Appendix: Vacuum Cleaner World

# Introduction

To properly understand the engine it is often needed to have good examples of how to use the engine, as such this part will cover how to make the Vacuum Cleaner World that is a basic Agent based world.

Basically the idea is that the world consists for two tiles, which for the purpose of this example we will call them Tile 0 and Tile 1. The agent is a vacuum cleaner and has the ability to move from one tile to another, furthermore the agent can sense if there is dirt on its current tile. The agent also has the ability to suck if it sucks on a tile with dirt then the dirt is removed. The job of the agent is thus to remove all dirt from both of the tiles, in a more advanced version the tiles gets dirt after an uncertain amount of time, however in this example the dirt is only there from the beginning of the world creation.

The full code for this example can also be found in the VacuumCleanerWorldExample project as part of our source code.

# World

We start by creating the world for the Vacuum Cleaner world, however before we do this we must first define what a position in the world is and what information is need to add an agent.

Position in the vacuum world:

**[VacuumPosition.cs]**

Information need to add an agent to the vacuum world:

**[VacuumSpawnLocation.cs]**

Now we construct the world:

**[VacuumWorld.cs]**

Finally we create the factory that is meant to build the world on command from the engine

**[VacuumWorldBuilder.cs]**

# The entities and agents

The job of the agent in vacuum world is suck up dirt as such we must add a dirt entity:

**[DirtEntity.cs]**

Before we can add the Vacuum cleaner we must first create the modules it uses, in our case it only needs a module to see the dirt on its current location.

However before that we need to make a percept for the agent, this percept we will call VacuumVision and it will show the vacuum’s current position and if the tile it is on is dirty.

**[VacuumVision.cs]**

Now we can add the module

**[VacuumVisionModule.cs]**

And at last we can finally add the Vacuum Agent:

**[VacuumCleanerAgent.cs]**

# Actions and events

The vacuum cleaner can perform two actions which are to suck and to move. We would also like that an event gets raised when either action is done.

Thus we must first define both events

**[VacuumMovedEvent.cs]**

**[VacuumSuckedEvent.cs]**

Now that the events has been defined it is time to make the actions,

We start by creating the move action, to prevent the agent from being too fast we slow it down by putting a delay on its move action.

**[MoveVacuumCleanerAction.cs]**

Next we create the suck action

**[SuckAction.cs]**

# Controller

In order for the agent to be able to react to its environment we must provide it with an controllers, in this example we will use C# as the agent language.

First we start by making the controller, the controller contains logic on how the agent should behave, in this example we make the agent suck if there is dirt otherwise it will simply move and look for dirt elsewhere.

**[VacuumAgentController.cs]**

Second we make an AgentManager class this class’s object will be responsible for constructing a controller for the agent as such it must be provided with the name of the agent so it can be located in the world:

**[VacuumAgentManager.cs]**

# View

All other parts are now done and it is time to make a visualization of the world and the agent, to keep the example as simple as possible we will use an extension meant for making loggers and use a log as our view. This way we can track all actions taken by the vacuum agent.

**[VacuumWorldView.cs]**

# Design map and wire the parts together

Before the engine can be started its needed to design a map for the engine, there are multiple options how the world looks we have chosen the one where the agent is on Tile 0 and there is only dirt on Tile 1.

**[VacuumMap1.cs]**

When the map has been designed the only part left of the engine is to wire all its components together, this is done in the C# programs main method:

**Notice!** The log is written to file called viewlog.log

**[Program.cs]**

# Testing the vacuum world

Opening the **viewlog.log** we can see how the agent has progressed:

*Vacuum moved from: Tile(0), to: Tile(1)*

*Vacuum sucked*

*Vacuum moved from: Tile(1), to: Tile(0)*

*Vacuum moved from: Tile(0), to: Tile(1)*

We can see that the agent moved to Tile 1 and sucked up the dirt, after which it went back and forth to locate more dirt.