# System features

## Overview of engine

The goal of the engine is to allow for simulation of a world where agents within are allowed to act, as such what is important for this is that it can accurately model a state-machine. To model a state-machine one must have the ability to save a state and perform actions to change the current state.

### State

In our domain model we have state stored in three object types:

* World
* Entities
* Modules

##### World

The world is the place where all entities are meant to inhabit as either agents of the world or simply objects for the entities to interact with. The world is not defined by us as shown on the domain uml it is an abstract class meaning it is the developer using our engine that defines the world. As such the world can be any type of world needed, it could be a 3-d world, a 2-d world, a world based on tiles or hexagons or simply be nodes with an undefined number of edges connecting each other.

##### Entities

The world is empty without anything inside it as such we have the entities which are meant to model the objects one would have the world to contain. For example in our reference implementation using our engine, we have a maze with packages scattered about. It is then the task of the agents to collect these packages; the entities here are not only the walls of the maze and the packages but also the agents since they inhabit the world as well. The agents are different from entities in the way that they all have a name this name is unique and is meant to be a way of distinguishing the agents from one another.

##### Modules

The modules can be viewed as the constraints and as the abilities of all entities. For instance if you wanted to constrain entities from moving into each other than you would create a Movement blocking module, this module would then contain information on whether or not a given entities is allowed to pass it.

### Actions

A world is static and unexciting if one is not allowed to perform any changes to it, for this we have what we have chosen to name actions. There are two different kinds of action types there are environment actions and there are entity actions the core difference between them is that entity actions are meant as actions a single entity performs such as moving the entity or having the entity pick up another object. Environment actions are actions that affect the entire world. In our domain model we have chosen to add two actions that are built into the engine, the first is an entity action that gets all the percepts for a given entity called “GetallPerceptsAction” and the other is an environment action that can shut down the engine called “CloseEngineAction”.

### Events and triggers

The engine relies heavily upon events, this means that all actions performed within the engine is meant to trigger events in responds. This can be used to either activate new actions within the engine, or be meant to transfer data to the views listening.

In order to listen to the events, a trigger need to be created with all the events it listens to registered to it. Furthermore a trigger needs a condition and an action, the condition is a predicate that determines whether the trigger is fired, the action is the function that is fired.