McGill

MSDL

Modelling, Simulation and Design Lab

Statecharts modeling of a robot's behavior

MSDL Research Presentation Day 2008

Silvia Mur Blanch

Introduction: main characters

- **Silvia Mur Blanch**

I joined the MSDL in January ‘08, took the course COMP763 and was introduced to statecharts and the wonderful world of AToM³.

All in all I had a great time there, especially after I managed to make the robot move around while whistling the *Imperial March*.

- **iRobot *Create***

It’s full of possibilities, it can do practically anything, but vacuuming. It has 32 built-in sensors.

I communicate with it using a Class I BlueTooth dongle and an adapted version of a high-level Python interface called *PyRobot*.

Project background

This project is based on:

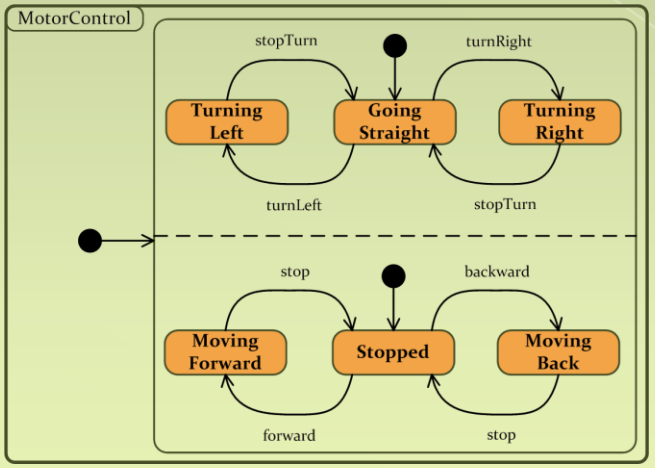
"*Model-based design of a computer-controlled game character behavior*"

Jörg Kienzle, Alexandre Denault, Hans Vangheluwe

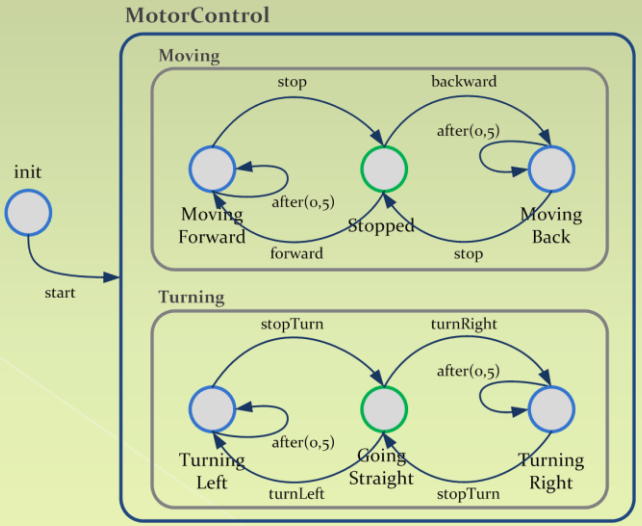
- Modeling game AI at an appropriate abstraction level, using an appropriate modeling language
- Event-based approach: modularity, efficiency, implementation independent
- Rhapsody Statecharts:
  - State/event-based
  - Autonomous/reactive behavior
  - Notion of (real) time

Chapter I: Statecharts modeling of TankWars

- Class project for COMP763
- Same statecharts, different philosophy: in the original designs, the statecharts are purely event-based; in the new designs, time is used to force guards checks for transition triggering



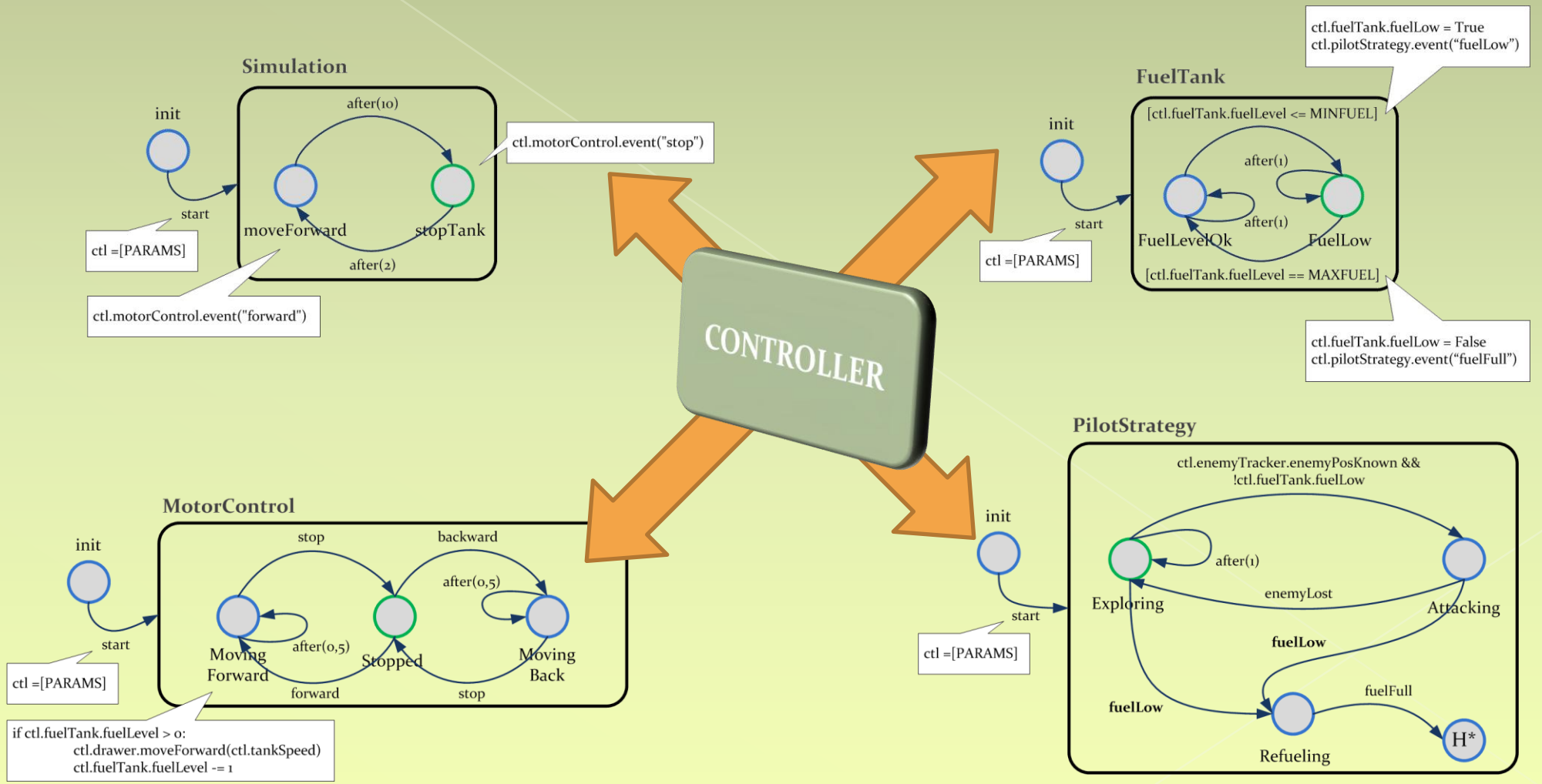
VS.



- Necessity to use a controller class to link tank's components/statecharts and allow communication between them: ***narrow cast***
- Implementation of a very simple simulation environment using Python Tkinter
- The description of the simulation process is implemented with a statechart too

Chapter I: Statecharts modeling of TankWars

Simulation using Statecharts



Chapter II: Modeling the robot's behavior

- Found high-level Python interface for iRobot Create called *PyRobot* and adapted it, adding some new functions
- Communication with the robot through a *BlueTooth Serial Port* (using a *BlueSoleil* Class I BlueTooth dongle)
- First experiments consisted on simply making the robot move around, directly calling PyRobot functions

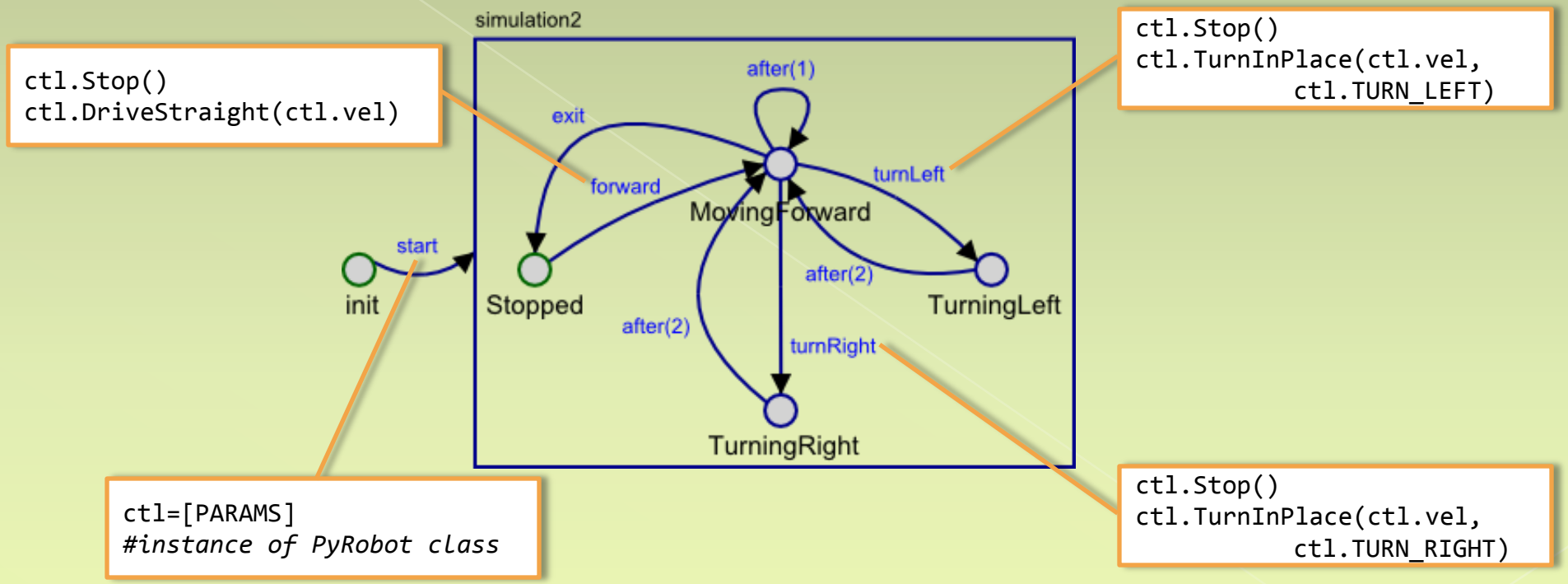
```
port = raw_input("++ enter port: ")
C = Create(tty=port)
C.Control() #puts robot in full mode

vel = raw_input("++ enter velocity: ")
C.Drive(vel, RADIUS_STRAIGHT) #makes the robot move forward

packet_id = 6, #requests all sensor packets
C.RequestSensorData(packet_id) #requests sensor data
C.PrintSensorData() #prints sensor data on the console screen
```

Chapter II: Modeling the robot's behavior

- Next step was to implement robot's behavior with statecharts that use the *PyRobot* class as a controller



- Latest simulation introduces reading and reacting to robot's sensor data. For example: after bumping against an obstacle, turn 180 degrees and keep moving forward

Chapter II: Modeling the robot's behavior

- Design and development of the *CreateRemote* project, based on the *DigitalWatch*

Chapter III: World discovery

- So many possibilities, very little time. We wanted to work with the *command module* but it's out of the range of this project
- The project is now oriented to exploring and mapping out the world by moving around a room and bumping against its walls/furniture/obstacles
- Every time the robot bumps into an obstacle, new data is added to the robot's own map of the world (at the beginning of the execution it will be like a blank canvas)
- After some time, the robot should have a very approximate representation of the world/room and its walls/obstacles
- When the world is mapped, it should be possible to automatically avoid its obstacles when tracing a path from one point of the map to another
- Still updating project information on MSDL personal webpage! http://moncs.cs.mcgill.ca/people/silvia

Thanks for all!

(and, hopefully, see you again someday)