

BTstudio, a web tool for programming robots with Behavior Trees



Asociación de Robótica e Inteligencia Artificial JdeRobot
<https://jderobot.github.io>

*josemaria.plaza@gmail.com, javizqh@gmail.com,
oscar.robotics@tutanota.com*

- Introduction
- Using BT Studio
- How it has been developed?
- Experimental validation
- Conclusions

Introduction

JdeRobot.org

- Develops open source sw in Robotics and AI
- Started in 2018, 20+ members
- Projects:
RoboticsAcademy, BTstudio, Unibotics, VisualCircuit...
- Activities:
Google Summer of Code (2015,2017-2024), internships
- <https://jderobot.github.io>, YouTube, LinkedIn, Twitter



Making Behavior Trees more accessible for Robotics applications

- Development trend of using **Behavior Trees** in Robotics applications, fairly complex ones beyond simple reactive ones. Similar to HFSM.
- Goal: **to facilitate the quick deployment of BT-based robotics applications within ROS2**
- Inspired on already established tools: [Groot](#) and [Groot2](#).
- Built on top of [py_trees](#) for better compatibility.
- Provides a similar experience to [BehaviorTrees.CPP](#) 3.8 but for Python.

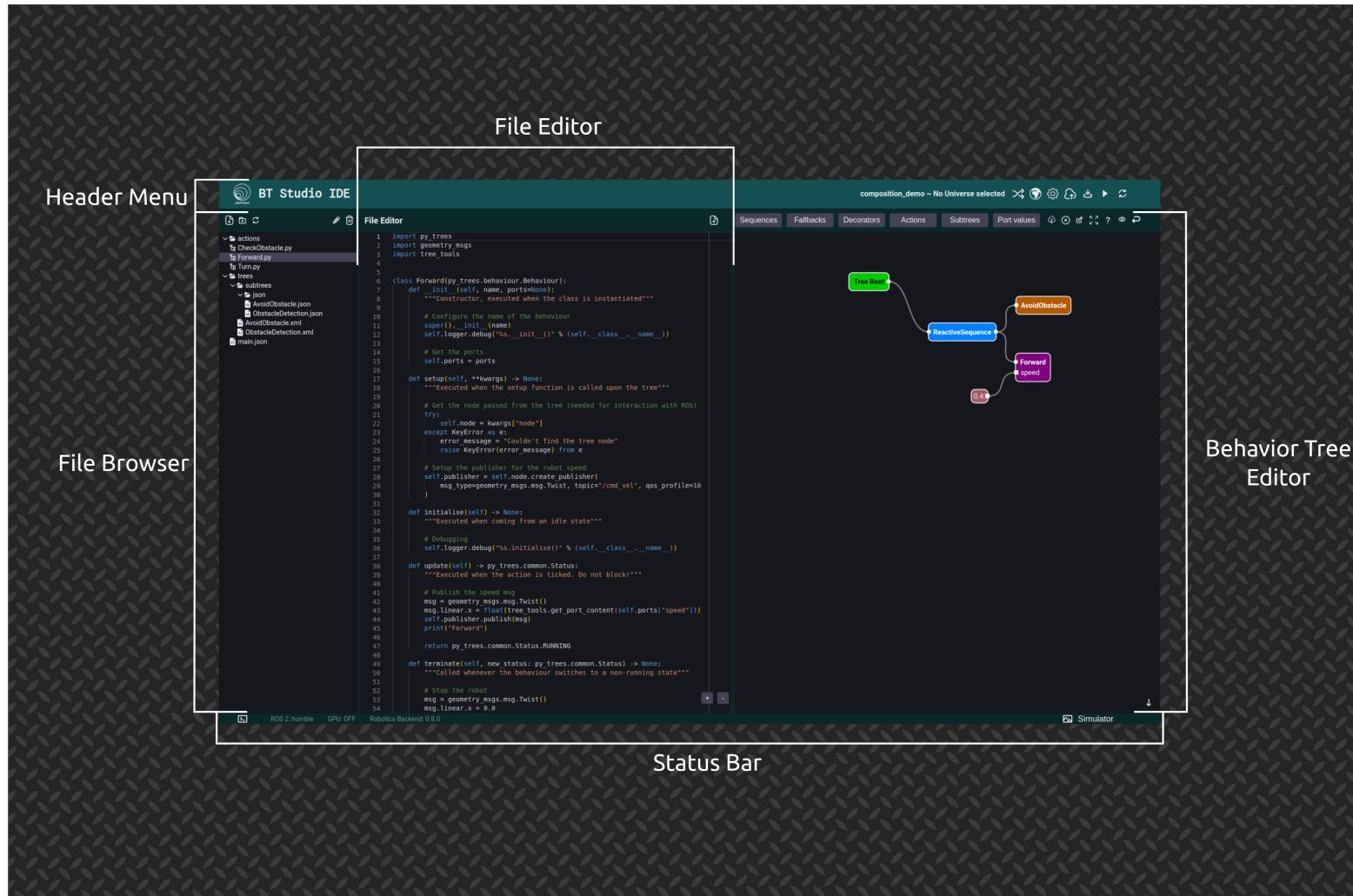
BTstudio

- Web based
- Python language
- Develop applications for ROS2 Humble
- Reuse of behavior trees and modification in a graphical interface
- Free and open-source
- Execution in a dockerized environment (*Robotics Backend*)
- (optional) Streamlines the process of creating a ROS2 package

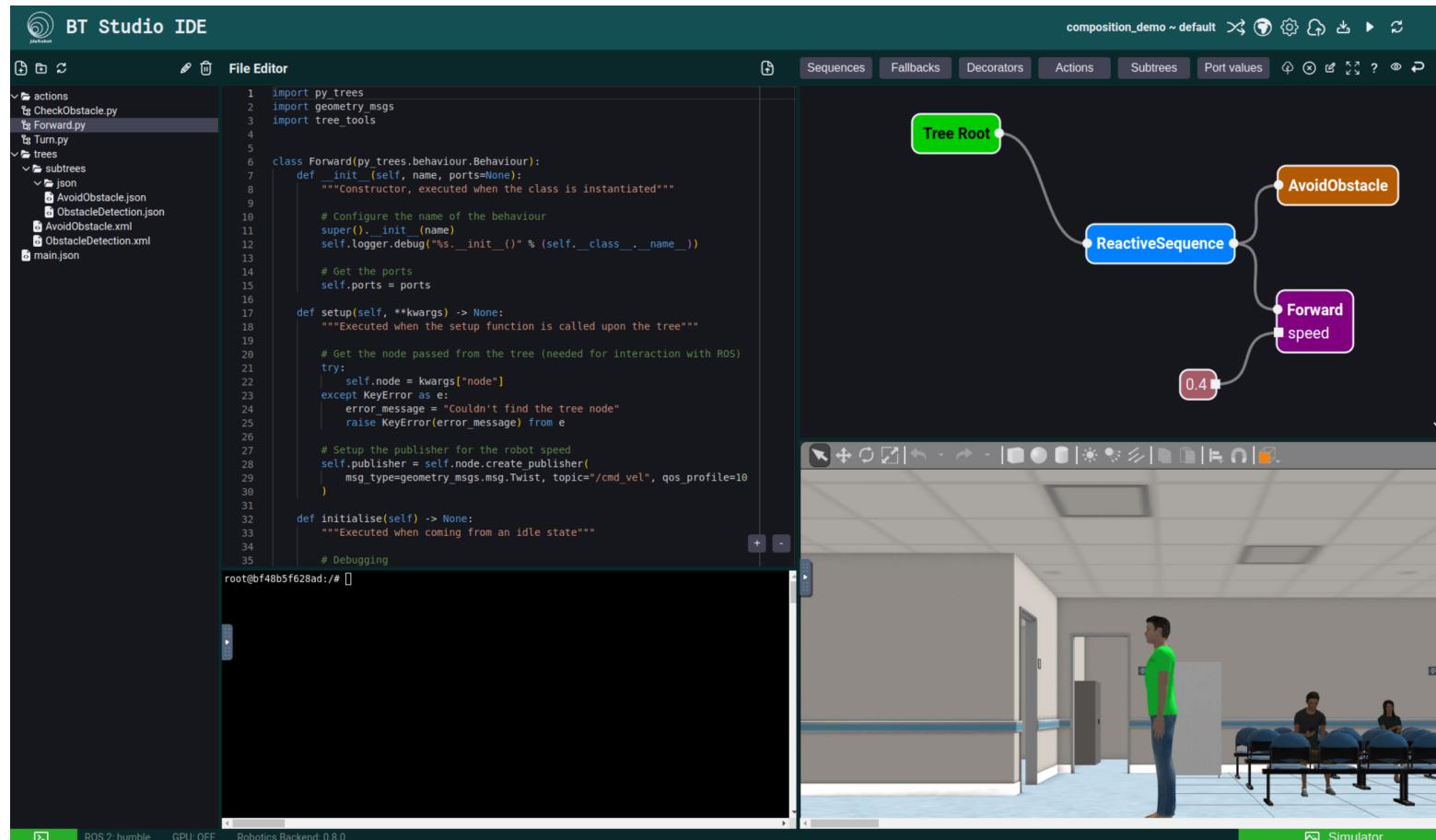
Features

- Behavior Tree **actions** are created in a *text editor*
- Behavior Tree **structure** is defined using a *visual editor*
- Integrated *execution viewer* for dockerized execution, VNC based
- Integrated *text console*
- Both on real robots and on simulated robots (Gazebo, Webots...)
- Manage multiple projects
- Manage multiple universes

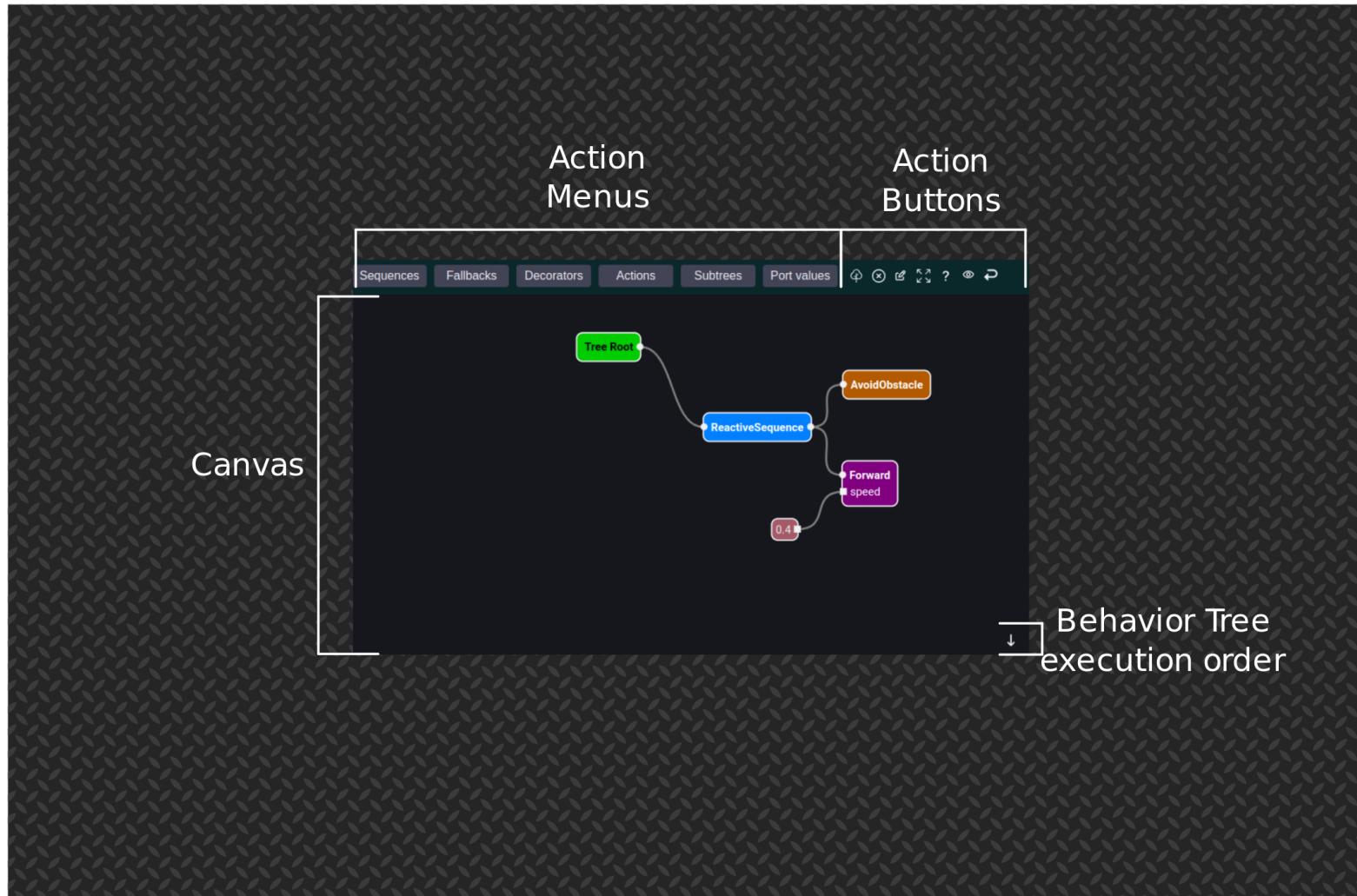
User Interface



■ Text editor + Visual BT editor + Console + Execution viewer



Visual Behavior Tree Editor



- Intuitive and reactive editor
- Customizable colors for each action
- Customizable order of BT execution (bottom-to-top or top-to-bottom)
- Support for Subtrees and composition
- Everything you need for developing BT based applications

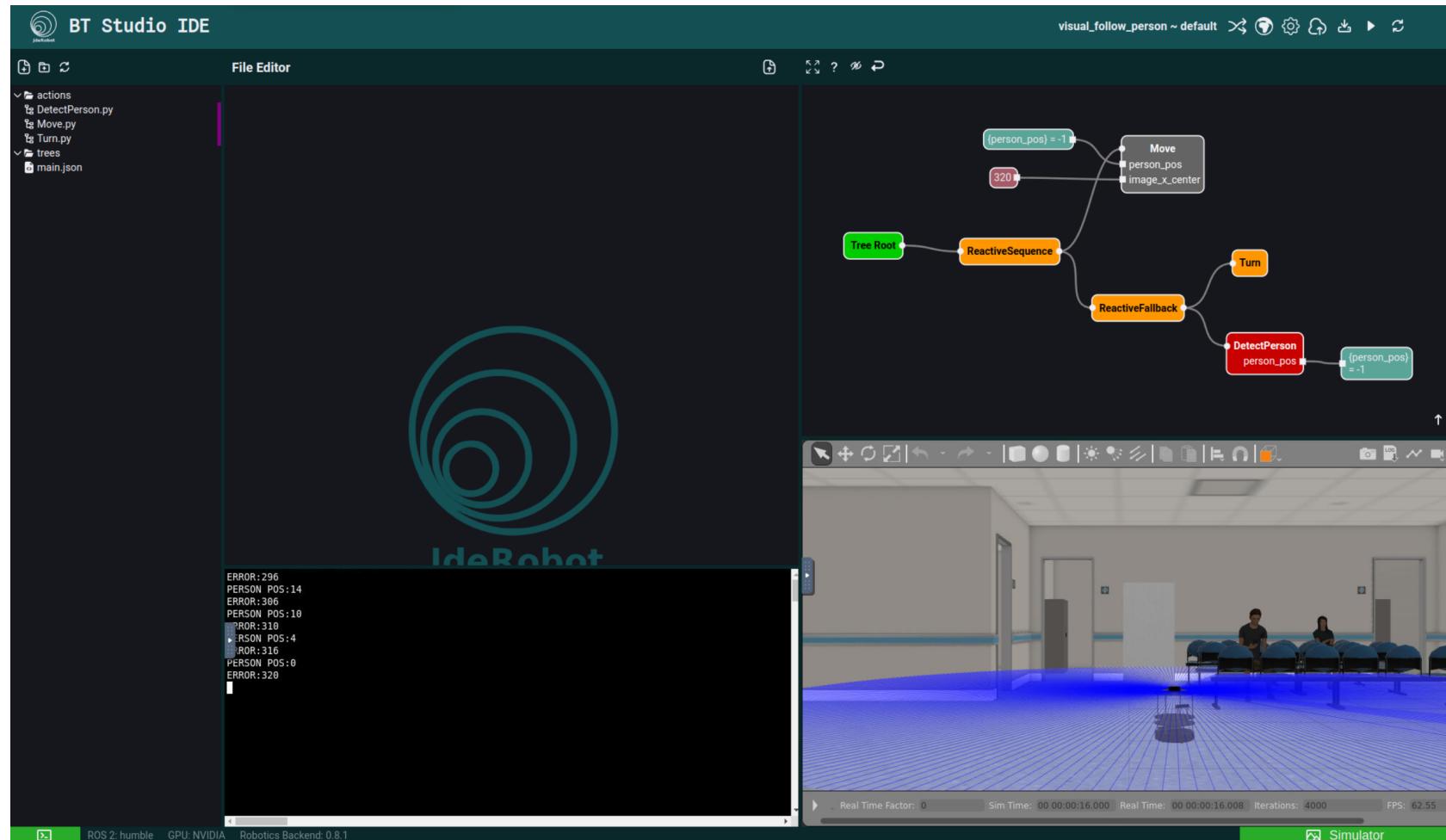
Subtrees

- BT composition
- Speed up the development of complex robotics applications
- Reusing of existing BTs for common functionality
- Library

Execution monitoring of the Behavior Tree

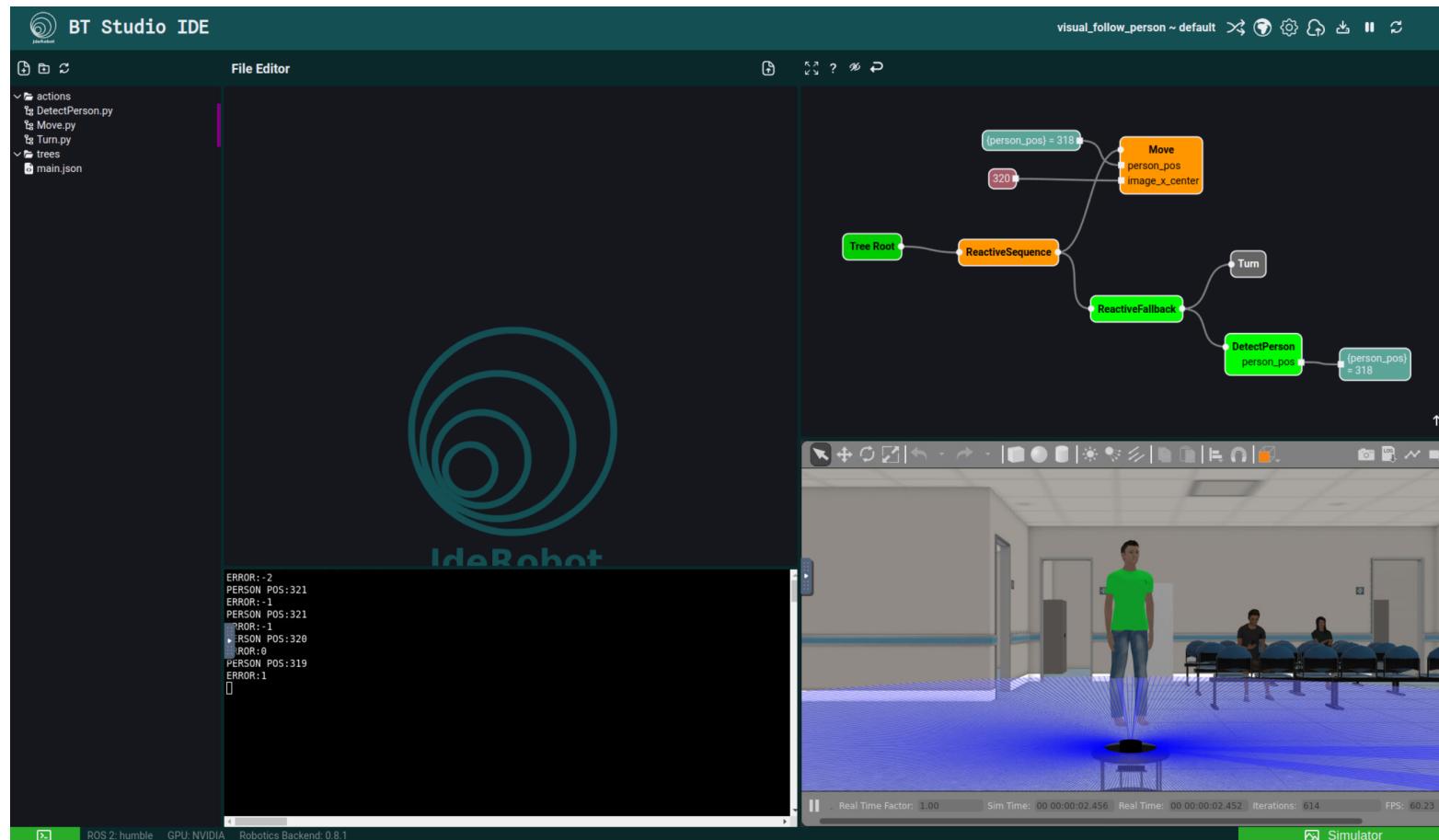
Monitor the execution status of the behavior tree from inside the docked execution

- Real time updates
- Move through the subtrees seamlessly
- Also displays the content of blackboard tags



Dockerized Execution

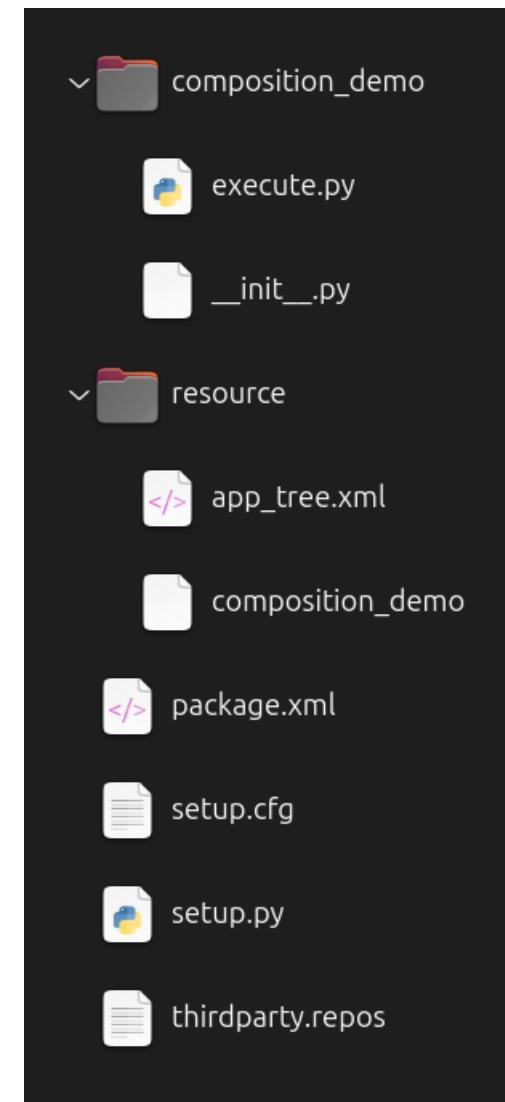
- Use multiple universes seamlessly
- Ready to use universes provided by the Robotics Backend
- Use your own custom universes
- Control the flow of execution: Run, Pause or Restart



Local Application Package

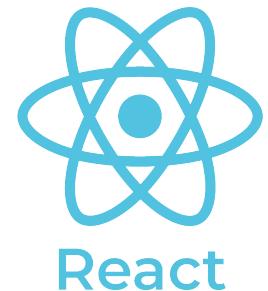
- ROS2 Humble or Jazzy is needed
- A testing environment is provided with the Webots simulator and a tree execution visualizer as third-party repos.
- Compile and run the app using the executor provided
- The actions and behavior tree are merged into a single xml source file.

- `app_tree.xml`: behavior tree and source code
- `execute.py`: launcher for the application
- The rest is the same as a basic ROS2 package



Developers: How it works?

- Web technologies
 - backend: Django
 - frontend: React, HTML5, CSS
- Robotics technologies
 - ROS2
 - Based around py_trees
- DevOps technologies
 - Docker



Action Structure

- The structure is the same as py_trees actions

```
● ● ●

1 class Action(py_trees.behaviour.Behaviour):
2     def __init__(self, name, ports=None):
3         """Constructor, executed when the class is instantiated"""
4
5     def setup(self, **kwargs) -> None:
6         """Executed when the setup function is called upon the tree"""
7
8     def initialise(self) -> None:
9         """Executed when coming from an idle state"""
10
11    def update(self) -> py_trees.common.Status:
12        """Executed when the action is ticked. Do not block!"""
13
14    def terminate(self, new_status: py_trees.common.Status) -> None:
15        """Called whenever the behaviour switches to a non-running state"""


```

Translation process

- Translating from the user code and the diagram is done in the backend
- The 2 parts are combined into a xml single file divided into 2 sections: the BehaviorTree and the Code
- In the BehaviorTree section resides the Behavior Tree and is the same as what is generated by Groot2.
- The code section is used instead of external files for containing each action source code

Working demos

Follow Person Demo

- Video

Bump & Go Demo

- Video

Receptionist Demo

- Video

Conclusions

- BTstudio facilitates the quick deployment of behavior tree-based robotics applications within ROS2
- Integrated in Unibotics robot programming website

- Present BTstudio to the open source community and ROS community
- Develop more relevant robotics applications and subtree library
- Jump to BehaviorTree.CPP 4.X