# The StarWumpus user guide

## Overview

StarWumpus is a series of extremely dark caves where there are bottom pits and Imperial Stormtroopers randomly spaced. The hero of the adventure must move from cave to cave, sensing what is in adjoining caves using his Jedi powers to avoid destruction and proceed to the exit to complete the task.

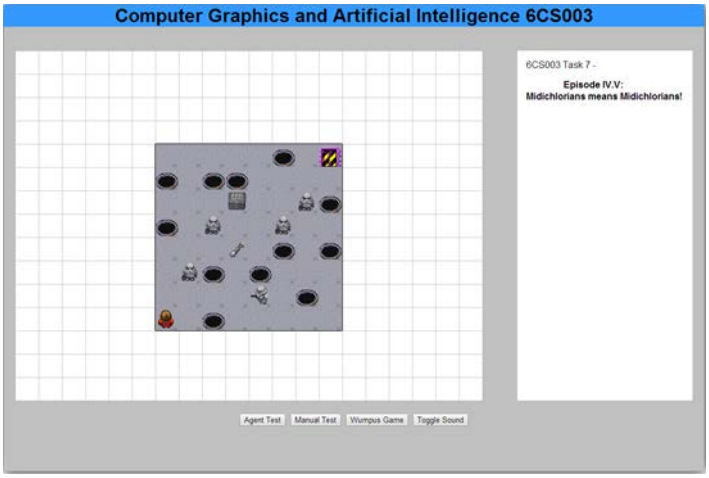


Figure 1 – The StarWumpus environment

## Controls

The simulation is presented in three panels. In the centre is the main simulation. to the right is the perception panel that displays information about the game. At the bottom is a panel of buttons (shown below) to select the various modes for the game or simulation.

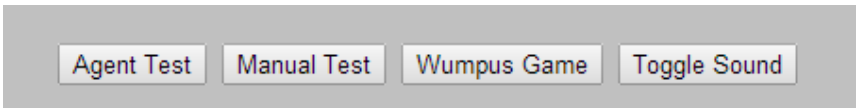


Figure 2 – The mode buttons

## Key Controls

The hero (who for no reason will be called Luke from now on) can be made to explore the caves either manually or under the control of a software agent. In manual mode Luke's movements are controlled from the keyboard using the WASD keys. The controls are:

- W - move forward
- A - turn left
- S - move backwards

- D - turn right

Luke may also manipulate (pick up or use) items found in his current location towards completing the quest, using the following key:

- U - use/pick-up

Luke may also make use of his heightened Jedi senses to determine if there is danger in the cave immediately ahead:

- L – use Jedi perception

## Perception

When Luke moves to a new cave he is able to sense some information that is displayed in the right hand message panel. However his senses are easily confused (because it is very dark and rather cavey) so what he senses could be things that are immediately ahead or to the left or right of him. Reading from left to right the perceptions are:

| Name | Values | Notes |
|---|---|---|
| stench | true or false | a Stormtrooper is near |
| breeze | true or false | a cave is near |
| bump | true or false | Luke has walked into a wall |
| glitter | dark/control-panel/lightsabre/door | The glow of a nearby item |
| scream | silence/danger/disturbance | Hearing or Jedi sense |
| feel | nothing/control-panel/lightsabre | the item is in Luke's current location |

Table 1 – The NPC perception table

And they are displayed to the top left panel on the browser as follows

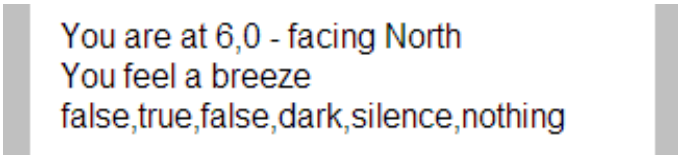


Figure 3 – The perception panel

Luke may also use his Jedi perceptions to sense if there is danger immediately ahead and if there is the scream field is updated with "danger" in the case of a cave and "disturbance" if it is a Stormtrooper.

Figure 4 – The Jedi perception

## Controlling the NPC with an AI

The agent's reasoning must be implemented within the Agent object which has two components; the constructor and the process loop. The constructor can be used to initialise your agent's knowledge base (if your design requires this feature). The agent's process loop is called once on each process cycle for the game and returns an action for the NPC to perform. The possible actions are shown below.

| Action | Notes |
|---|---|
| forward | move forward |
| left | turns left |
| right | turns right |
| backward | move backwards |
| use | use or pick-up an object |
| sense | use Jedi perception |

The sensor information from the NPC is returned in a list of values called percepts (short for perception) which are passed into the Agent object. The fields of the list can be accessed using enumerated using the name values shown in the perception table above and illustrated in the program code below.

## Direction of travel

You may also access information about the direction that the NPC is facing using the facing attribute. To access this value within the AI use the facing attribute (i.e this.hero.facing) an example is shown in code listing 1. Facing returns an integer between 0-3 to indicate the direction, the values translate as:

| Value | Direction |
|---|---|
| 0 | north |
| 1 | east |
| 2 | south |
| 3 | west |

## Where to start your intelligent agent design

The best strategy for the design of your agent, irrespective of the reasoning mechanism you use, is to have your agent walk forward as a default action. Walking forward might not always be the best thing for the agent to do so before the forward action can be taken other decisions should be tested and alternative actions, such as turning left or right or using an object, taken instead.

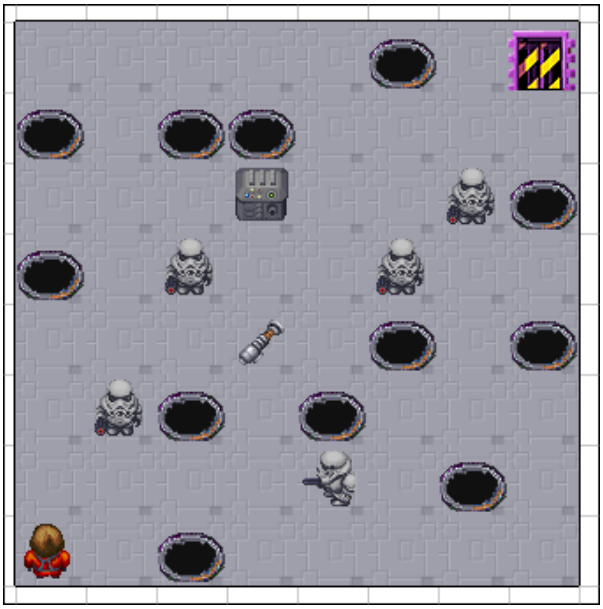

Figure 5 – The simulation environment

The code for a simple agent to control the Luke NPC is given below. Note: it contains no intelligence and simply selects actions at random until the NPC is killed.

```
// The constructor
function Agent(hero) {
    "use strict";
    // the line below must NOT be changed, it
    // allows information from the game NPC
    this.hero = hero;   // pointer to the NPC
}

// Method called by the game to initialise the agent
function Agent_init() {
    "use strict";
    // ==== Data for your agent can be initialised here ====
    // The table below is just for this demonstration, where actions
    // are chosen at random and the is a bias towards walking forward
    this.actionTable = ["forward", "forward", "forward", "forward", "right", "left", "use", "fire", "back"];
}

// Method called by the game to run the agent, it returns an action to control the NPC
function Agent_process() {
    "use strict";
    var percept = this.hero.percept;  // sensors from the NPC
    var stench = percept[PercEnum.stench];
    var breeze = percept[PercEnum.breeze], glitter = percept[PercEnum.glitter];
```

```
    var bump = percept[PercEnum.bump], scream = percept[PercEnum.scream];
    var facing = this.hero.facing;      // The direction the NPC is facing
    var randomChoice = Math.floor(Math.random() * this.actionTable.length); // pick any action
    if (stench || breeze || glitter || bump || scream){
        // do AI reasoning here
    }
    // ...and return the action to be performed
    return this.actionTable[randomChoice];
}
```

Listing 1 – A random StarWumpus agent

## Using the Behaviour tree version

A more elaborate agent architecture that makes use of a behaviour tree is given in the file jsToBeUdatedBT.js and may be used to control the Luke NPC. There are some basic patterns of behaviour contained, to illustrate how behaviour trees may be used, but it is very limited. You should not need to change any code in the Agent_process method, doing so without care is likely to stop the agent working, the agent behaviours should be specified in/via the Agent_init method. To use the behaviour tree version rename the original jsToBeUdated.js to something like jsToBeUdated-old.js (in the event that you want to recover it later) then rename jsToBeUdatedBT.js to jsToBeUdated.js. StarWumpus will run the agent code in any file called jsToBeUdated.js.