Presented by Orbital Research, Inc.

**orbitalresearch.com**

---

## Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

# Section 1. Example SWEEP Simulation

## Scenario

A problem the arises frequently in real-world scenarios is that of gathering a group of objects into a pile. In nature, this type of behavior can be seen the way ants separate their food from their waste and their dead. A potential military application of a gathering behavior would be for clearing a minefield by disarming and deactivating all the mines in a given area.

The scenario that is be used in this tutorial extends the object gathing scenario. There will be two types of objects in the environment, mines and garbage. There are two goals for the scenario:

1. Gather all the garbage into the dumpster
2. Disarm all the mines in the area

Each agent is able to sense when they are in the area of either garbage or a mine. Additionaly, each agent has specific abilities related both to garbage and mines. With regards to garbage, each agent can pick-up and put down garbage, but can only cary one piece of garbage at a time. As for the mines, the disarming process is a two-step procedure, thus to successfully disarm a mine, two separate actions must be performed. Each mine disarming action has the side-effect that once performed, the performing agent is then disabled for the duration of the simulation.

The final aspect of the simulation will be a competitive one. There will be two swarms, each with equal abilities and algorithms. The two swarms will compete against each other, try to collect more garbage and disarm more mines than the other swarm. The implementation of unique algorithms for each competing swarm will be let as an exercise for the reader.

---

## <Environments>

**Description**
The *<Environments>* section describes the different environments that will be used in the simulation.

For the simulation, a grid-based environment named "world" is used. A grid-based environment is more than adequate for this simulation. The environment is defined to have a size of 50x50. The garbage objects will be represented by the integer value 1, and the mine values will be represented by the integer value 4 for an armed mine, 3 for a partially-disarmed mine, and 2 for a completely disarmed mine. There will be 50 garbage objects and 10 mine objects randomly placed throughout the environment.

```
<environments>
    <environment name="world">
        <builder class="standard.environment.EnvironmentBuilder"/>
        <model class="grid.environment.SafeGridModel">
            <attributes>
                <attribute name="rows" value="50"/>
                <attribute name="cols" value="50"/>
            </attributes>
            <initializers>
                <initializer class="grid.environment.RandomInitialization">
                    <builder class="grid.environment.RandomInitializationBuilder"/>

                    <seed value="3141592654"/>

                    <!-- Garbage Object-->
                    <object value="1" number="50"/>
                    <!-- Mine Object -->
                    <object value="4" number="10"/>
                </initializer>
            </initializers>
            <updates></updates>
        </model>
    </environment>
</environments>
```

---

## <Model>

**Description**
The <*Model*> section describes the different types of *physical* agents that will be used in a simulation. The main purpose of an agent model, sometimes referred to as an *avatar*, is to define the basic behaviors and senses available with respect to the operating environment.

For example, in this scenario, the available behaviors and sensors are related to the garbage objects and the mine objects that are scattered throughout the environment. The basic sensors are:

- on-garbage
- on-dumpster
- near-garbage
- on-armed-mine
- on-partially-disarmed-mine
- near-mine

- holding-garbage

And the basic actions are:

- pick-up-garbage
- put-down-garbage
- disarm-mine-stage-1
- disarm-mine-stage-2
- move-random
- move-towards-garbage
- move-towards-dumpster
- move-towards-mine

All the the sensors and actions are implemented using the basic actions and sensors defined in the core SWEEP package. Though custom defined actions and sensors can be implemented in Java, this approach was not taken so as to show how a non-programmer might use SWEEP.

The <attributes> section establishes a set of variables that the different sensors and actions can access. Using attributes in this way makes changing parameter values much easier. For example, without the attribute variables, changing all of the sensor ranges would require one to change the value in each sensor definition. But using an attribute variable allows a user to make a single change and have that change propogate throughout.

```
<models>
    <model name="avatar" class="grid.model.GridAvatar">
        <builder class="standard.model.ModelBuilder"/>

        <!-- attributes define model parameters -->
        <attributes>
            <!-- Basic attributes -->
            <attribute name="empty" value="0"/>
            <attribute name="environment" value="world"/>
            <attribute name="holdingGarbage" value="false"/>


            <!-- Parameters for garbage actions/sensors -->
            <attribute name="garbage.grid" value="world"/>
            <attribute name="garbage.value" value="1"/>
            <attribute name="garbage.downvalue" value="6"/>
            <attribute name="garbage.range.on" value="0"/>
            <attribute name="garbage.range.near" value="5"/>

            <!-- Paramters for mine actions/sensors -->
            <attribute name="mine.grid" value="world"/>
            <attribute name="mine.value.armed" value="4"/>
            <attribute name="mine.value.partial" value="3"/>
            <attribute name="mine.value.disarmed" value="2"/>
            <attribute name="mine.range.on" value="0"/>
            <attribute name="mine.range.near" value="5"/>
        </attributes>

        <!-- define available sensors -->
        <sensors>
            <!-- Sensor: true when on a piece of garbage -->
            <sensor name="on-garbage" class="grid.model.sensor.FilteredValueSensor">
                <attributes>
                    <attribute name="environment" value="garbage.grid"/>
                    <attribute name="range"       value="garbage.range.on"/>
                    <attribute name="targetValue" value="garbage.value"/>
                </attributes>
            </sensor>

            <!-- Sensor: true when in range of garbage -->
            <sensor name="near-garbage" class="grid.model.sensor.FilteredValueSensor">
                <attributes>
                    <attribute name="environment" value="garbage.grid"/>
                    <attribute name="range" value="garbage.range.near"/>
                    <attribute name="targetValue" value="garbage.value"/>
                    <attribute name="varName" value="garbage.location"/>
                </attributes>
            </sensor>

            <!-- Sensor: true when holding garbage -->
            <sensor name="holding-garbage" class="sweep.model.sensor.InternalBooleanSensor
                <attributes>
                    <attribute name="attrName" value="holdingGarbage"/>
                </attributes>
            </sensor>

            <!-- Sensor: true when on a fully armed mine -->
            <sensor name="on-armed-mine" class="grid.model.sensor.FilteredValueSensor">
                <attributes>
                    <attribute name="environment" value="mine.grid"/>
                    <attribute name="range" value="mine.range.on"/>
                    <attribute name="targetValue" value="mine.value.armed"/>
                </attributes>
            </sensor>

            <!-- Sensor: true when on a partially disarmed mine -->
            <sensor name="on-partially-disarmed-mine" class="grid.model.sensor.FilteredVal
                <attributes>
                    <attribute name="environment" value="mine.grid"/>
                    <attribute name="range" value="mine.range.on"/>
                    <attribute name="targetValue" value="mine.value.partial"/>
                </attributes>
            </sensor>

            <!-- Sensor: true when on the dumpster -->
            <sensor name="on-dumpster" class="grid.model.sensor.LocationSensor">
                <attributes>
                    <attribute name="environment" value="garbage.grid"/>
                    <attribute name="location" value="const:10,10"/>
                </attributes>
            </sensor>
```

```
            <!-- Sensor: true when near a (fully|partially) armed mine -->
            <sensor name="near-mine" class="grid.model.sensor.FilteredValueSensor">
                <attributes>
                    <attribute name="environment" value="mine.grid"/>
                    <attribute name="range" value="mine.range.near"/>
                    <attribute name="targetValue" value="mine.value.armed,mine.value.parti
                    <attribute name="varName" value="mine.location"/>
                </attributes>
            </sensor>
        </sensors>

        <actions>
            <!-- Action: if on a piece of garbage, pick it up -->
            <action name="pick-up-garbage" builder="sweep.model.action.ConditionalActionBu
                <if>
                    <condition sensor="on-garbage" value="true"/>
                    <condition state="holdingGarbage" value="false"/>
                    <then>
                        <action name="pick-up-garbage-helper"/>
                        <action name="set-holding-garbage"/>
                    </then>
                </if>
            </action>

            <!-- Action: if holding garbage and on dumpster, put it down -->
            <action name="put-down-garbage" builder="sweep.model.action.ConditionalActionE
                <if>
                    <condition sensor="on-dumpster" value="true"/>
                    <condition state="holdingGarbage" value="true"/>
                    <then>
                        <!--<action name="put-down-garbage-helper"/>-->
                        <action name="reset-holding-garbage"/>
                        <action name="increment-garbage-count"/>
                    </then>
                </if>
            </action>

            <!-- Action: pick up a piece of garbage -->
            <action name="pick-up-garbage-helper" class="grid.model.action.SetAction">
                <attributes>
                    <attribute name="environment" value="garbage.grid"/>
                    <attribute name="value" value="empty"/>
                </attributes>
            </action>

            <!-- Action: put down a piece of garbage -->
            <action name="put-down-garbage-helper" class="grid.model.action.SetAction">
                <attributes>
                    <attribute name="environment" value="garbage.grid"/>
                    <attribute name="value" value="garbage.downvalue"/>
                </attributes>
            </action>

            <!-- Action: set the holding garbage flag -->
            <action name="set-holding-garbage" class="sweep.model.action.SetStateVariableA
                <attributes>
                    <attribute name="flag" value="holdingGarbage"/>
                    <attribute name="value" value="true"/>
```

```
    </attributes>
</action>

<!-- Action: reset the holding garbage flag -->
<action name="reset-holding-garbage" class="sweep.model.action.SetStateVariabl
    <attributes>
        <attribute name="flag" value="holdingGarbage"/>
        <attribute name="value" value="false"/>
    </attributes>
</action>

<!-- Action: increment the garbage count -->
<action name="increment-garbage-count" class="sweep.model.action.IncrementCoun
    <attributes>
        <attribute name="name" value="garbage-counter"/>
    </attributes>
</action>

<!-- Action: if on a mine, disarm the first stage -->
<action name="disarm-mine-stage-1" builder="sweep.model.action.ConditionalActi
    <if>
        <condition sensor="on-armed-mined" value="true"/>
        <then>
            <action name="disarm-mine-stage-1-helper"/>
            <action name="set-disarmed-phase-1"/>
            <action name="disable-agent"/>
        </then>
    </if>
</action>

<!-- Action: if on a mine, disarm the second stage -->
<action name="disarm-mine-stage-2" builder="sweep.model.action.ConditionalActi
    <if>
        <condition sensor="on-partial-mine" value="true"/>
        <then>
            <action name="disarm-mine-stage-2-helper"/>
            <action name="set-disarmed-phase-2"/>
            <action name="disable-agent"/>
        </then>
    </if>
</action>

<!-- Action: First stage of disarming the mine -->
<action name="disarm-mine-stage-1-helper" class="grid.model.action.SetAction">
    <attributes>
        <attribute name="environment" value="mine.grid"/>
        <attribute name="value" value="mine.value.partial"/>
    </attributes>
</action>

<!-- Action: Second stage of disarming the mine -->
<action name="disarm-mine-stage-2-helper" class="grid.model.action.SetAction">
    <attributes>
        <attribute name="environment" value="mine.grid"/>
        <attribute name="value" value="mine.value.disarmed"/>
    </attributes>
</action>
```

```
<!-- Action: set the phase 1 disarming flag -->
<action name="set-disarmed-phase-1" class="sweep.model.action.IncrementCounter
    <attributes>
        <attribute name="name" value="disarmed-phase-1"/>
    </attributes>
</action>
<!--
<action name="set-disarmed-phase-1" class="sweep.model.action.SetStateVariable
    <attributes>
        <attribute name="flag" value="disarmed-phase-1"/>
        <attribute name="value" value="true"/>
    </attributes>
</action>  -->

<!-- Action: set the phase 2 disarming flag -->
<action name="set-disarmed-phase-2" class="sweep.model.action.IncrementCounter
    <attributes>
        <attribute name="name" value="disarmed-phase-2"/>
    </attributes>
</action>
<!--
<action name="set-disarmed-phase-2" class="sweep.model.action.SetStateVariable
    <attributes>
        <attribute name="flag" value="disarmed-phase-2"/>
        <attribute name="value" value="true"/>
    </attributes>
</action>-->

<!-- Action: disable the agent for the rest of the sim -->
<action name="disable-agent" class="sweep.model.action.DisableAction"/>

<!-- Action: move in a random direction -->
<action name="move-random" class="grid.model.action.MoveRandomAction">
    <attributes>
        <attribute name="environment" value="environment"/>
    </attributes>
</action>

<!-- Action: move towards the dumpster -->
<action name="move-towards-dumpster" class="grid.model.action.MoveTowardsActio
    <attributes>
        <attribute name="environment" value="garbage.grid"/>
        <attribute name="location" value="const:10,10"/>
    </attributes>
</action>

<!-- Action: move towards the closest piece of garbage -->
<action name="move-towards-garbage" class="grid.model.action.MoveTowardsAction
    <attributes>
        <attribute name="environment" value="garbage.grid"/>
        <attribute name="location" value="av:garbage.location"/>
    </attributes>
</action>

<!-- Action: move towards the closest mine -->
<action name="move-towards-mine" class="grid.model.action.MoveTowardsAction">
    <attributes>
        <attribute name="environment" value="mine.grid"/>
```

```
                    <attribute name="location" value="av:mine.location"/>
                </attributes>
            </action>
        </actions>

        <initializers>
            <initializer name="random-placement" class="grid.model.RandomPlacementInitiali
        </initializers>
    </model>
</models>
```

---

## <Controller>

**Description**
The <*Controller*> section defines the basic agent logic. This section is where the swarm algorithm itself is programmed. The controller uses sensor inputs to make decisions, and from these decisions the desired behaviors are selected.

SWEEP uses a state machine model by default to encode agent logic. It is possible for a develper to implement other types of controllers besides state machines. Examples of other controllers include pure Java controllers, LISP or PROLOG evaluators, embedded Jython, or interactive control through some input method (keyboard, joystick, etc).

The state machine has a very simple setup. Each state is defined as a collection of transitions. Each transition has a boolean logic condition, that if met, enables a particular set of behaviors. Also, each transitions defines the appropriate next state to go to if the transition fires. Each state also has a default rule that is fired if not other rule is activated.

For the example scenario used here, the state machine uses three states. The first state manages the searching of the environment. The second state handles the collection of garbage. And the third state manages the destruction of mines. Below is a psuedo-state-machine of the algorithm used in this example.

**State: Search**
- if *near-garbage* then *move-towards-garbage* and goto **collect-garbage**
- if *near-mine* then *move-towards-mine* and goto **disarm-mine**
- default: *move-random* and goto **search**

**State: Collect-Garbage**
- if not *holding-garbage* and *on-garbage* then *pick-up-garbage* and goto **collect-garbage**
- if not *holding-garbage* and *near-garbage* then *move-towards-garbage* and goto **collect-garbage**

- if *holding-garbage* and not *on-dumpster* then *move-towards-dumpster* and goto **collect-garbage**
- if *holding-garbage* and *on-dumpster* then *put-down-garbage* and goto **search**
- default: *move-random* and goto **search**

**State: Disarm-Mine**

- if *on-armed-mine* then *disarm-phase-1* and goto **disarm-mine**
- if *on-partially-armed-mine* then *disarm-phase-2* and goto **search**
- if *near-mine* then *move-towards-mine* and goto **disarm-mine**
- default: *move-random* and goto **search**

```
<controllers>
    <controller name="basic-state-machine" format="state">
        <builder class="sweep.controller.FSABuilder"/>
        <initialState name="searching"/>
        <states>

            <!-- State: controls the search beahviors -->
            <state name="searching">
                <!-- Rule: if in range of garbage, go into collecting mode -->
                <transition nextState="collect-garbage">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="near-garbage" value="true"/>
                    </rule>
                    <action name="move-towards-garbage"/>
                </transition>

                <!-- Rule: if in range of a mine, go into disarming mode -->
                <transition nextState="disarm-mine">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="near-mine" value="true"/>
                    </rule>
                    <action name="move-towards-mine"/>
                </transition>

                <!-- Rule: otherwise, keep searching randomly -->
                <default nextState="searching">
                    <action name="move-random"/>
                </default>
            </state>

            <!-- State: controls the collection of garbage -->
            <state name="collect-garbage">
                <!-- Rule: if on a piece of garbage, pick it up -->
                <transition nextState="collect-garbage">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="holding-garbage" value="false"/>
                        <sensor name="on-garbage" value="true"/>
                    </rule>
                    <action name="pick-up-garbage"/>
                </transition>

                <!-- Rule: if near garbage, move towards it -->
                <transition nextState="collect-garbage">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="holding-garbage" value="false"/>
                        <sensor name="near-garbage" value="true"/>
                    </rule>
                    <action name="move-towards-garbage"/>
                </transition>

                <!-- Rule: if holding garbage, move towards the dumpster -->
                <transition nextState="collect-garbage">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="holding-garbage" value="true"/>
                        <sensor name="on-dumpster" value="false"/>
                    </rule>
                    <action name="move-towards-dumpster"/>
                </transition>

                <!-- Rule: if holding garbage and on the dumpster, dispose of the garbage
                <transition nextState="searching">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="holding-garbage" value="true"/>
                        <sensor name="on-dumpster" value="true"/>
                    </rule>
                    <action name="put-down-garbage"/>
                </transition>

                <!-- Rule: go back to searching -->
                <default nextState="searching">
                    <action name="move-random"/>
                </default>
            </state>

            <!-- State: controls the disarming of mines -->
            <state name="disarm-mine">
                <!-- Rule: if on an armed mine, perform the first disarming step -->
                <transition nextState="disarm-mine">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="on-armed-mine" value="true"/>
                    </rule>
                    <action name="disarm-mine-stage-1"/>
                </transition>

                <!-- Rule: if on a partially disarmed mine, finish the job -->
                <transition nextState="searching">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="on-partially-disarmed-mine" value="true"/>
                    </rule>
                    <action name="disarm-mine-stage-2"/>
                </transition>
```

```
                <!-- Rule: if near a mine, move towards it -->
                <transition nextState="disarm-mine">
                    <rule logical="and">
                        <builder class="sweep.controller.SensorRuleBuilder"/>
                        <sensor name="near-mine" value="true"/>
                    </rule>
                    <action name="move-towards-mine"/>
                </transition>

                <!-- Rule: go back to searching -->
                <default nextState="searching">
                    <action name="move-random"/>
                </default>
            </state>
        </states>
    </controller>
</controllers>
```

---

## <Swarm>

**Description**
The <*Swarm*> section defines the construction of the swarms to be used. A swarm is defined as being a collection of entities. For our purposes, an entity is a collection consisting of an agent, an avatar/model, and a controller. A swarm can have any number of entities. The specific size of the swarm(s) is specified in the <*Main*> section.

For the example, two swarm types have been defined. Though their composition is identical, the separate definition effectively create two distinct swarm types. For each swarm, the entity is composed of a default agent, and the model and controller previously defined.

```
<swarms>
        <swarm name="swarm-A">
                <builder class="standard.swarm.SwarmBuilder"/>
                <entity name="swarm-A-entity">
                        <agent type="agent"/>
                        <controller type="basic-state-machine"/>
                        <model type="avatar"/>
                </entity>
        </swarm>

        <swarm name="swarm-B">
                <builder class="standard.swarm.SwarmBuilder"/>
                <entity name="swarm-B-entity">
                        <agent type="agent"/>
                        <controller type="basic-state-machine"/>
                        <model type="avatar"/>
                </entity>
        </swarm>
</swarms>
```

## <Probe>

**Description**
The <*Probe*> section defines points in the simulation where data can be extracted or injected. Probes can be used for basically anything, including: debugging, changing internal variables, monitoring single/multiple agent(s), and facilitating user interactivity.

For this example, two probes have been defined. The first probe (CounterProbe) is used to set an upper limit of 500 iterations on the runtime of the simulation. The second probe (SummaryProbe) is used to collect and summarize statistical data about the simulation; more specifically the number of garbage objects collected and the number of mines disabled.

```
<probes>
    <probe name="terminator">
        <builder class="sweep.probe.CounterProbeBuilder"/>
        <iterations num="500"/>
    </probe>

    <probe name="summarizer">
        <builder class="sweep.probe.SummaryProbeBuilder"/>
        <sum swarm="my-swarm-A" var="garbage-counter"/>
        <sum swarm="my-swarm-A" var="disarmed-phase-1"/>
        <sum swarm="my-swarm-A" var="disarmed-phase-2"/>
        <sum swarm="my-swarm-B" var="garbage-counter"/>
        <sum swarm="my-swarm-B" var="disarmed-phase-1"/>
        <sum swarm="my-swarm-B" var="disarmed-phase-2"/>
        <screen/>
    </probe>
</probes>
```

## <Agent> and <Info>

The sections <*Agent*> and <*Info*> are not currently used in this version of SWEEP.

## <Main>

**Description**

The <*Main*> section serves a similar purpose to the main() method in C or Java, that being to handle the high-level setup and execution of the simulation. This section defines many important features of the simulation including swarm sizes, environmental parameters, and the specific probes to be used.

For 50 garbage object and 10 mine objects, two swarms of size 30 should be sufficient to address the problem in 500 timesteps. Aside from declaration, no specific actions need be taken with regards to the environment or probes.

**N.B.** <*Main*> is not yet able to fully customize every aspect of the simulation at runtime due certain implementation limitations that will require rearchitecting to address.

```
<main>
    <info name="about" type="info-type1"/>

    <swarm name="my-swarm-A" type="swarm-A">
        <entity type="swarm-A-entity" number="10"/>
    </swarm>

    <swarm name="my-swarm-B" type="swarm-B">
        <entity type="swarm-B-entity" number="10"/>
    </swarm>

    <!-- should be able to override environment attributes here too -->
    <environment name="world" type="world"/>

    <probe name="probe" type="terminator"/>
    <probe name="summary" type="summarizer"/>
</main>
```

---

## Running the simulation

**Building SWEEP**
Building SWEEP requires that *make* be installed. From the SWEEP_WPAFB directory, type: *make*. This will handle compiling SWEEP.

In order to run the simulation graphically, first, from a command line, change to the SWEEP_WPAFB directory. From that directory, run the following command: *java -cp jars/dom4j-full.jar:bin sweep.viewer.Main simulation.xml*

Alternatively, if you are running this on a Linux-type platform (or have *make* installed on Windows), just execute the command *make run*, which is a small script that handles all the command-line parameters.

When you execute the simulation, you should see textual output similar to this

```
simulation.xml
...adding the required probes
...starting sweep
...opening ServerSocket on port 3333
...got a socket...
my-swarm-A:garbage-counter:20
my-swarm-A:disarmed-phase-1:2
my-swarm-A:disarmed-phase-2:3
my-swarm-B:garbage-counter:12
my-swarm-B:disarmed-phase-1:3
my-swarm-B:disarmed-phase-2:1
```

If there are problems running the simulation, make sure that your firewall software is not blocking port 3333, as the viewer and SWEEP communicate using the network.

---

## Full simulation XML file

```
<?xml version="1.0"?>

<simulation builder="standard.simulation.SimulationBuilder">

<main>
        <info name="about" type="info-type1"/>

        <swarm name="my-swarm-A" type="swarm-A">
                <entity type="swarm-A-entity" number="10"/>
        </swarm>

        <swarm name="my-swarm-B" type="swarm-B">
                <entity type="swarm-B-entity" number="10"/>
        </swarm>

        <!-- should be able to override environment attributes here too -->
        <environment name="world" type="world"/>

        <probe name="probe" type="terminator"/>
        <probe name="summary" type="summarizer"/>
</main>

<environments>
        <environment name="world">
                <builder class="standard.environment.EnvironmentBuilder"/>
                <model class="grid.environment.SafeGridModel">
                        <attributes>
                                <attribute name="rows" value="50"/>
                                <attribute name="cols" value="50"/>
                        </attributes>
                        <initializers>
                                <initializer class="grid.environment.RandomInitialization'
                                        <builder class="grid.environment.RandomInitializat
                                        <seed value="3141592654"/>

                                        <!-- Garbage Object-->
                                        <object value="1" number="50"/>
                                        <!-- Mine Object -->
                                        <object value="4" number="10"/>
                                </initializer>
                        </initializers>
                        <updates></updates>
                </model>
        </environment>
</environments>

<models>
        <model name="avatar" class="grid.model.GridAvatar">
                <builder class="standard.model.ModelBuilder"/>

                <!-- attributes define model parameters -->
                <attributes>
                        <!-- Basic attributes -->
                        <attribute name="empty" value="0"/>
                        <attribute name="environment" value="world"/>
                        <attribute name="holdingGarbage" value="false"/>


                        <!-- Parameters for garbage actions/sensors -->
                        <attribute name="garbage.grid" value="world"/>
                        <attribute name="garbage.value" value="1"/>
                        <attribute name="garbage.downvalue" value="6"/>
                        <attribute name="garbage.range.on" value="0"/>
                        <attribute name="garbage.range.near" value="5"/>

                        <!-- Paramters for mine actions/sensors -->
                        <attribute name="mine.grid" value="world"/>
                        <attribute name="mine.value.armed" value="4"/>
                        <attribute name="mine.value.partial" value="3"/>
                        <attribute name="mine.value.disarmed" value="2"/>
                        <attribute name="mine.range.on" value="0"/>
                        <attribute name="mine.range.near" value="5"/>
                </attributes>

                <!-- define available sensors -->
                <sensors>
                        <!-- Sensor: true when on a piece of garbage -->
                        <sensor name="on-garbage" class="grid.model.sensor.FilteredValueSe
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="range"       value="garbage.range
                                        <attribute name="targetValue" value="garbage.value
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when in range of garbage -->
                        <sensor name="near-garbage" class="grid.model.sensor.FilteredValue
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="range" value="garbage.range.near"
                                        <attribute name="targetValue" value="garbage.value
                                        <attribute name="varName" value="garbage.location'
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when holding garbage -->
                        <sensor name="holding-garbage" class="sweep.model.sensor.InternalE
                                <attributes>
                                        <attribute name="attrName" value="holdingGarbage"/
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when on a fully armed mine -->
                        <sensor name="on-armed-mine" class="grid.model.sensor.FilteredValu
                                <attributes>
                                        <attribute name="environment" value="mine.grid"/>
                                        <attribute name="range" value="mine.range.on"/>
                                        <attribute name="targetValue" value="mine.value.ar
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when on a partially disarmed mine -->
                        <sensor name="on-partially-disarmed-mine" class="grid.model.sensor
                                <attributes>
                                        <attribute name="environment" value="mine.grid"/>
                                        <attribute name="range" value="mine.range.on"/>
                                        <attribute name="targetValue" value="mine.value.pa
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when on the dumpster -->
                        <sensor name="on-dumpster" class="grid.model.sensor.LocationSensor
                                <attributes>
                                        <attribute name="environment" value="garbage.grid'
                                        <attribute name="location" value="const:10,10"/>
                                </attributes>
                        </sensor>

                        <!-- Sensor: true when near a (fully|partially) armed mine -->
                        <sensor name="near-mine" class="grid.model.sensor.FilteredValueSen
                                <attributes>
                                        <attribute name="environment" value="mine.grid"/>
                                        <attribute name="range" value="mine.range.near"/>
                                        <attribute name="targetValue" value="mine.value.ar
                                        <attribute name="varName" value="mine.location"/>
                                </attributes>
                        </sensor>
                </sensors>

                <actions>
                        <!-- Action: if on a piece of garbage, pick it up -->
                        <action name="pick-up-garbage" builder="sweep.model.action.Conditi
                                <if>
                                        <condition sensor="on-garbage" value="true"/>
                                        <condition state="holdingGarbage" value="false"/>
                                        <then>
                                                <action name="pick-up-garbage-helper"/>
                                                <action name="set-holding-garbage"/>
                                        </then>
                                </if>
                        </action>

                        <!-- Action: if holding garbage and on dumpster, put it down -->
                        <action name="put-down-garbage" builder="sweep.model.action.Condit
                                <if>
                                        <condition sensor="on-dumpster" value="true"/>
                                        <condition state="holdingGarbage" value="true"/>
                                        <then>
                                                <!--<action name="put-down-garbage-helper"
                                                <action name="reset-holding-garbage"/>
                                                <action name="increment-garbage-count"/>
                                        </then>
                                </if>
                        </action>

                        <!-- Action: pick up a piece of garbage -->
                        <action name="pick-up-garbage-helper" class="grid.model.action.Set
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="value" value="empty"/>
                                </attributes>
                        </action>

                        <!-- Action: put down a piece of garbage -->
                        <action name="put-down-garbage-helper" class="grid.model.action.Se
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="value" value="garbage.downvalue"/
                                </attributes>
                        </action>

                        <!-- Action: set the holding garbage flag -->
                        <action name="set-holding-garbage" class="sweep.model.action.SetSt
                                <attributes>
                                        <attribute name="flag" value="holdingGarbage"/>
                                        <attribute name="value" value="true"/>
                                </attributes>
                        </action>

                        <!-- Action: reset the holding garbage flag -->
                        <action name="reset-holding-garbage" class="sweep.model.action.Set
                                <attributes>
                                        <attribute name="flag" value="holdingGarbage"/>
                                        <attribute name="value" value="false"/>
                                </attributes>
                        </action>

                        <!-- Action: increment the garbage count -->
                        <action name="increment-garbage-count" class="sweep.model.action.I
                                <attributes>
                                        <attribute name="name" value="garbage-counter"/>
                                </attributes>
                        </action>

                        <!-- Action: if on a mine, disarm the first stage -->
                        <action name="disarm-mine-stage-1" builder="sweep.model.action.Con
```

```
        <if>
                <condition sensor="on-armed-mined" value="true"/>
                <then>
                        <action name="disarm-mine-stage-1-helper"/
                        <action name="set-disarmed-phase-1"/>
                        <action name="disable-agent"/>
                </then>
        </if>
</action>

<!-- Action: if on a mine, disarm the second stage -->
<action name="disarm-mine-stage-2" builder="sweep.model.action.Cor
        <if>
                <condition sensor="on-partial-mine" value="true"/>
                <then>
                        <action name="disarm-mine-stage-2-helper"/
                        <action name="set-disarmed-phase-2"/>
                        <action name="disable-agent"/>
                </then>
        </if>
</action>

<!-- Action: First stage of disarming the mine -->
<action name="disarm-mine-stage-1-helper" class="grid.model.action
        <attributes>
                <attribute name="environment" value="mine.grid"/>
                <attribute name="value" value="mine.value.partial'
        </attributes>
</action>

<!-- Action: Second stage of disarming the mine -->
<action name="disarm-mine-stage-2-helper" class="grid.model.action
        <attributes>
                <attribute name="environment" value="mine.grid"/>
                <attribute name="value" value="mine.value.disarmed
        </attributes>
</action>

<!-- Action: set the phase 1 disarming flag -->
<action name="set-disarmed-phase-1" class="sweep.model.action.Incr
        <attributes>
                <attribute name="name" value="disarmed-phase-1"/>
        </attributes>
</action>
<!--
<action name="set-disarmed-phase-1" class="sweep.model.action.SetS
        <attributes>
                <attribute name="flag" value="disarmed-phase-1"/>
                <attribute name="value" value="true"/>
        </attributes>
</action>  -->

<!-- Action: set the phase 2 disarming flag -->
<action name="set-disarmed-phase-2" class="sweep.model.action.Incr
        <attributes>
                <attribute name="name" value="disarmed-phase-2"/>
        </attributes>
</action>
```

```
                        <!--
                        <action name="set-disarmed-phase-2" class="sweep.model.action.SetS
                                <attributes>
                                        <attribute name="flag" value="disarmed-phase-2"/>
                                        <attribute name="value" value="true"/>
                                </attributes>
                        </action>-->

                        <!-- Action: disable the agent for the rest of the sim -->
                        <action name="disable-agent" class="sweep.model.action.DisableActi

                        <!-- Action: move in a random direction -->
                        <action name="move-random" class="grid.model.action.MoveRandomActi
                                <attributes>
                                        <attribute name="environment" value="environment"/
                                </attributes>
                        </action>

                        <!-- Action: move towards the dumpster -->
                        <action name="move-towards-dumpster" class="grid.model.action.Move
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="location" value="const:10,10"/>
                                </attributes>
                        </action>

                        <!-- Action: move towards the closest piece of garbage -->
                        <action name="move-towards-garbage" class="grid.model.action.MoveT
                                <attributes>
                                        <attribute name="environment" value="garbage.grid"
                                        <attribute name="location" value="av:garbage.locat
                                </attributes>
                        </action>

                        <!-- Action: move towards the closest mine -->
                        <action name="move-towards-mine" class="grid.model.action.MoveTowa
                                <attributes>
                                        <attribute name="environment" value="mine.grid"/>
                                        <attribute name="location" value="av:mine.location
                                </attributes>
                        </action>
                </actions>

                <initializers>
                        <initializer name="random-placement" class="grid.model.RandomPlace
                </initializers>
        </model>
</models>

<controllers>
        <controller name="basic-state-machine" format="state">
                <builder class="sweep.controller.FSABuilder"/>
                <initialState name="searching"/>
                <states>

                        <!-- State: controls the search beahviors -->
                        <state name="searching">
                                <!-- Rule: if in range of garbage, go into collecting mode
                                <transition nextState="collect-garbage">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="near-garbage" value="true"/>
                                        </rule>
                                        <action name="move-towards-garbage"/>
                                </transition>

                                <!-- Rule: if in range of a mine, go into disarming mode -
                                <transition nextState="disarm-mine">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="near-mine" value="true"/>
                                        </rule>
                                        <action name="move-towards-mine"/>
                                </transition>

                                <!-- Rule: otherwise, keep searching randomly -->
                                <default nextState="searching">
                                        <action name="move-random"/>
                                </default>
                        </state>

                        <!-- State: controls the collection of garbage -->
                        <state name="collect-garbage">
                                <!-- Rule: if on a piece of garbage, pick it up -->
                                <transition nextState="collect-garbage">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="holding-garbage" value="fals
                                                <sensor name="on-garbage" value="true"/>
                                        </rule>
                                        <action name="pick-up-garbage"/>
                                </transition>

                                <!-- Rule: if near garbage, move towards it -->
                                <transition nextState="collect-garbage">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="holding-garbage" value="fals
                                                <sensor name="near-garbage" value="true"/>
                                        </rule>
                                        <action name="move-towards-garbage"/>
                                </transition>

                                <!-- Rule: if holding garbage, move towards the dumpster -
                                <transition nextState="collect-garbage">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="holding-garbage" value="true
                                                <sensor name="on-dumpster" value="false"/>
                                        </rule>
                                        <action name="move-towards-dumpster"/>
                                </transition>

                                <!-- Rule: if holding garbage and on the dumpster, dispose
                                <transition nextState="searching">
                                        <rule logical="and">
```

```
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="holding-garbage" value="true
                                                <sensor name="on-dumpster" value="true"/>
                                        </rule>
                                        <action name="put-down-garbage"/>
                                </transition>

                                <!-- Rule: go back to searching -->
                                <default nextState="searching">
                                        <action name="move-random"/>
                                </default>
                        </state>

                        <!-- State: controls the disarming of mines -->
                        <state name="disarm-mine">
                                <!-- Rule: if on an armed mine, perform the first disarmir
                                <transition nextState="disarm-mine">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="on-armed-mine" value="true"/
                                        </rule>
                                        <action name="disarm-mine-stage-1"/>
                                </transition>

                                <!-- Rule: if on a partially disarmed mine, finish the job
                                <transition nextState="searching">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="on-partially-disarmed-mine"
                                        </rule>
                                        <action name="disarm-mine-stage-2"/>
                                </transition>

                                <!-- Rule: if near a mine, move towards it -->
                                <transition nextState="disarm-mine">
                                        <rule logical="and">
                                                <builder class="sweep.controller.SensorRul
                                                <sensor name="near-mine" value="true"/>
                                        </rule>
                                        <action name="move-towards-mine"/>
                                </transition>

                                <!-- Rule: go back to searching -->
                                <default nextState="searching">
                                        <action name="move-random"/>
                                </default>
                        </state>
                </states>
        </controller>
</controllers>

<!-- Not currently being used -->
<agents>
        <agent name="agent">
                <builder class="standard.agent.AgentBuilder"/>
                <attributes></attributes>
                <memory></memory>
        </agent>
```

```
</agents>

<swarms>
        <swarm name="swarm-A">
                <builder class="standard.swarm.SwarmBuilder"/>
                <entity name="swarm-A-entity">
                        <agent type="agent"/>
                        <controller type="basic-state-machine"/>
                        <model type="avatar"/>
                </entity>
        </swarm>

        <swarm name="swarm-B">
                <builder class="standard.swarm.SwarmBuilder"/>
                <entity name="swarm-B-entity">
                        <agent type="agent"/>
                        <controller type="basic-state-machine"/>
                        <model type="avatar"/>
                </entity>
        </swarm>
</swarms>

<!-- Not currently being used -->
<information>
        <info name="info-type1">
                <builder class="sweep.info.NullInfoBuilder"/>
        </info>
</information>

<probes>
        <probe name="terminator">
                <builder class="sweep.probe.CounterProbeBuilder"/>
                <iterations num="200"/>
        </probe>

        <probe name="summarizer">
                <builder class="sweep.probe.SummaryProbeBuilder"/>
                <sum swarm="my-swarm-A" var="garbage-counter"/>
                <sum swarm="my-swarm-A" var="disarmed-phase-1"/>
                <sum swarm="my-swarm-A" var="disarmed-phase-2"/>
                <sum swarm="my-swarm-B" var="garbage-counter"/>
                <sum swarm="my-swarm-B" var="disarmed-phase-1"/>
                <sum swarm="my-swarm-B" var="disarmed-phase-2"/>
                <screen/>
        </probe>
</probes>
</simulation>
```

## Section 2. End Notes

### Acknowledgements

- Dan Palmer
- Chad Hantak
- John Carroll University
- Orbital Research, Inc.
- Wright Patterson AFB

### Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT style sheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial Building tutorials with the Toot-O-Matic demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11. We'd love to know what you think about the tool.