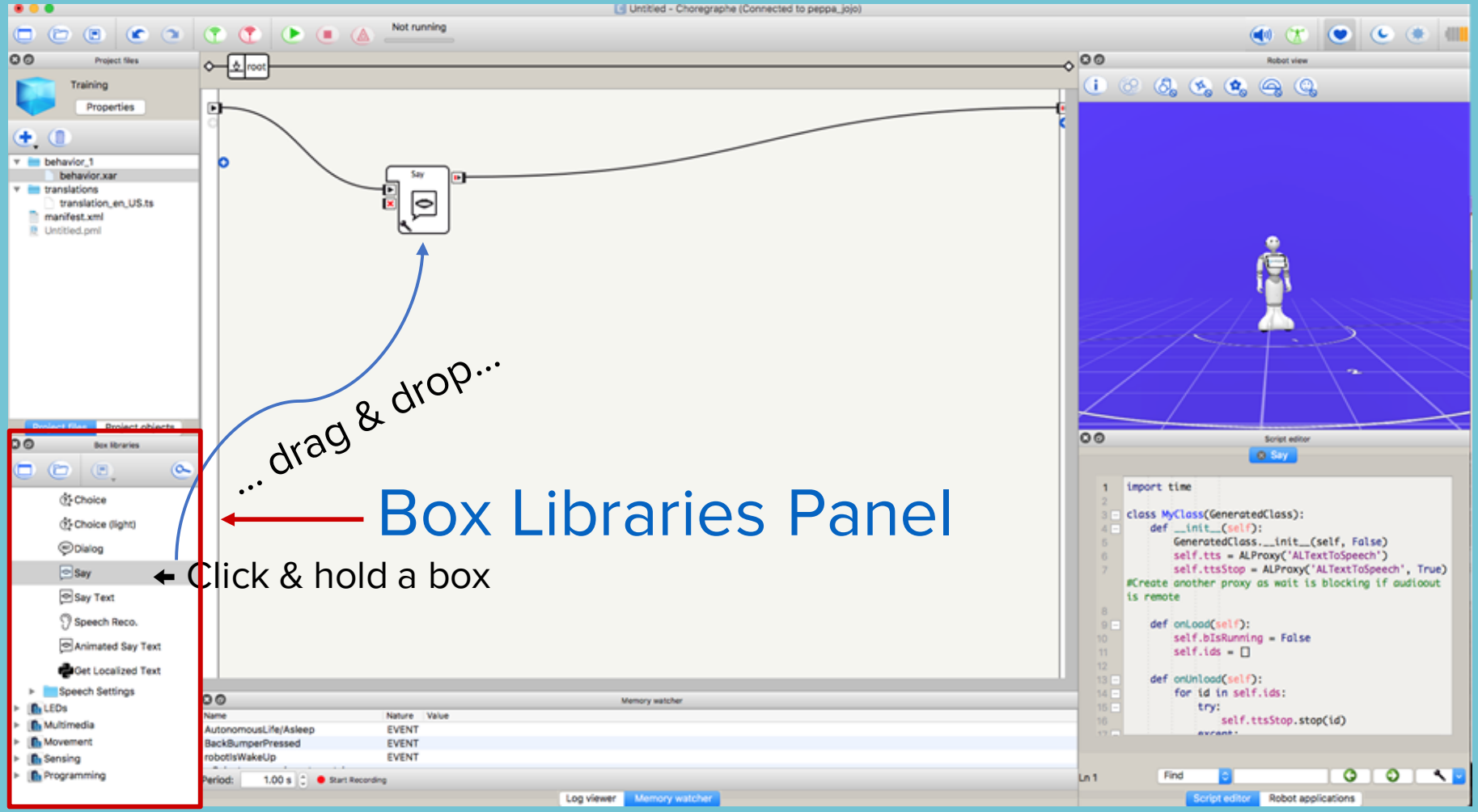
Tool Introduction | Choregraphe

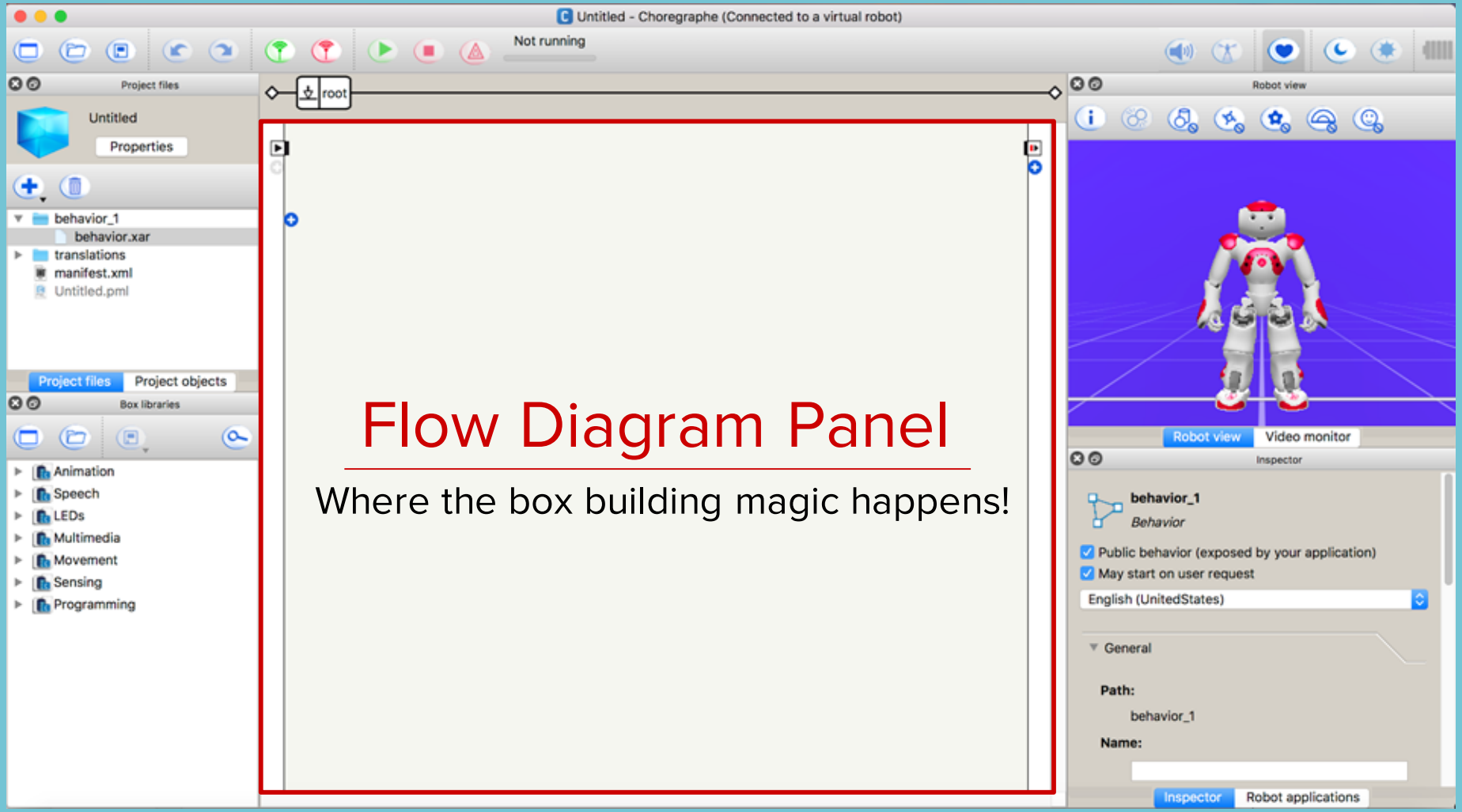


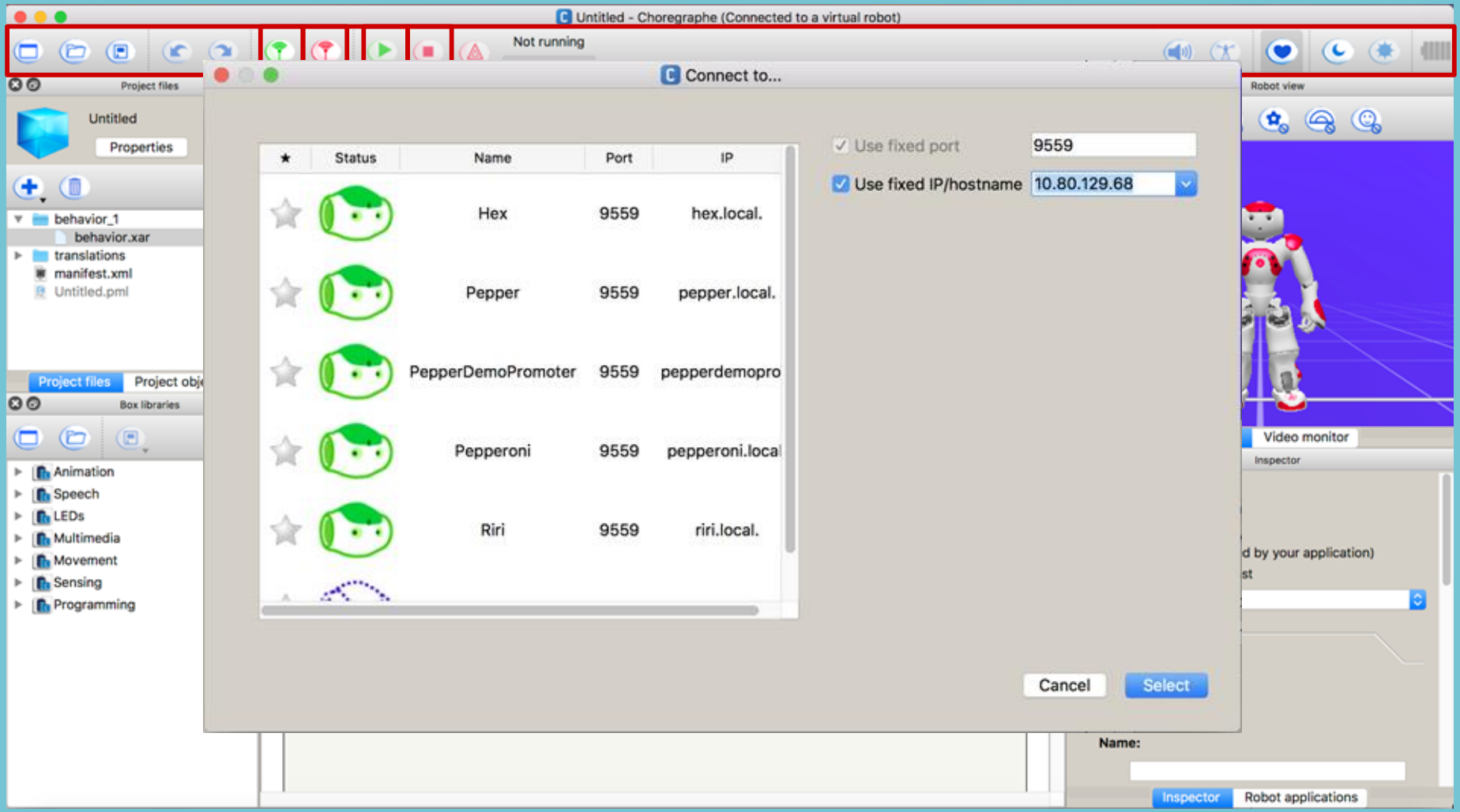

Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Let’s get you oriented!
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications

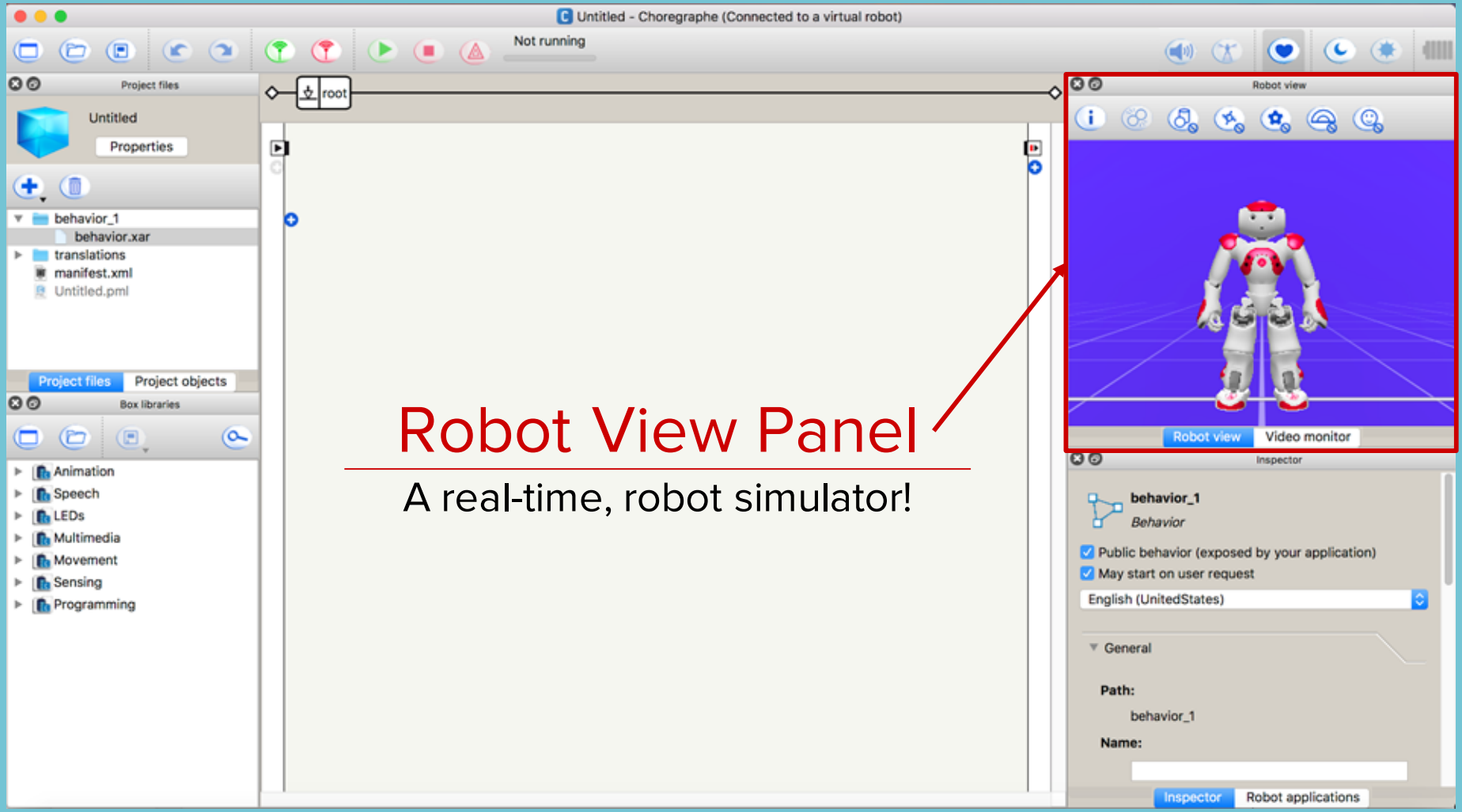

Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
Create directory...
Create behavior...
Create dialog topic...
Create planar move...
Import behavior...
Import files...
Import folder...
Edit here!
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Project Content Panel
Where you manage project content!
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications

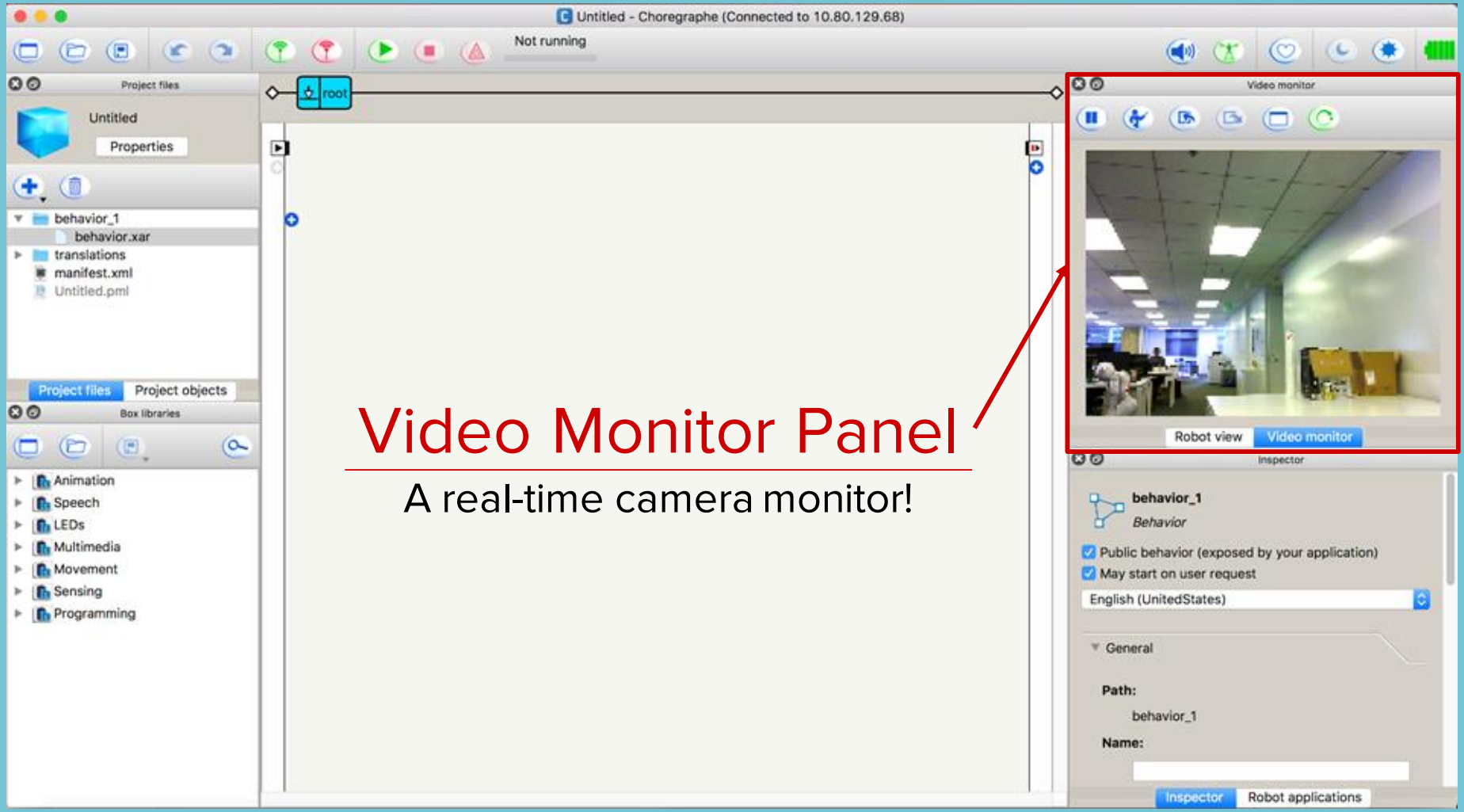

Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Box Libraries Panel
Where you find boxes to build with!
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications

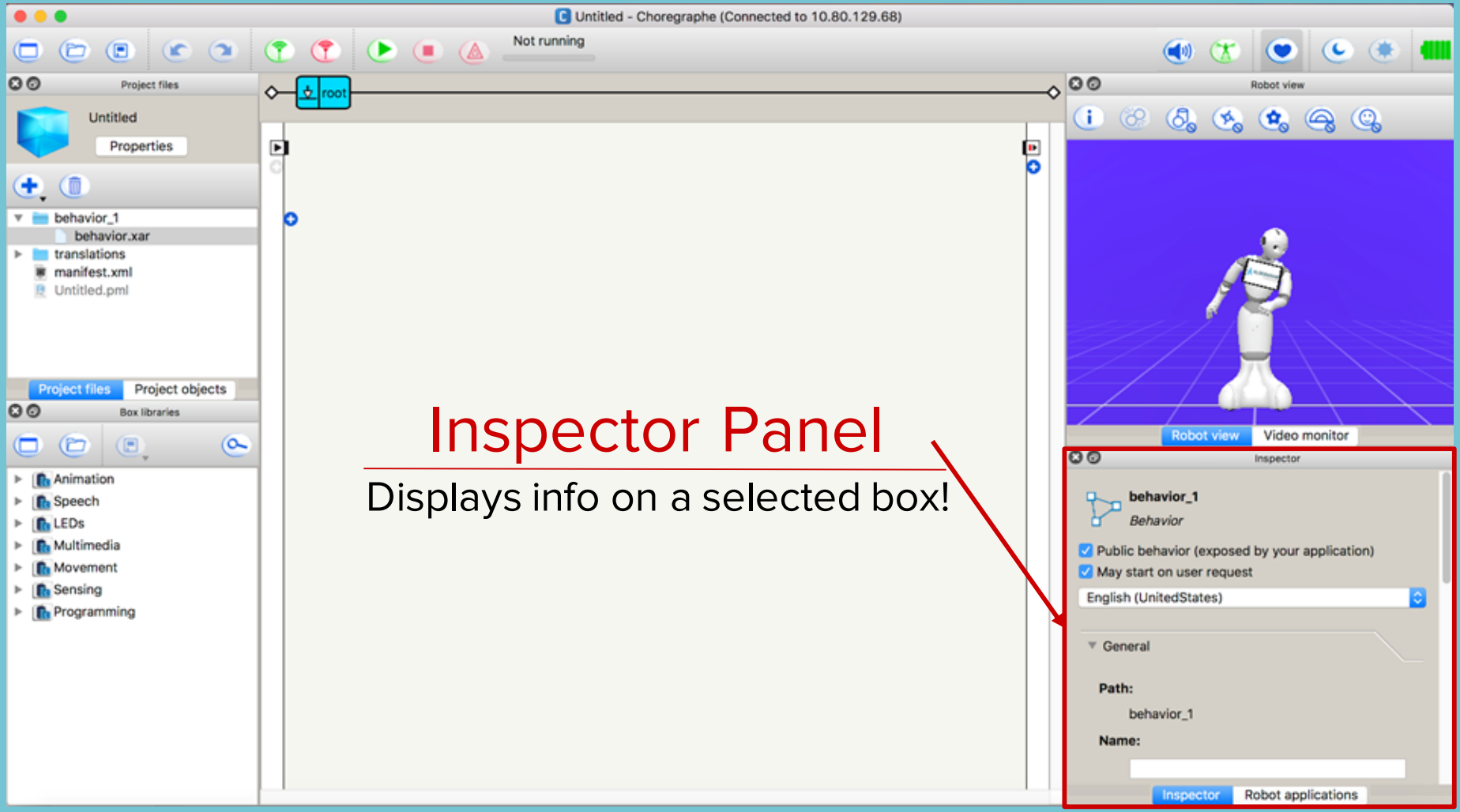

Untitled - Choregraphe (Connected to peppa_jojo)
Not running
Project files
Training
Properties
behavior_1
behavior.xar
translations
translation_en_US.ts
manifest.xml
Untitled.pml
root
Say
Robot view
Box libraries
Choice
Choice (light)
Dialog
Say
Say Text
Speech Reco.
Animated Say Text
Get Localized Text
Speech Settings
LEDs
Multimedia
Movement
Sensing
Programming
... drag & drop...
Box Libraries Panel
Click & hold a box
Memory watcher
Name
Nature
Value
AutonomousLife/Asleep
EVENT
BackBumperPressed
EVENT
robotIsWakeUp
EVENT
Period:
1.00 s
Start Recording
Log viewer
Memory watcher
Script editor
Say
import time
class MyClass(GeneratedClass):
def __init__(self):
GeneratedClass.__init__(self, False)
self.tts = ALProxy('ALTextToSpeech')
self.ttsStop = ALProxy('ALTextToSpeech', True)
#Create another proxy as wait is blocking if audioout is remote
def onLoad(self):
self.bIsRunning = False
self.ids = []
def onUnload(self):
for id in self.ids:
try:
self.ttsStop.stop(id)
Ln 1
Find
Script editor
Robot applications

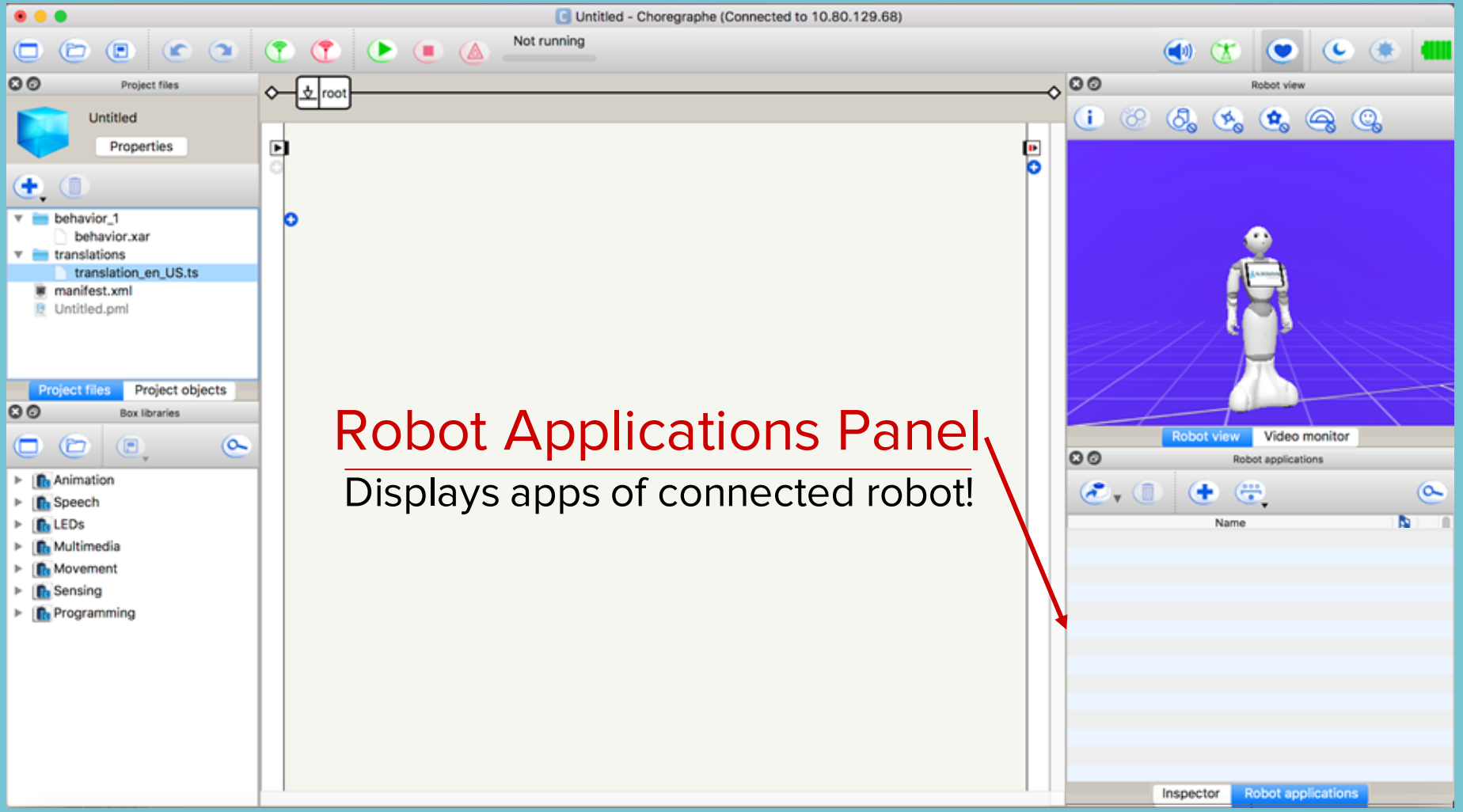

Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Flow Diagram Panel
Where the box building magic happens!
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications

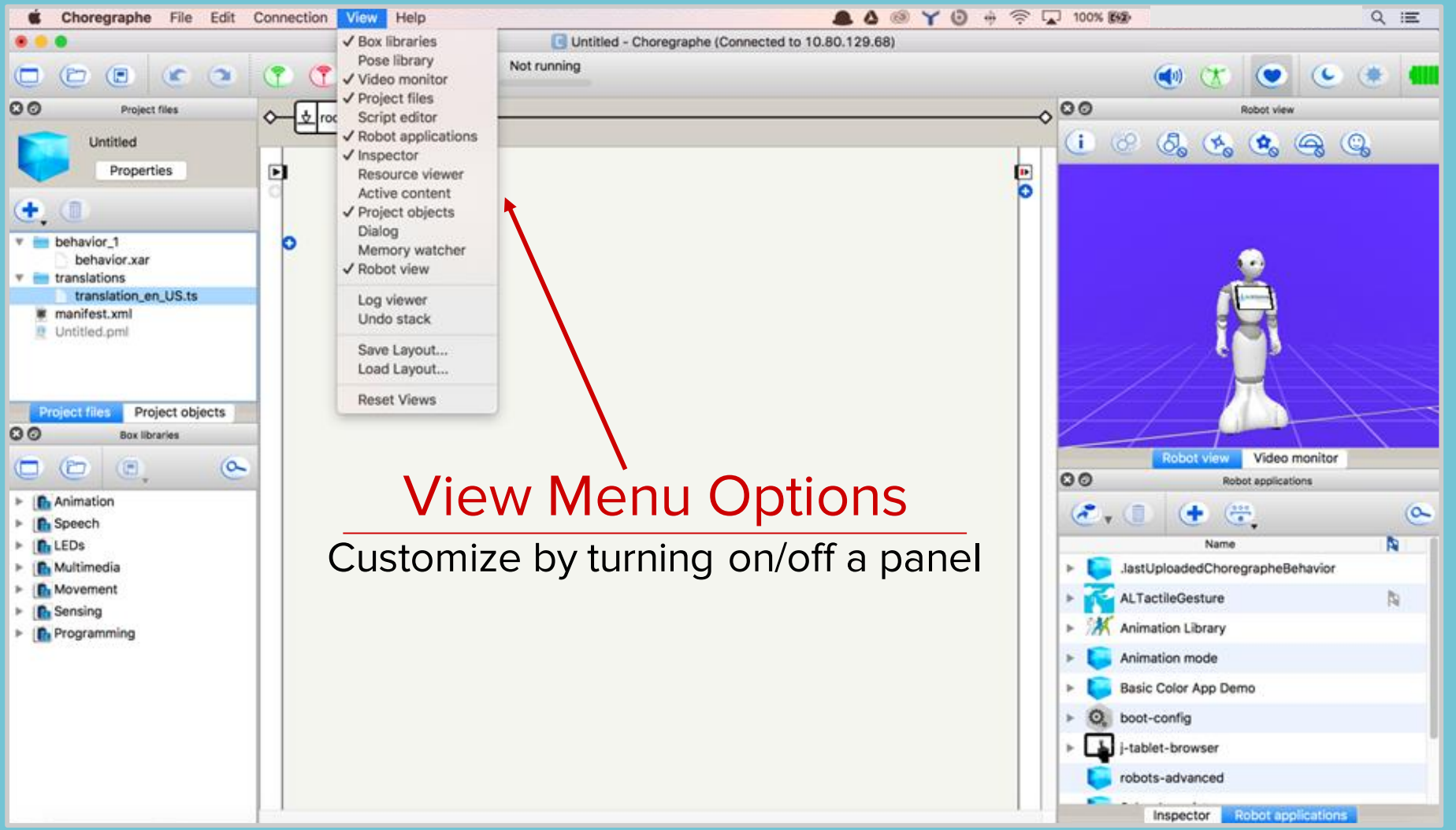

Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
Connect to...
★
Status
Name
Port
IP
Hex
9559
hex.local.
Pepper
9559
pepper.local.
PepperDemoPromoter
9559
pepperdemopro
Pepperoni
9559
pepperoni.local
Riri
9559
riri.local.
Use fixed port
9559
Use fixed IP/hostname
10.80.129.68
Cancel
Select
Robot view
Video monitor
Inspector
Name:
Inspector
Robot applications


Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications
Robot View Panel
A real-time, robot simulator!


Untitled - Choregraphe (Connected to 10.80.129.68)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Video monitor
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications
Video Monitor Panel
A real-time camera monitor!


Untitled - Choregraphe (Connected to 10.80.129.68)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Robot view
Robot view
Video monitor
Inspector
behavior_1
Behavior
Public behavior (exposed by your application)
May start on user request
English (UnitedStates)
General
Path:
behavior_1
Name:
Inspector
Robot applications
Inspector Panel
Displays info on a selected box!

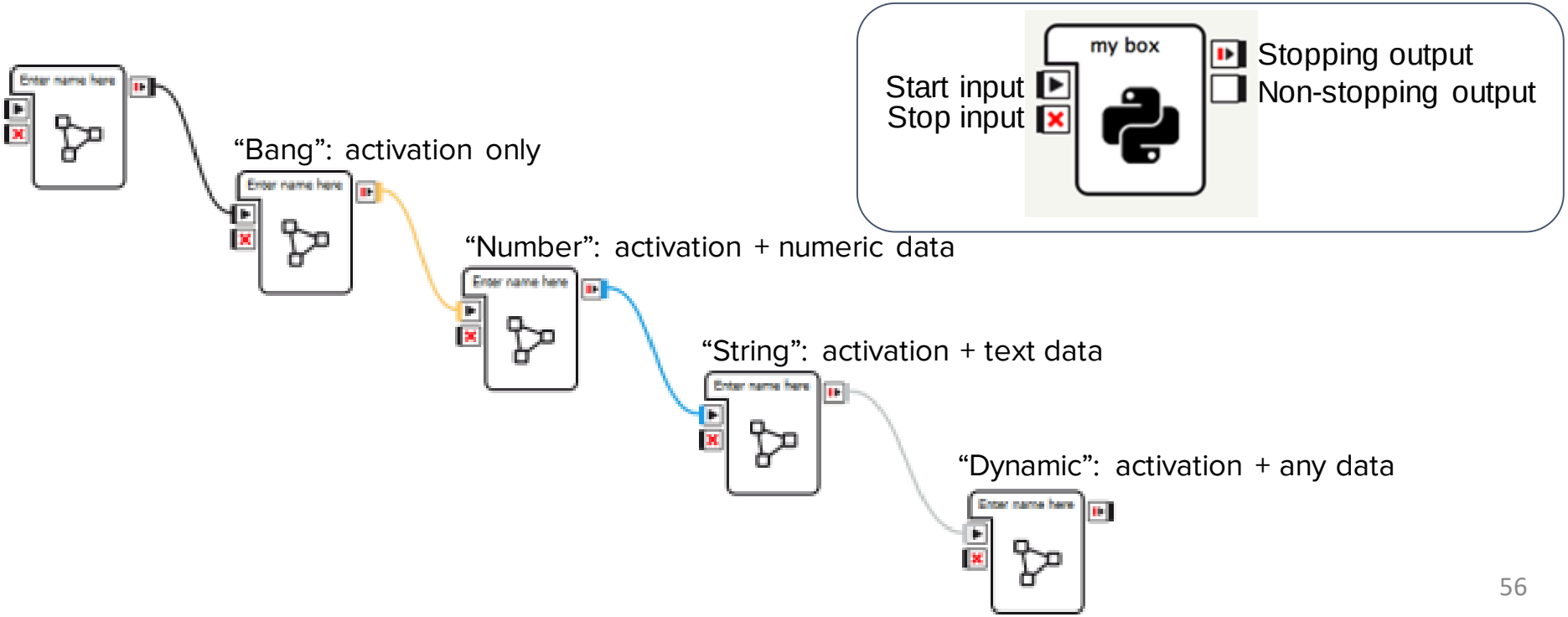

Untitled - Choregraphe (Connected to 10.80.129.68)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
translation_en_US.ts
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Robot view
Robot view
Video monitor
Robot applications
Name
Inspector
Robot applications
Robot Applications Panel
Displays apps of connected robot!


Choregraphe File Edit Connection View Help
Box libraries
Pose library
Video monitor
Project files
Script editor
Robot applications
Inspector
Resource viewer
Active content
Project objects
Dialog
Memory watcher
Robot view
Log viewer
Undo stack
Save Layout...
Load Layout...
Reset Views
Untitled - Choregraphe (Connected to 10.80.129.68)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
translation_en_US.ts
manifest.xml
Untitled.pml
Project files
Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
Robot view
Robot view
Video monitor
Robot applications
Name
.lastUploadedChoregrapheBehavior
ALTactileGesture
Animation Library
Animation mode
Basic Color App Demo
boot-config
j-tablet-browser
robots-advanced
Inspector
Robot applications
View Menu Options
Customize by turning on/off a panel

Choregraphe - Topic II

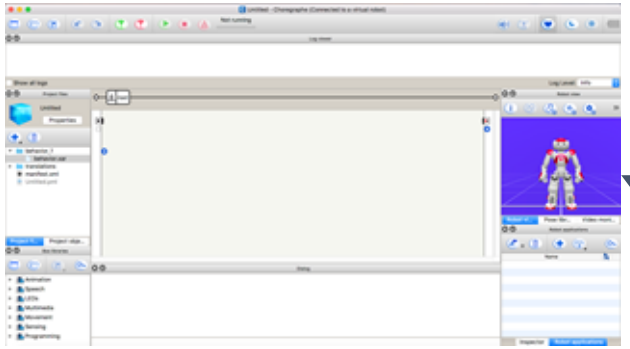
Boxes

Boxes: the building blocks of a Choregraphe app



GUI Abstraction | Choregraphe

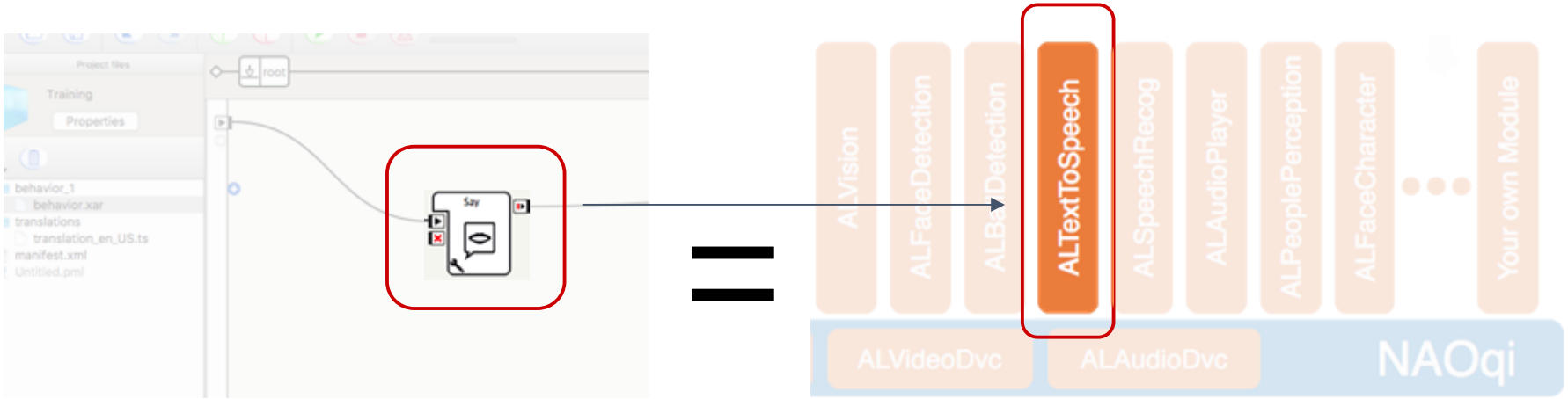
- The O.S. inside Pepper's Head is called NAOqi
- NAOqi works with services



- Choregraphe abstracts the calling of services with a GUI

Boxes = a GUI Abstraction | Choregraphe

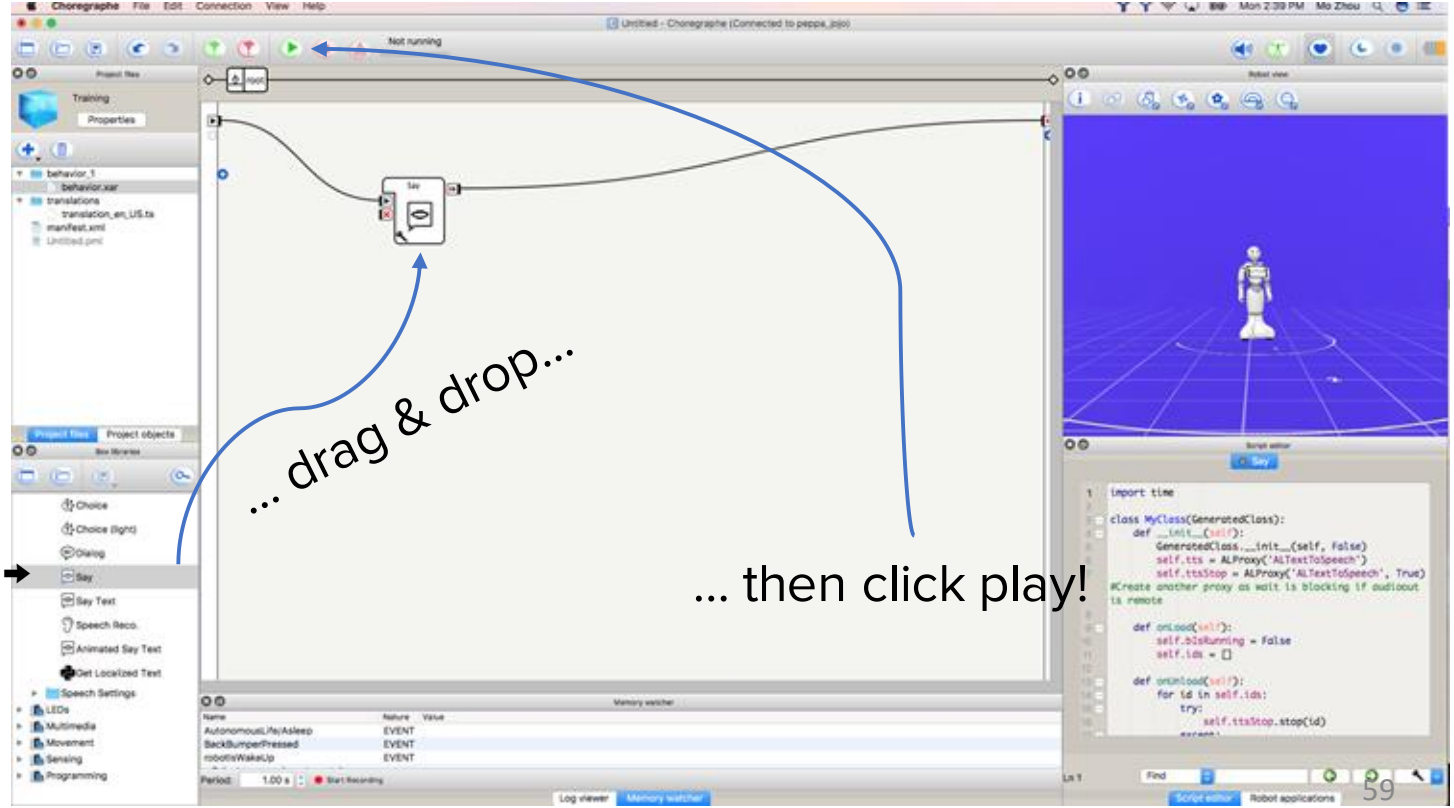
- Choregraphe abstracts the calling of services with a GUI
- **Say** box => calls **ALTextToSpeech** service



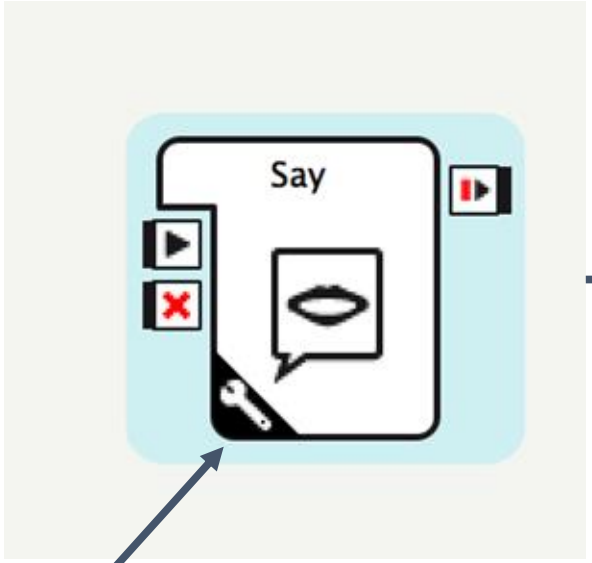
Building Your First App | Choregraphe

Your First App!

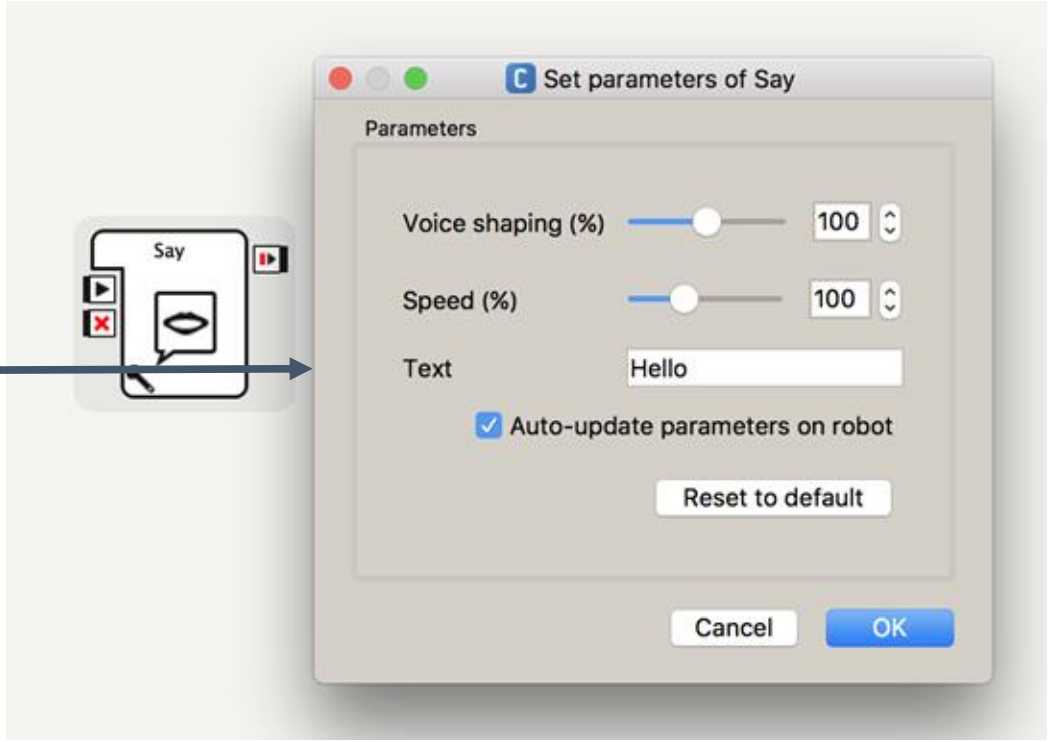
aka “Hello World!”



Introduction to Parameters | Choregraphe



Click wrench to access parameters



Types of Boxes | Choregraphe

Animation
- Timeline

Multimedia
- Play Sound
- Record Sound
- Play Video

Templates
- Python Script
- Diagram

Speech
- Dialog
- Say

Tablet
- Show App
- Tablet Touch

Touch
- Bumpers
- Tactile Head

LEDs
- Eye LEDs

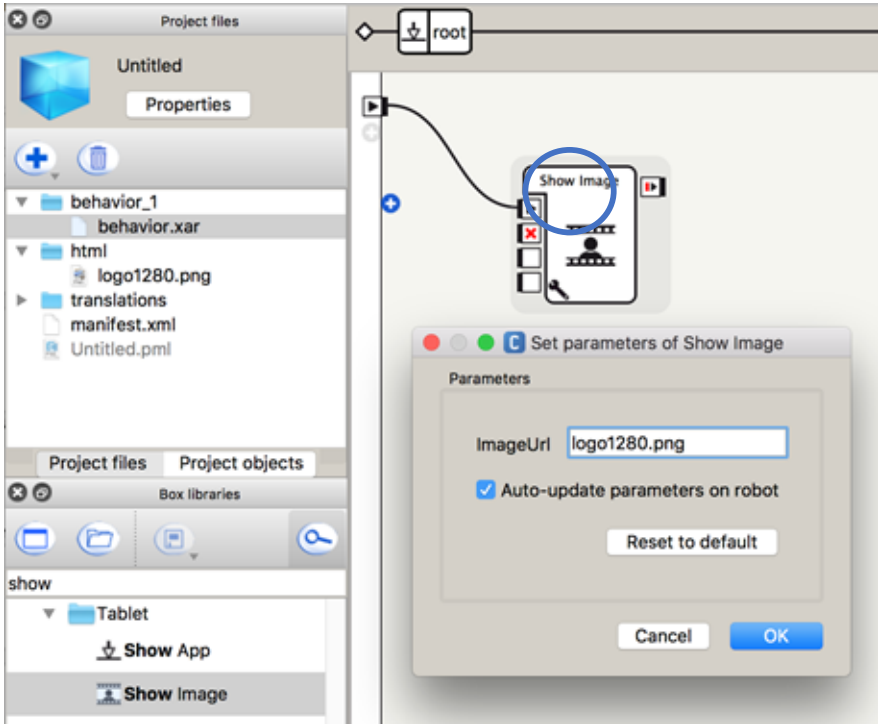
Vision
- Face Detection

Time
- Delay

Tablet: Show Image | Types of Boxes

Using the tablet…

1) Create a new directory: “html”
2) Import an image into this folder
3) Find the box “Show image”
4) Add a link to it
5) Edit the parameter to the name of your image
6) Play the behavior!



Play Sound | Types of Boxes

Public domain:

- https://freesound.org/browse/
- http://soundbible.com/
- http://pdsounds.org/

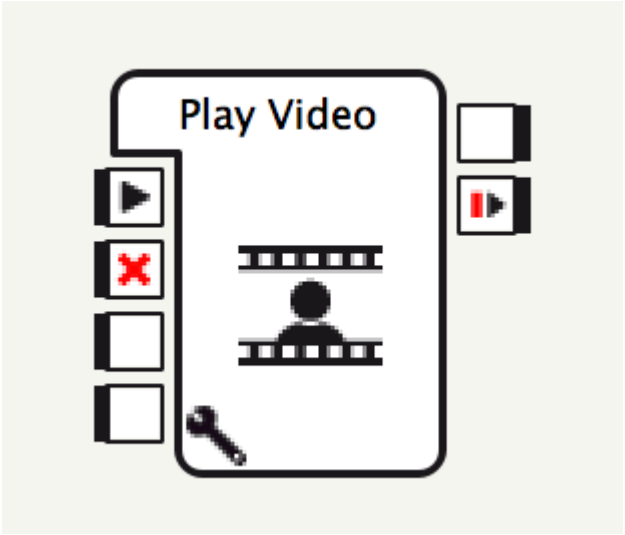
Play Video | Types of Boxes

Public domain / Creative Commons:

- https://www.pond5.com/free
- https://vimeo.com/creativecommons/cc0
- Youtube: How to …

Downloading YouTube/Vimeo/etc videos:

- www.keepvid.com

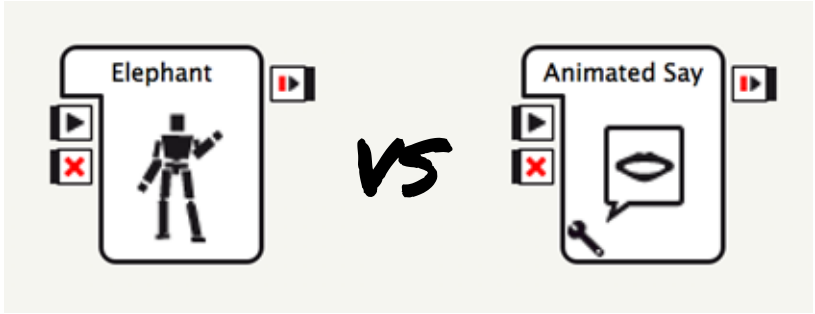


Animations | Types of Boxes

Animated Speech takes care of you most of the time…

Some circumstances call for a specific animation:

- Exaggerated gestures
- Dances
- Reactions

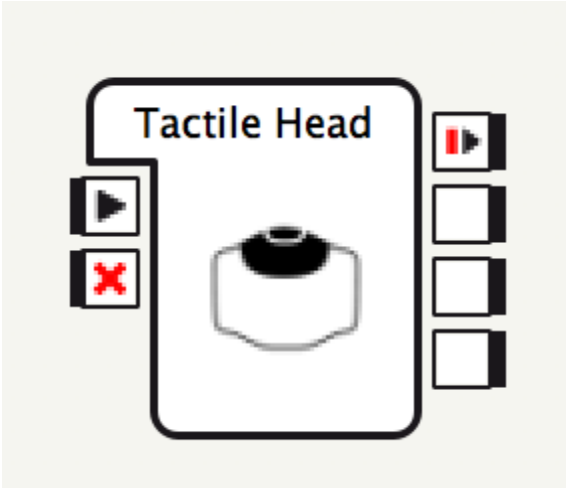


Touch Head | Types of Boxes

Pepper has “presence.”

--

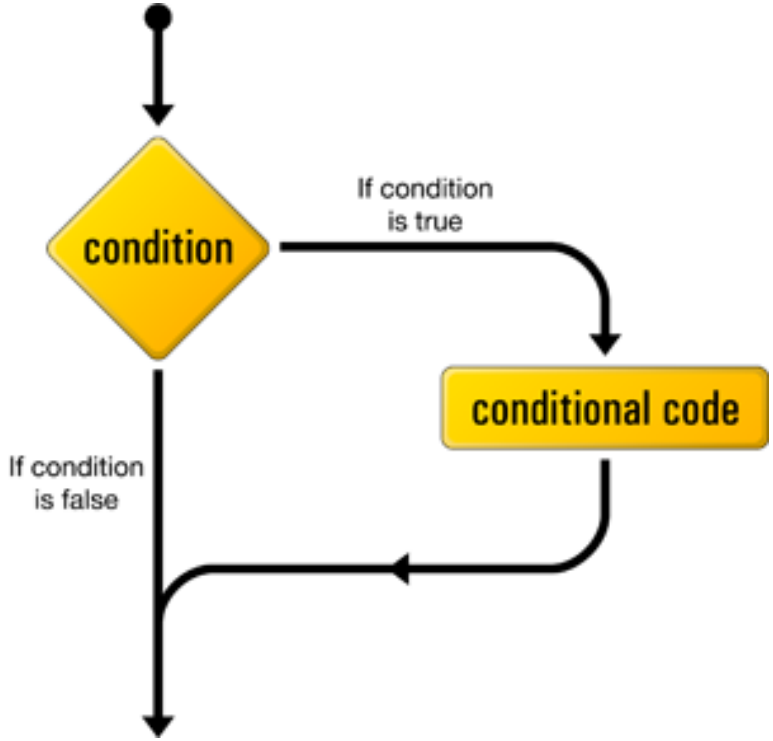
Take advantage of it!



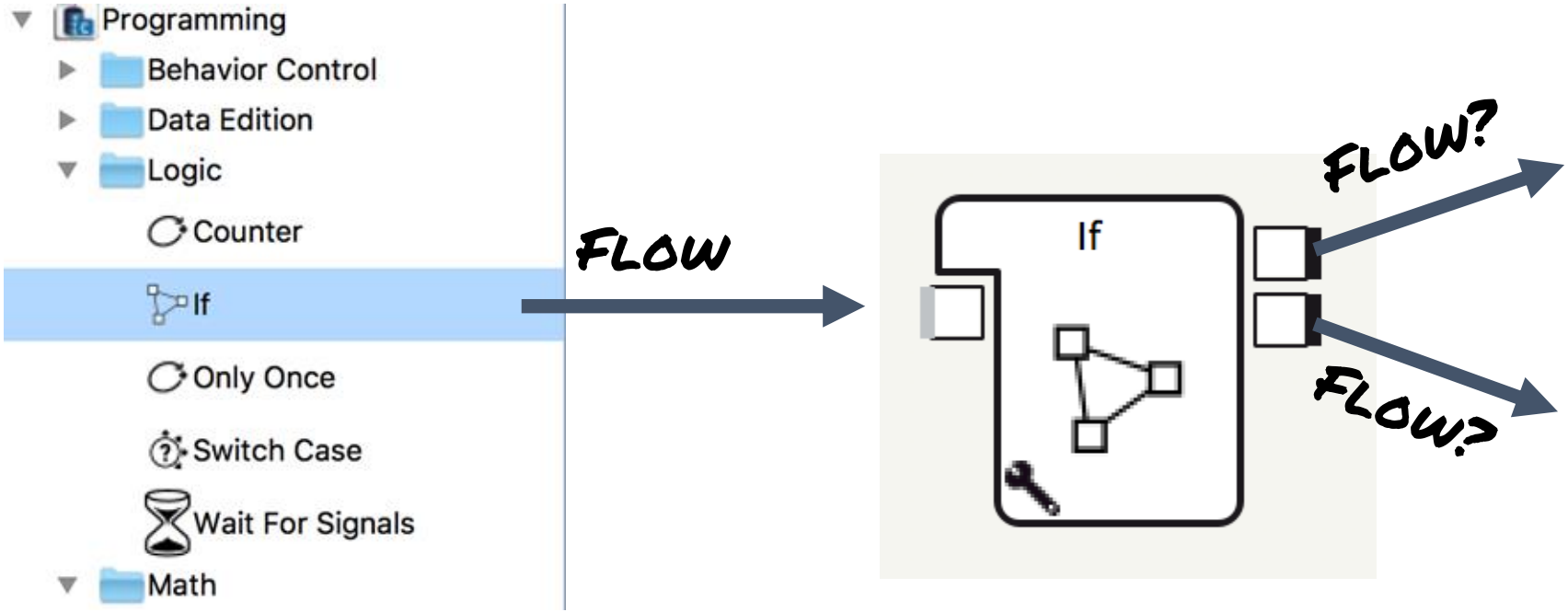

Intro to
Programming

Conditional Statements | Programming

Conditional statements allow an app to flow in different directions based on a user's input.



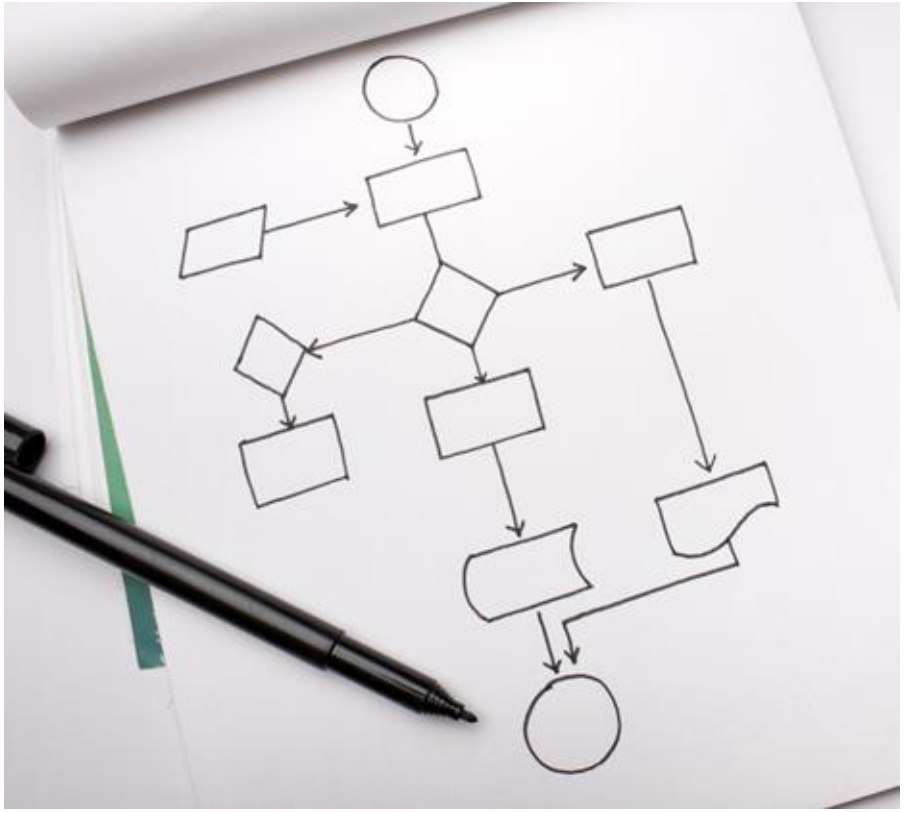
Programming: If | Types of Boxes



Flow Control | Programming

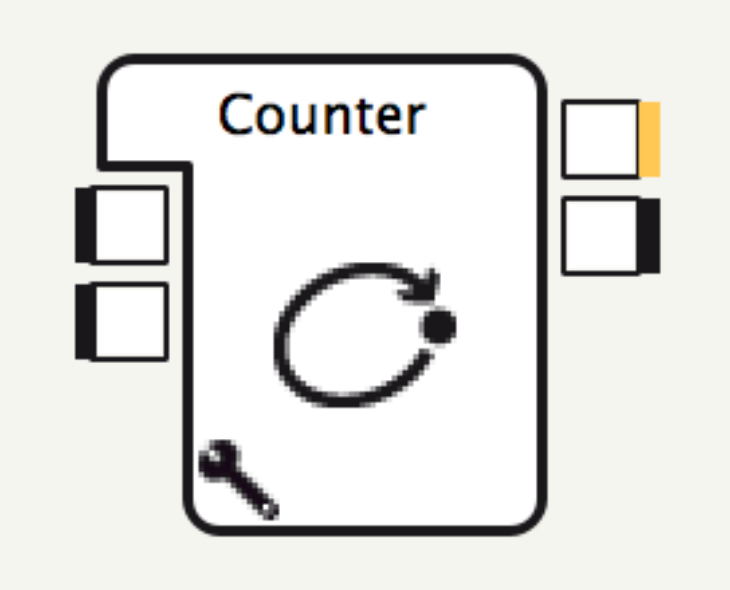
Flow Control in a Choregraphe application is determined by:

- Conditional Statements
- Loops
- Function Calls
- Signal Events



Counter | Types of Boxes

Loop boxes a certain number of times using Counter

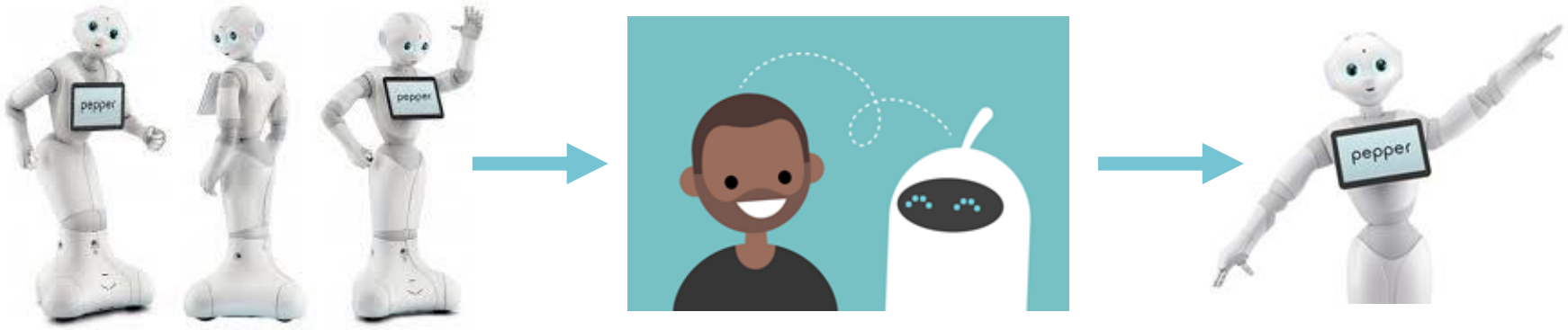


Choregraphe - Topic III

Prototyping an Application

Q: *What is an application?*

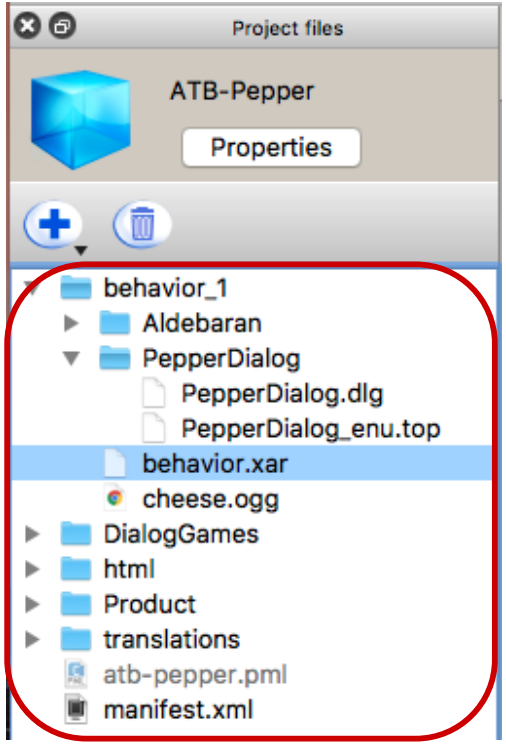
A: *A set of robot actions that tell a story in 4D and provide a human with an experience.*



Q: *What is a prototype?*

A: *A set of robot actions (think boxes!) that demonstrate a proof of concept of an application. In terms of files, it includes:*

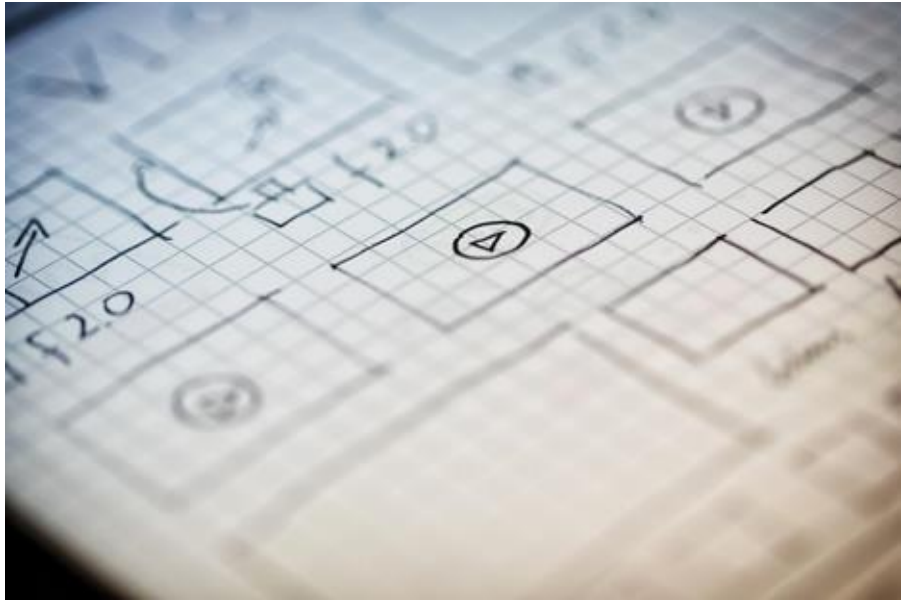
- **Behaviors** (.xar): what pepper can do
- **Dialog topics** (.dlg / .top): what pepper can talk about
- **Other resources** (media, scripts, web pages…): app content, html, complex actions, etc.
- **Properties** (icon, name, …): configurations, settings, etc.



Storyboarding Your Prototype | Choregraphe

Remember: your goal is just to make a proof of concept!

- Scope prototype to the key section(s) of your app idea
- Sketch out the flow of the app in frames, including:
  - the Tablet design
  - the Dialog
  - Robot movements



Quick Detour #1

NAOqi Access Methods

SSH + SFTP | NAOqi Access Methods

When working on the robot, everything** happens in the head processor:

- Recordings are stored in the head
- Logs are stored in the head

**The Tablet is mostly used as a display / monitor

What connection protocols can you use to access the robot?

SSH

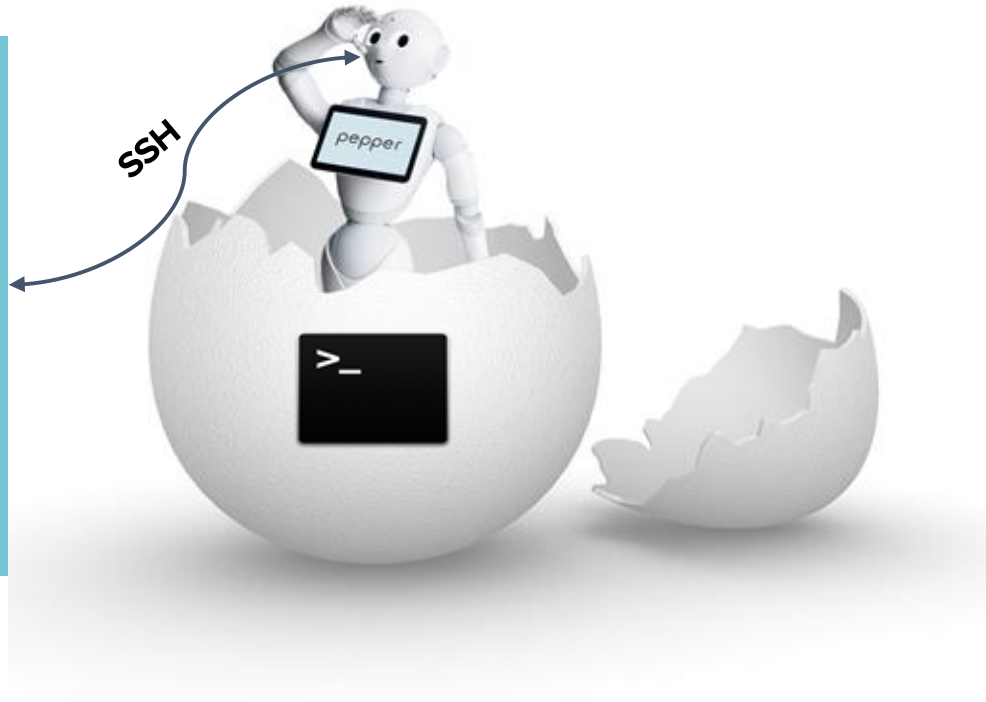
Access the terminal

SFTP

Access the file system

SSH via Command Shell | NAOqi Access Methods

>_ Access Pepper's Linux head (NAOqi) remotely with your command shell via SSH



SSH via Command Shell | NAOqi Access Methods

Terminal

> ssh nao@{Your-Robot's-IP Address}

Example:

> ssh nao@10.80.129.11

> Are you sure you want to continue connecting (yes/no)? yes

> Warning: Permanently added '10.80.129.97' (ECDSA) to the list of known hosts.

> Password: nao {default password = 'nao'}

Terminal

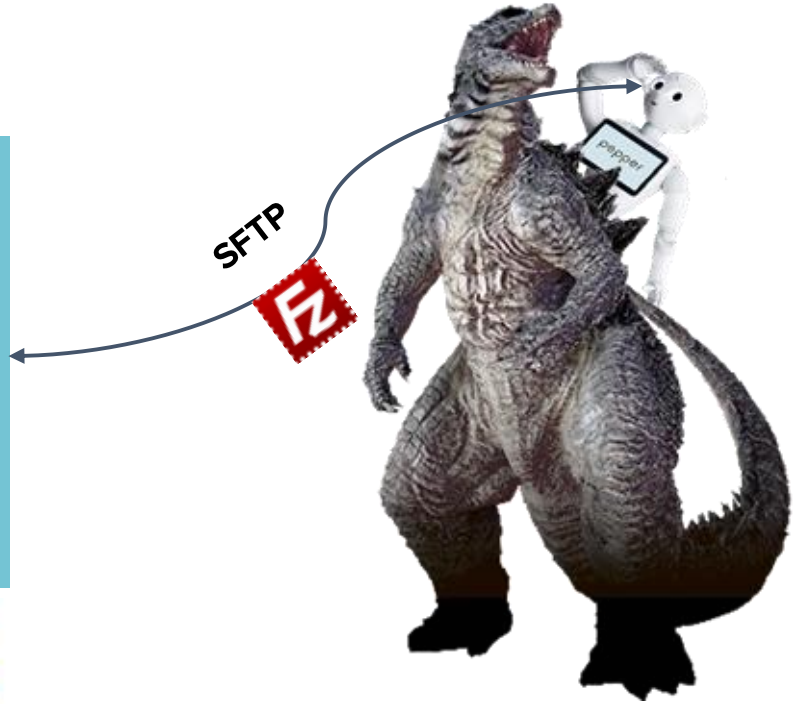
Challenge: Make the robot say "*Hello World!*"

Example:

```
> ssh nao@<Your-Robot's-IP>
> Password: nao
> pepper [0] ~ $  say "hello world"
```

SFTP via FileZilla | NAOqi Access Methods

Access Pepper's File System remotely with an SFTP Client, such as FileZilla



Other SFTP Clients:



Transmit
Cyberduck

WinSCP

Pro Tip #1

Debugging: Reflash the Controllers

Reflashing the Controllers | Debugging

- Motors or sensors misbehaving? (e.g. wrong angle, unreachable peripheral, etc.)
- Pepper detect and notify you of an error?

Reflash the controllers:

- Switch off
- Wait 10 seconds
- Press and hold the chest button for 8s (shoulders turn blue; activates reflash)
- Reflash boot will take ~15-20min

reflash /riːˈflash/ verb - Boot mode that checks all the internal microcontrollers and flashes (resets) their firmwares if needed. No user information will be impacted; it's only low-level.