# Changes to WUBWorld

The following features are new to this version of WUBWorld:

1) Wormholes can now transport agents from one part of the world to another. A wormhole is made up of two "Portals". You can detect these portals using the method **sense-portals**.

2) WUBs now release deadly shockwaves. Before the shockwaves are sent out, the WUB will turn black and begin to vibrate. If you see a WUB starting to shake, you'd better kill it or get out of the way!

3) Agents can now sense one another and communicate via method calls.

4) Agents can now undergo evolution using genetic mappings and genetic operators that you define in your code.

Information relevant to these new features is listed below.

### The following new methods are part of the Agent class:

• `sense-agents`: returns a list of nearby agents

• `sense-portals`: returns a list of nearby portals

### The following new method is part of the WUB class.

• `is-shaking`: tells whether the WUB is shaking or not. If a WUB is shaking, it's getting ready to release a shockwave.

Example:

```
if (nearestWUB is-shaking): {
    self repent.
}
```

### Communication with nearby agents:

Now that an individual can find nearby agents using the method **sense-agents,** communicating with nearby agents is as simple as sending that agent a method call. If your agents want to receive messages from other agents, they should implement a method which can accept the message and related information.

For example, if your agents decide to share information about where to find food, the following method could be used to pass the information from one agent to the other.

```
+ to hear-about-food at-location l (vector):
    self set-goal to l.
```

### Agent Evolution:

In this version of WUBWorld, each agent is given a genome: a list of real numbers which can be used to specify various aspects of the agent's behavior. When an agent dies, it will not disappear from the world. Instead, it will be reborn with a new genome. In order to implement evolution, you'll need to specify how the genome effects behaviors (the genetic mapping) and how the genome changes during evolution (the genetic operators). With a successful genetic mapping and genetic operators, the agents should evolve over time to become more effective at surviving in their complex environment.

To specify the genetic mapping, you'll need to implement a method called "map-genome" in your agent subclass. This method uses the "get-gene" method to access the genome and uses gene values to change how the agent behaves. A straightforward genome mapping might use the different gene values to set weights and thresholds for perceptrons.

```
+ to map-genome:
    # Example:
    perceptron set-weight number 0 to (self get-gene number 0).
```

To specify the genetic operator, you'll need to implement a method called "mutate" in your controller subclass. This method should introduce variation into the genomes over the course of evolution. The mutation operator should perturb the existing genome enough to introduce variation into the population, but not so much that children do not resemble their parents. Though mutation is the most straightforward operator, the method could actually perform other types of genetic operators as well, such as crossover.

```
+ to mutate with-genome g (list):
    # Example:
    g{0} += random[.01] - .005.
```