Model-based Prototyping of Real-time Systems With PapyrusRT and Unity

Michal Pasternak
When writing code for physical systems we like to see it control our system.

This typically requires a lot of...
Solution: Simulate the system and environment

Simulations typically require less:

Automating simulation creation reduces:
A prototyping tool which will allow for quick editing, configuring and generation of a 3D simulation environment that can easily be interfaced with for controlling and monitoring objects in the simulation.
Using the prototyping tool

Create objects from pre defined meta-objects

```java
Object Spirit : Rover {
    Action setForwardPower (amount : int) return (){
        LFmotor = amount
        RFmotor = amount
    }

    Action getPosString () return (position : string){
        position = posX","+posY","+posZ
    }

    config {
        network = true
        sizeX = 1
        sizeY = 1
        sizeZ = 1
        posX = 1
        posY = 1
        posZ = posY
    }
}

Object rock : Generic {
    Action getXY () return (position : string){
        position = posX","+posY
    }

    config{
        posX = 10
        posY = 10
        posZ = 1
        size = 1
        mesh = "Icosphere_001"
        texture = "Rock6"
        model = "RockSet"
        mass = 1000
    }
}
```
Using the prototyping tool

Configure Simulation and set up communication channels

```c
Env simulation {
    Instance plane : land
    Instance obstacle1: rock
    Instance obstacle2: rock
    Instance rover1: Spirit

    Channel control1 direction inout type TCP (port : 8886) assign rover1
    Channel monitor direction inout type TCP (port : 8887) assign obstacle1 obstacle2
}
```

The DSL generates the simulator and a control library for Papyrus

[Diagram showing relationships between DSL, Simulation .exe, Control Library, and Control Logic]
Using the prototyping tool

Now as the simulation runs, the rover can be controlled by sending the defined commands through channel “control1”.

The rocks are monitored on the channel: “monitor”
Interfacing with the simulation

User writes the control logic to be tested in his language of choice, or with Papyrus using the generated model library.

For Example:

A monitoring program to check obstacle position over time, to detect a collision.

A rover control script that moves the rover and (hopefully) does not crash into the obstacles.
Interfacing with the simulation

Generated Rover Simulation

Communication over Channel “monitor”

Communication over Channel “control1”

“Rover Control” program

“Rock Monitor” program

All three programs could run on separate machines on the network, or on the same computer.
Summary

- Quickly set up a 3D environment

- Add any 3D object to the simulation.

- Customizable commands and communication

- Control logic is written in any language

- Multiple programs can interface with simulation simultaneously
TIME FOR QUESTIONS