STATECHARTS MODELLING OF TANK WARS

[*Project Presentation*]

School of Computer Science
McGill University, Montréal, Canada

COMP 763 / Silvia Mur Blanch

Overview

- Recap
- The static part: the controller
- Designing the tank components' statecharts
- Recreating the simulation environment
- Summary

Recap

- EA TankWars: AI programming contest, C++ environment
- AI specification should be done at a higher lever of abstraction: modelled instead of coded (reusability)
- Since models define reactions to game events, event-based formalisms seem to be the most appropriate
- Formalisms:
  - State / event based
  - Autonomous / reactive behaviour
  - Notion of time
  - Modularity
  - Availability of simulators / code generators
- Time-sliced Vs. Event-based

Overview

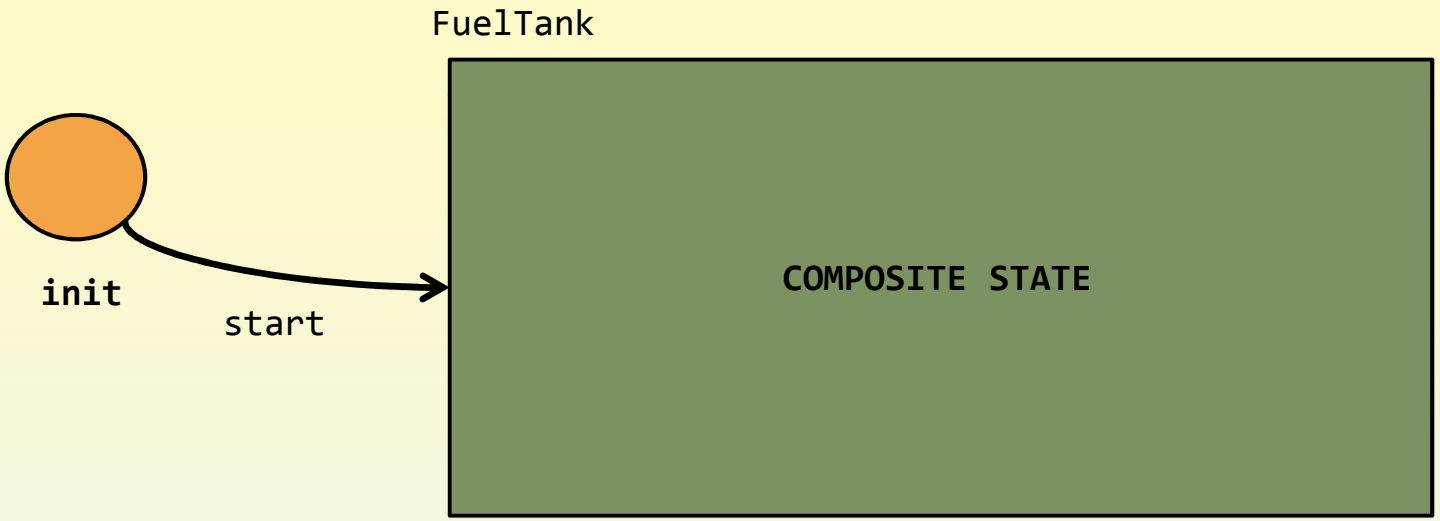
- Recap
- The static part: the controller
- Designing the tank components' statecharts
- Recreating the simulation environment
- Summary

The static part: the controller

- One big statechart (digital watch) Vs. many small statecharts
- Defining each component in different statecharts means event broadcasting is not possible
- Necessity to use controller: it will “contain” all the tank’s components and link them

The static part: the controller

- Defining each component of the tank with a statechart, e.g.:



The static part: the controller

- Event “start” to begin the simulation of the statecharts, (pass controller as parameter) e.g.:

```
print “fuelTank start”
ctl=[PARAMS]
```

- Defining an instance of each component / statechart of the tank, e.g.:

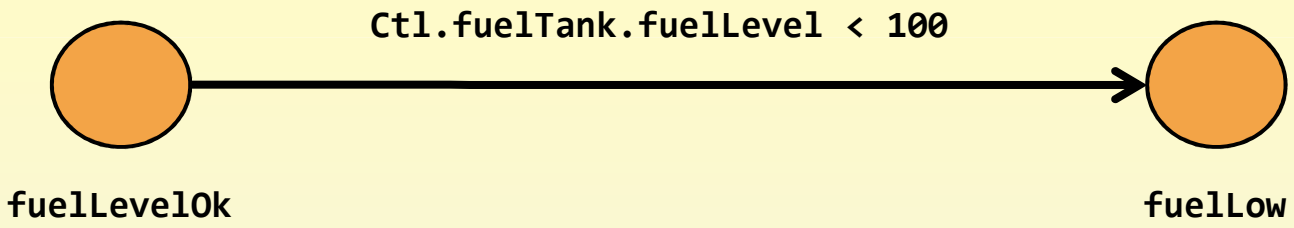
```
self.fuelTank = FuelTank.FuelTank()
self.fuelTank.initModel()
self.fuelTank.event(“start”, self)
```

- Adding variables to the tank’s components, e.g.:

```
self.fuelTank.fuelLevel = MAX_TANK_FUEL
```

The static part: the controller

- Tank's components generate events in other components (statecharts) by using "*narrow cast*", e.g.:
  - FuelTank



  - Action:

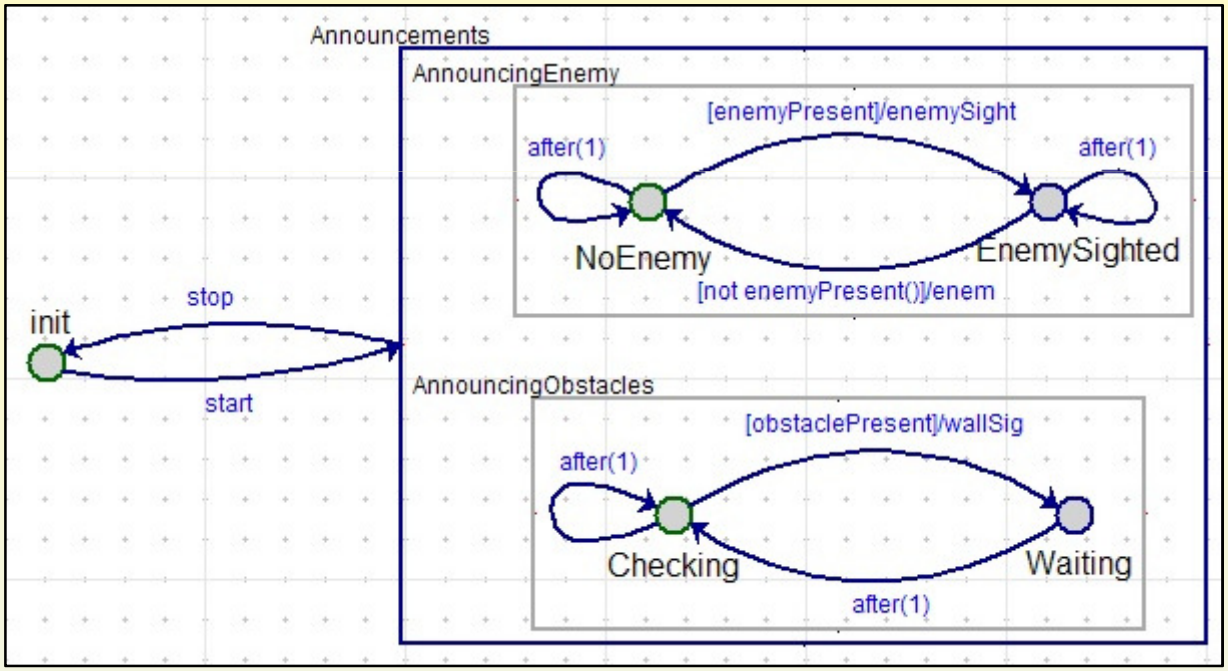
```
ctl.fuelTank.fuelLow = True
ctl.pilotStrategy.event("fuelLow")
```

Overview

- Recap
- The static part: the controller
- Designing the tank components' statecharts
- Recreating the simulation environment
- Summary

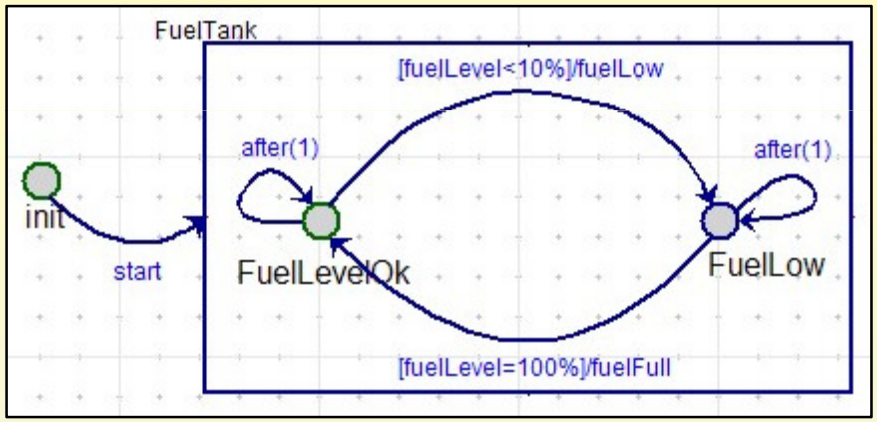
Designing the tank components' statecharts

- Announcements (sensors)



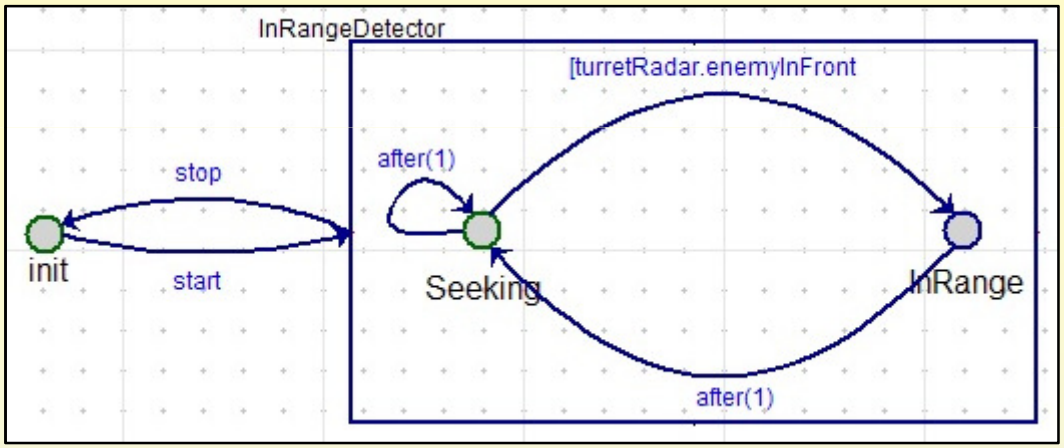
Designing the tank components' statecharts

- FuelTank (sensors)



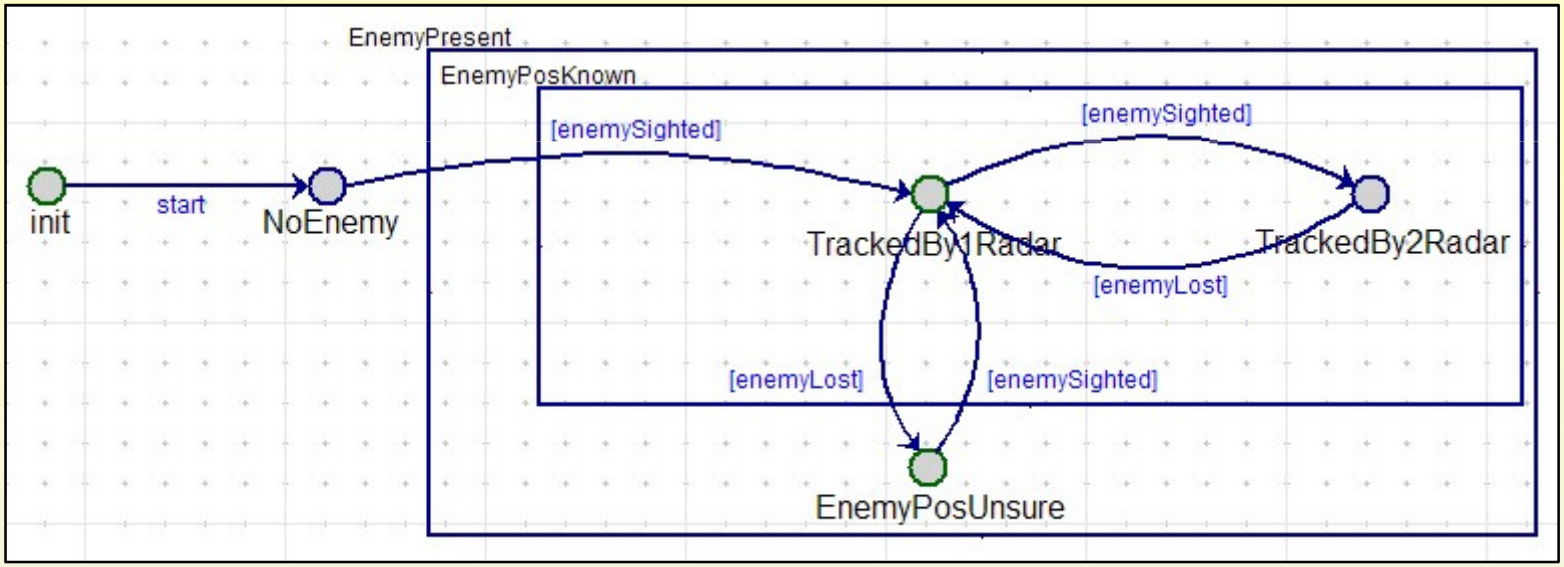
Designing the tank components' statecharts

- InRangeDetector (analyzers)



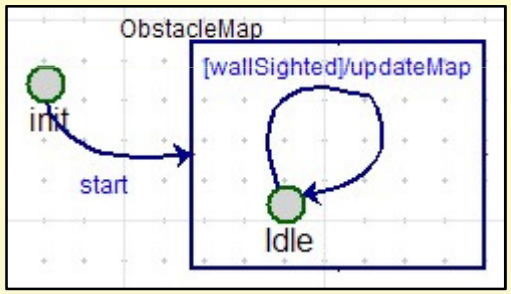
Designing the tank components' statecharts

- EnemyTracker (memorizers)



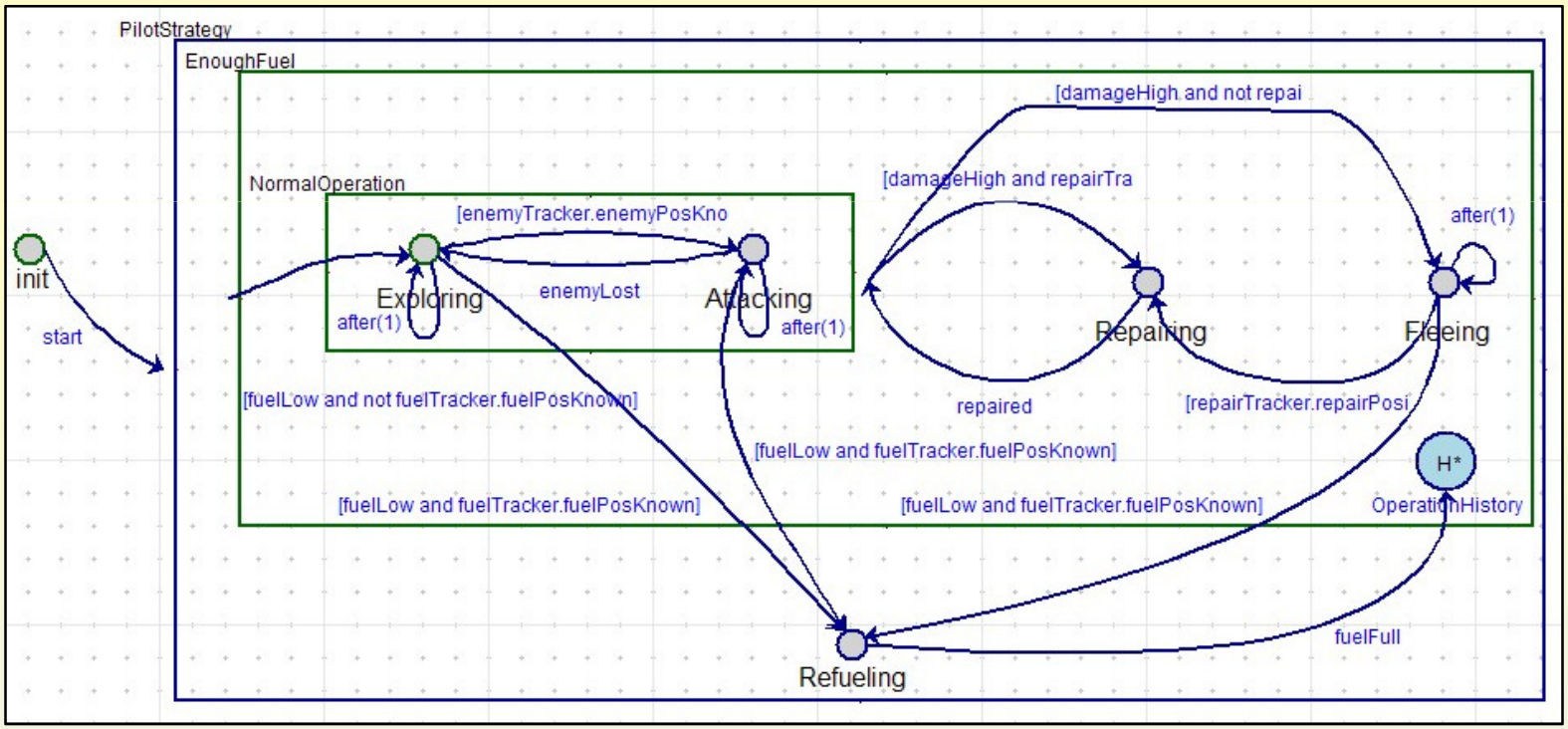
Designing the tank components' statecharts

- ObstacleMap (memorizers)



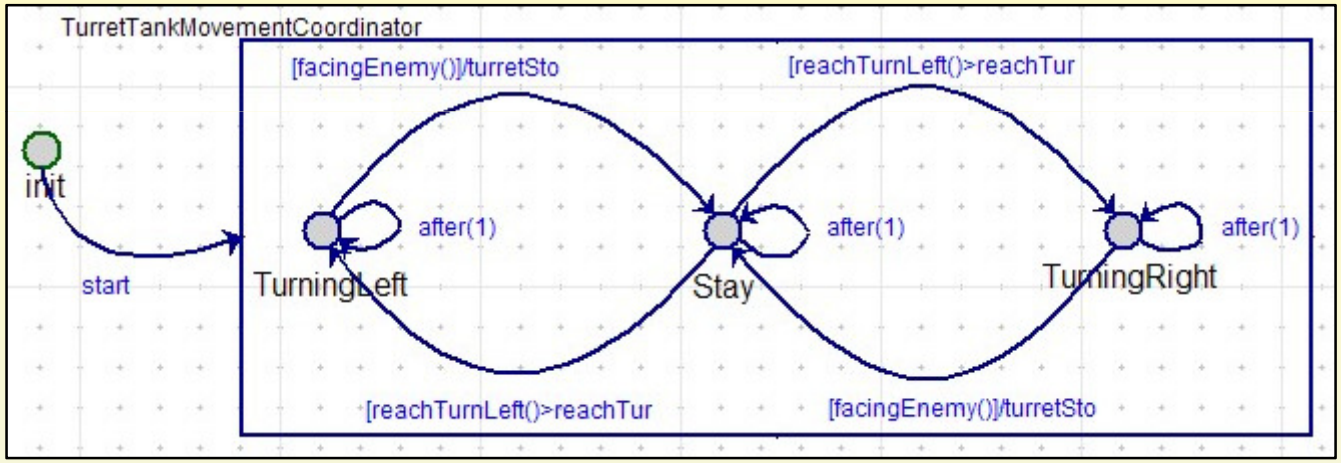
Designing the tank components' statecharts

- PilotStrategy (strategic deciders)



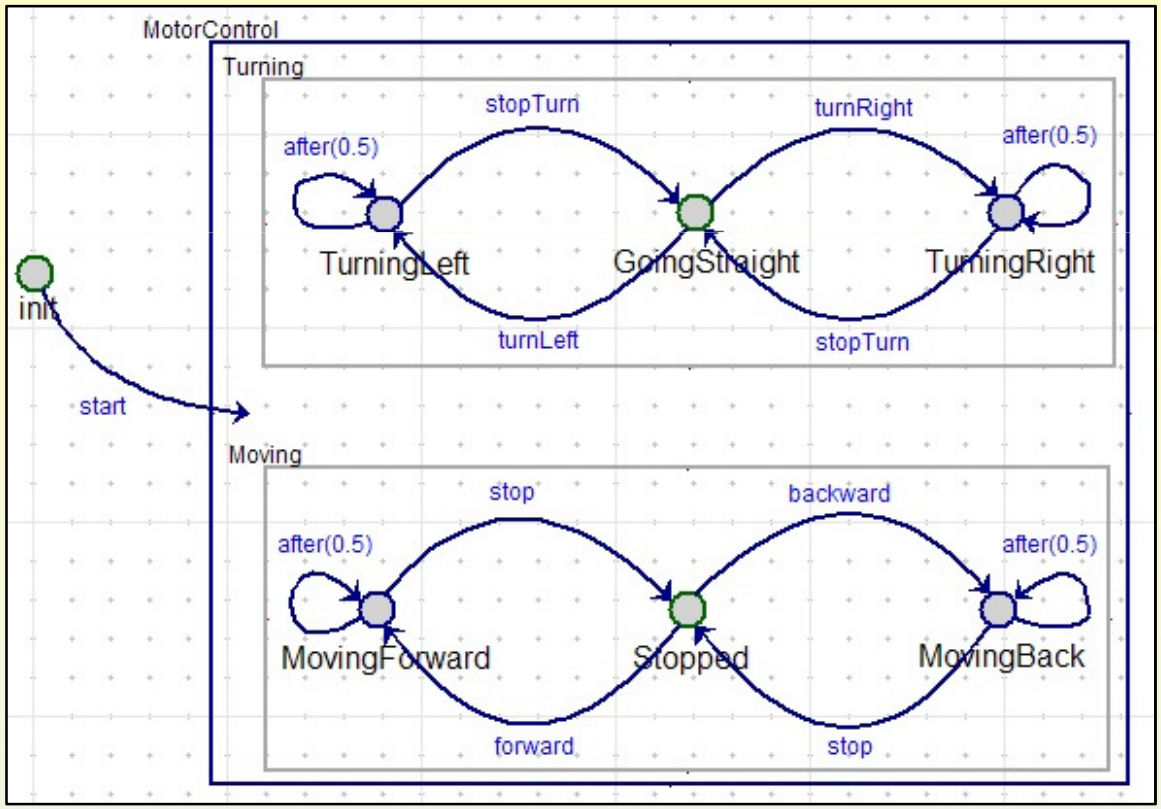
Designing the tank components' statecharts

- TurretTankMovementCoordinator (coordinators)



Designing the tank components' statecharts

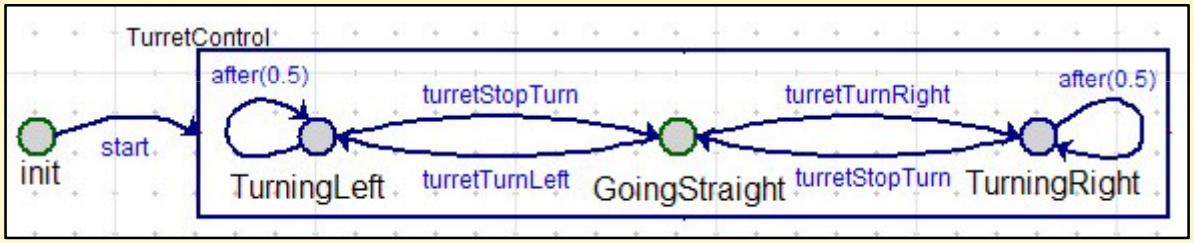
- MotorControl (actuator)



Statecharts modelling of Tank Wars

Designing the tank components' statecharts

- TurretControl (actuator)



Overview

- Recap
- The static part: the controller
- Designing the tank components' statecharts
- Recreating the simulation environment
- Summary

Recreating the simulation environment

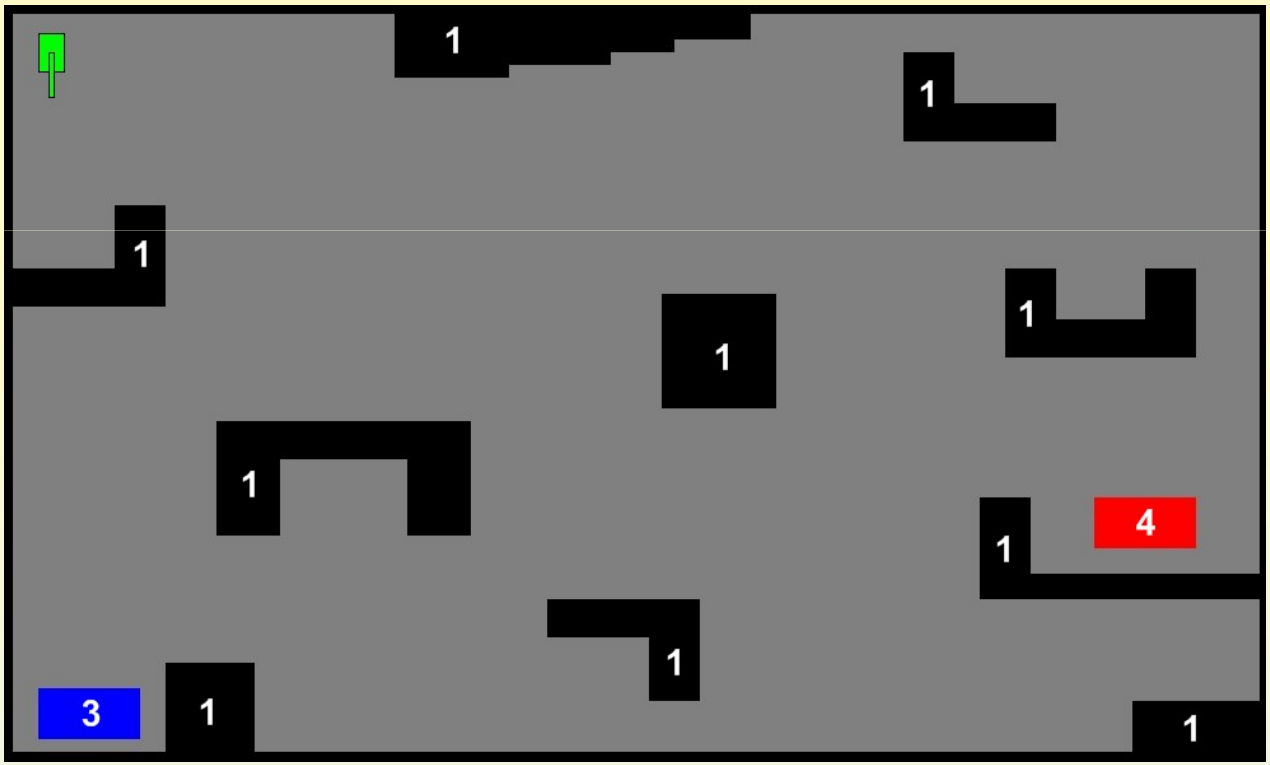
- Time-sliced to event-based: a lot of work!
- Implement a very simple simulation environment instead of using EA Tank Wars environment
- Use Python instead of C++
- Translate EA Tank Wars header files into Python
  - ai
  - coord
  - visor

Recreating the simulation environment

- Using Tkinter to draw world map and tank

Recreating the simulation environment

- World map defined using numbers in a plain text file



Recreating the simulation environment

- Moving the tank (translation)

```
def translatePoint(point, dx, dy):
      x = point.x + dx
      y = point.y + dy
      return Coord2D(x, y)
```

- Turning the tank and the turret (rotation)

```
def rotatePoint(point, origin, angle):
      x = origin.x + ((point.x - origin.x) *
               cos(angle) - (point.y - origin.y) * sin(angle))
      y = origin.y + ((point.x - origin.x) *
               sin(angle) + (point.y - origin.y) * cos(angle))
      return Coord2D(x, y)
```

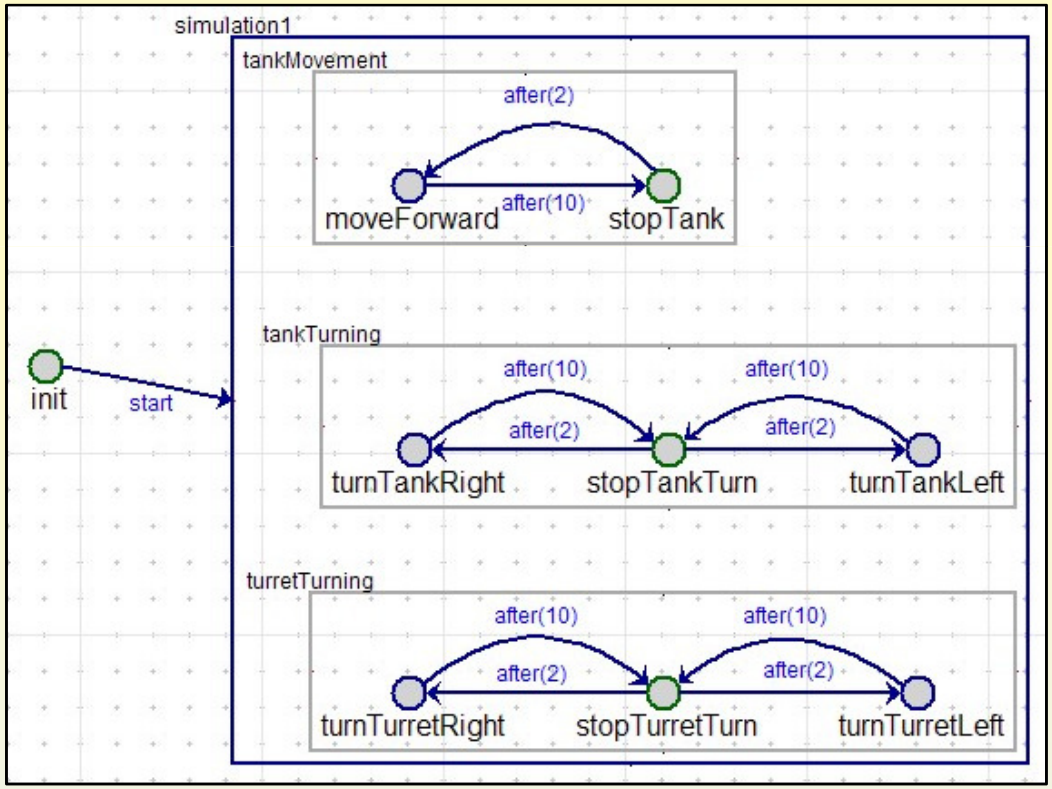
Recreating the simulation environment

- The simulation is also described with a statechart

```
root = Tk()
root.withdraw()
window = newWindow(root)

fr = FileReader.FileReader("input2.txt")
fr.loadFile()

dr = Drawer.Drawer(root, window, fr.lines, 40, 40)
ctl = TankController.TankController(dr)

sim = simulation.simulation()
sim.initModel()
sim.event("start", ctl)

root.mainloop()
```

Recreating the simulation environment

- Simulation example:



Overview

- Recap
- The static part: the controller
- Designing the tank components' statecharts
- Recreating the simulation environment
- Summary

Summary

- Succeeded:
  - Model tank's AI at a higher level of abstraction by using event-based formalism: statecharts
  - Necessity to use controller class to link tank's components and allow communication between them: "*narrow cast*"
  - Implementing a very simple environment for simulation instead of using EA Tank Wars
  - Describing simulation with a statechart too
- Not succeeded (yet):
  - Perceiving environment through the sensors (feeding the tank with input: obstacles, enemies, etc.)

Thank you!