Introduction to Game Programming and Robotics

Unit # 7

Acknowledgement

- Most of the example/material presented in this presentation is taken from tutorials provided by Microsoft or from the materials provided by HellpApps.com website.

Exclusivity

- Exclusivity is one of the more advanced VPL concepts, but something that you must understand so you can avoid unexpected behavior.
- how VPL decides which control flows are exclusive (i.e. which control flows will **not** be executed concurrently).
- A common example of where it becomes essential is when you update a variable in your program that is also referred to in other parts of the program.

Exclusivity (Cont'd)

- Consider a counter that is incremented from time to time from two different places in the program.
- At the lowest level inside the computer it is necessary to read (get) the value of the variable, add one to it, and then store (set) it back into memory.
- Assume that the first code path has just read the value, but before it can write the result back into memory the second code path updates the value.
- Now when the first code path stores a new value it will overwrite what the second code path has just written. The result is that the counter will be one less than it should be because the first code path was using *stale* information.

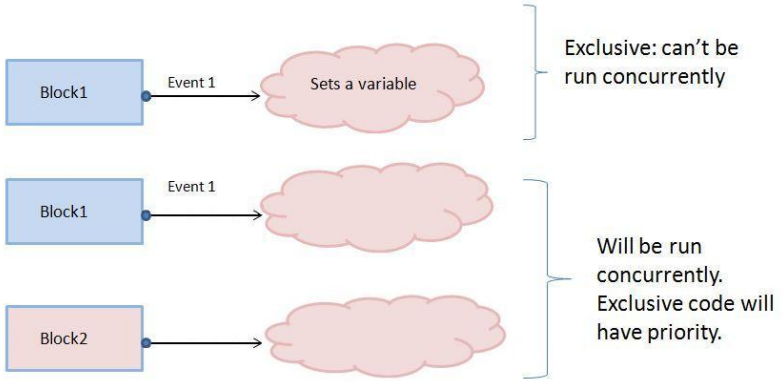
Atomic Operation

- To solve this problem, each code path must have exclusive access to the counter until it has completed the increment operation.
- This is referred to as an *atomic* operation because it is treated as indivisible even though it involves several steps: read value, increment, write value.
- No other code can alter the value of the counter until the increment operation is finished. The problem of simultaneous updates has been eliminated.

Concurrent vs. Exclusive Flow

- Consider the case where notifications arrive from a robot service.
- One notification might be about a bumper being pressed and another from a range sensor.
- Clearly different notifications from the same activity/service can be quite independent. If one of these control flows sets a variable, it will be exclusive, and cannot run concurrently with the other control flow.

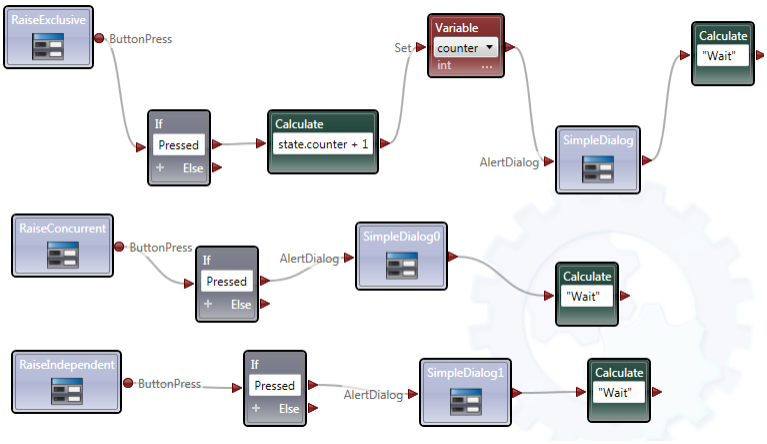
Concurrent vs. Exclusive Flow (Cont'd)



Concurrent vs. Exclusive Flow (Cont'd)

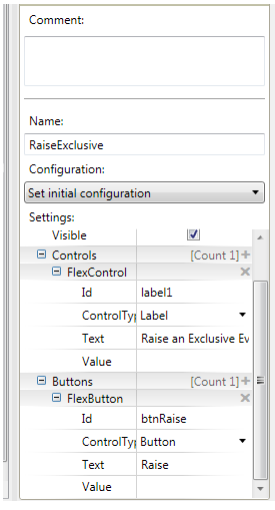
- Exclusive control flows have priority when it comes to execution if there are multiple flows waiting to execute.
- However, an exclusive flow cannot interrupt a concurrent flow, or vice versa.
- It is important not to have exclusive control flows that run for a long period of time blocking the execution of all other flows.
- While the execution of other flows is blocked, it is not possible to do a **get** on the state of the service to view the variables. For example, you should not in general call **Wait** on a **Timer** in an exclusive flow.

Exclusivity Example



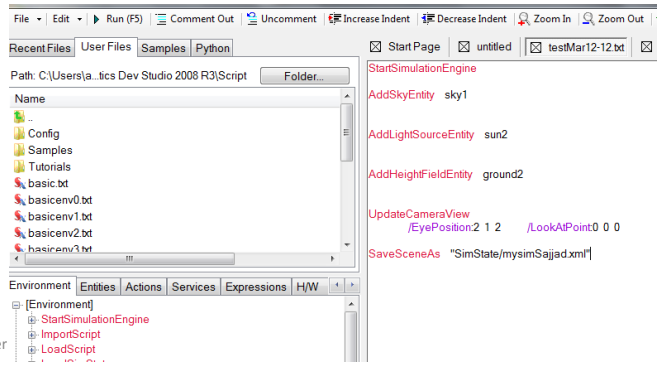
Exclusivity Example (Cont'd)

- RaiseExclusive, RaiseConcurrent and RaiseIndependent use FlexibleDialog properties

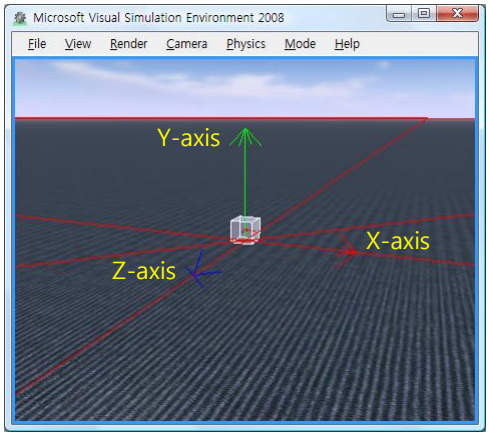


SPL

- SPL Script Editor
  - Under MRDS Installation Directory
- Make sure that you save the file in the default location (Script directory under MRDS)
- Filename shouldn't have any space



Understanding Simulation Environment



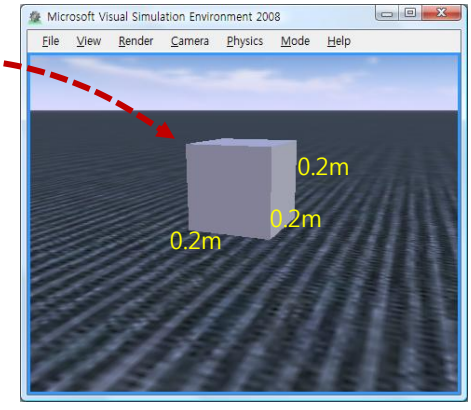
Dimensions of box shape

- Dimensions is denoted by meter scale unit
- For example, 0.5 means 0.5meter

```
StartSimulationEngine "SimState/basicsim.xml"

AddNewEntity  ent1    /Position:0 0.5 0

AddBoxShape
        /Dimensions:0.2 0.2 0.2
        /Mass:0.1
```



Change position of entity

- Change the position as 1m, 0.1m, 0m for x-axis, y-axis, and z-axis respectively

Start Page | Script_Y2009.M12.D09.txt

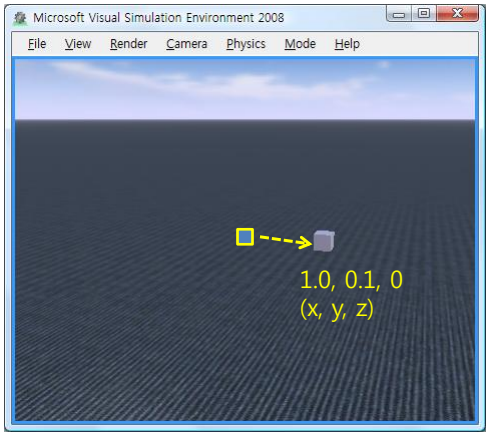
```
StartSimulationEngine "SimState/basicsim.xml"

AddNewEntity  ent1    /Position:1 0.1 0

AddBoxShape
        /Dimensions:0.2 0.2 0.2
        /Mass:0.1
```

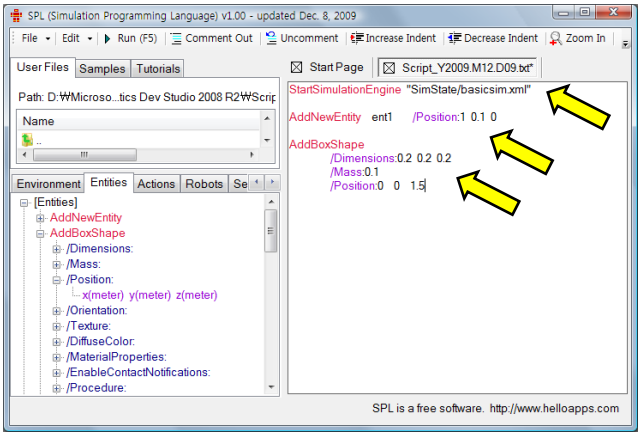
Change position of entity

- Save and run script



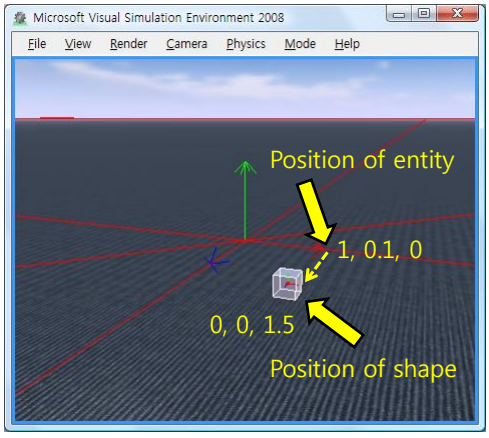
Change the position of shape

- Modify each position and dimensions as follows



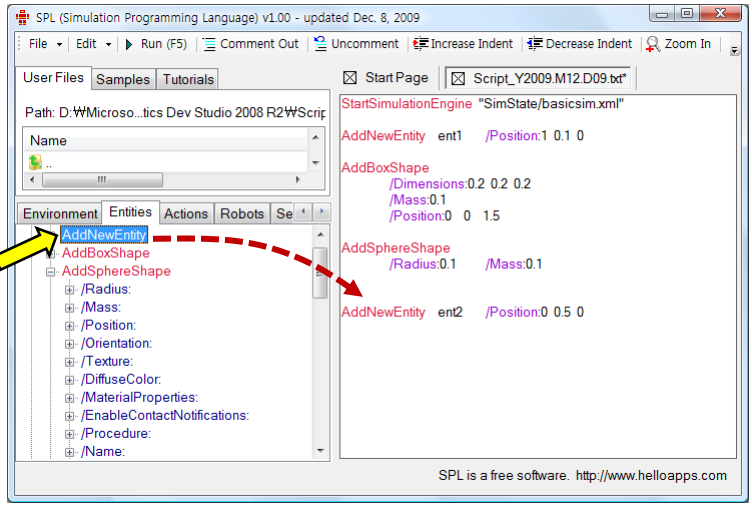
Change the position of shape

- Save and run script



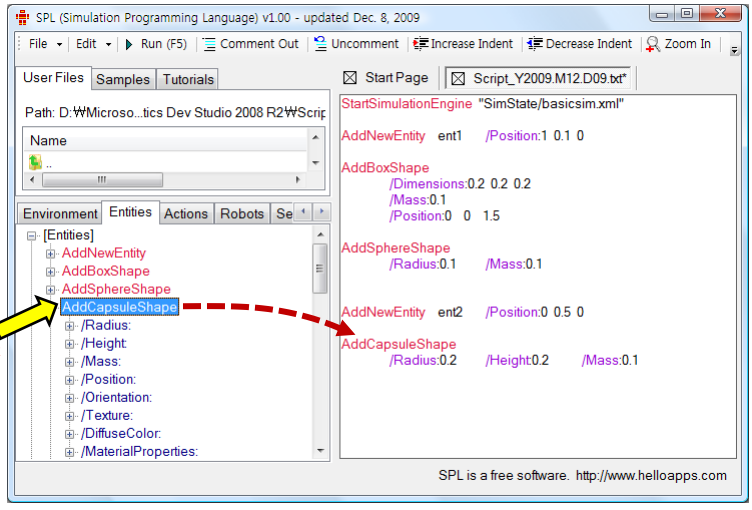
Add multi-entities

- Add an another “AddNewEntity” command



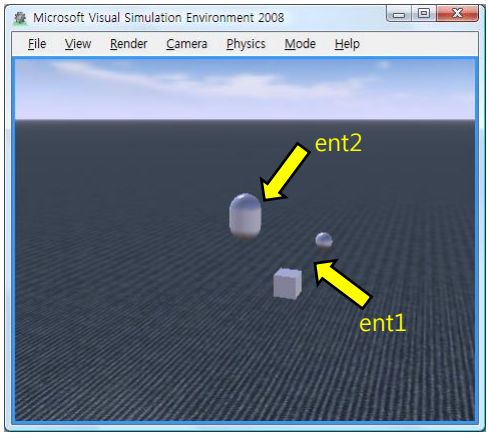
Add multi-entities

- Add “AddCapsuleShape” command



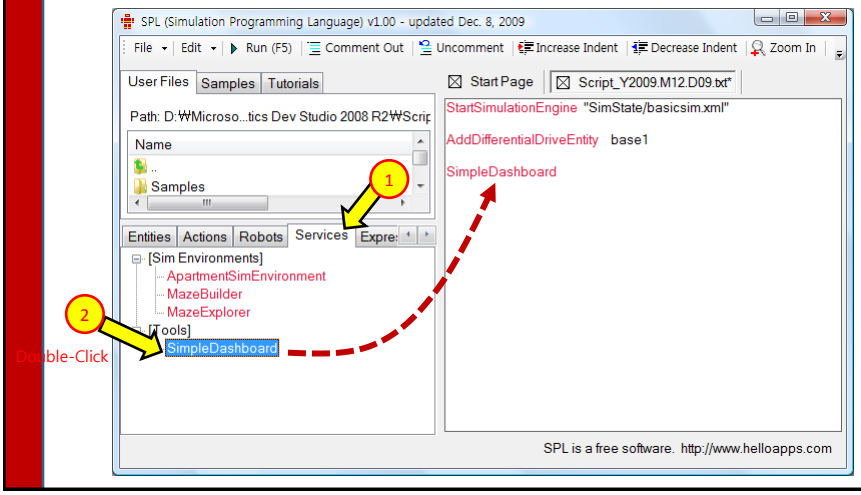
Add multi-entities

- Save and run script



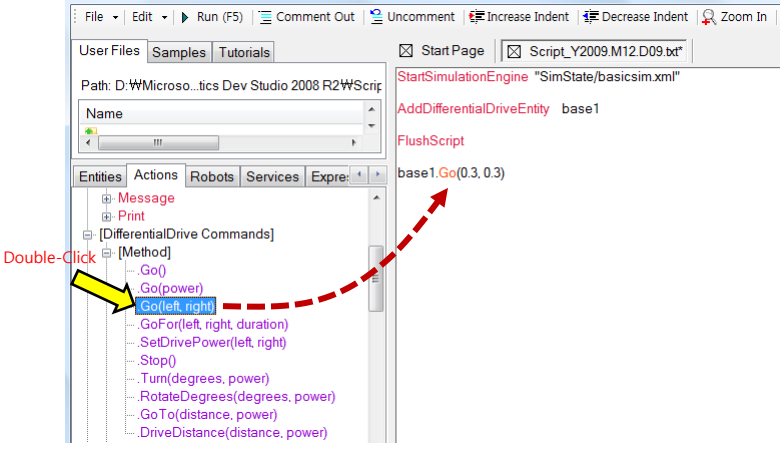
Drive DifferentialDrive entity

- Select “Services” tab
- Add “SimpleDashboard” command by double-clicking



Drive DifferentialDrive entity with script

- Add “Go(left, right)” command by double-clicking



Drive DifferentialDrive entity with script

- Go Forwards commands
  - base1.Go()
  - base1.Go(0.5)
  - base1.Go(0.5, 0.5)
  - base1.SetDrivePower(0.5, 0.5)

Drive DifferentialDrive entity with script

- Turn commands
  - base1.Go(-0.5, 0.5)
  - Base1.Go(0.2, 0.5)
  - base1.SetDrivePower(-0.5, 0.5)
  - base1.Turn(90, 0.3) //turn 90 degrees with 0.3 power
  - base1.RotateDegrees(90, 0.3)

Drive DifferentialDrive entity with script

- Go Distance commands
  - base1.GoTo(2.0, 0.5) //Go forwards 2.0m with 0.5 power
  - base1.DriveDistance(2.0, 0.5)

Drive DifferentialDrive entity with script

- Go Duration commands
  - base1.GoFo(0.5, 0.5, 1000)
    - Go with 0.5 power during 1000 milliseconds
    - After duration time, robot keeps its power