Use Cases für die Realisierung der Roboter im Rocket Unity Game

[1 Modellierung von Akteuren und deren Verhalten 2](#_Toc402365227)

[1.1 Beschrieb der Situation: 2](#_Toc402365228)

[1.1.1 Initialisierung 2](#_Toc402365229)

[1.1.2 Spielerinteraktion 3](#_Toc402365230)

[1.1.3 Recharching State 3](#_Toc402365231)

[1.1.4 Patrolling 3](#_Toc402365232)

[1.1.5 Betätigung Schalter im Detail 4](#_Toc402365233)

# Modellierung von Akteuren und deren Verhalten

**Ausgangslage: Funktionen und Attribute**

|  |  |
| --- | --- |
| Folgende Attribute besitzt ein Akteur:  Diese Funktionen und Attribute erbt Roboter von der „Actor“-Klasse.  **Legende:**  Attribute  Funktionen |  |

|  |  |
| --- | --- |
| Die Roboterklass sieht folgende Attribute und Funktionen vor:  Auszüge von GenMyModel:  **ClassDiagram - baseDesignRobot** |  |

|  |  |
| --- | --- |
| **Die eigentliche Roboterinstanz „Robot1“ kennt 2 States: Recharching und Patrolling.**  **Default ist dabei Recharging** |  |

## Beschrieb der Situation:

### Initialisierung

In einem separaten Raum C ist wird beim Spielstart ein Roboter „Arnold“ initialisiert. Als zuerst wird von Unity die Initialen Werte initIsActiv = false und initBatteryLevel = 50 werden in isActive und batteryLevel abgefüllt. . Weiter wird der Roboter in den **State** „**Recharging**“ gestellt beim initialisieren. Da der Roboter beim Initialisieren inaktiv ist, wird also eine Spielerinteraktion benötigt, um den Roboter zuerst in Fahrt zu bringen. Wie wir später sehen werden, muss sich der Roboter immer wieder mit Strom aufladen, deshalb wird auch eine „PowerSupply“ im Level Initialisiert mit hasPower = true als Initialwert.

|  |  |
| --- | --- |
| **PowerSupply:** |  |

### Spielerinteraktion

An der Wand im Raum C ist ein Schalter, der die Funktion toggleActive() vom Roboter „Arnold“ aufruft. Damit hat der Spieler die Möglichkeit, den Roboter ein-, und auszuschalten.

Er ist im Moment noch inaktiv. Also schaltet der Spieler mit dem Schalter den Roboter ein.

Sobald dieser beim nächsten Update() beim Unity Spielzyklus das „isActive = true“ und batteryLevel > 0 festgestellt wird, wird die execute() Funktion von Recharching dieses Roboters ausgeführt.

Der Roboter funktioniert nur, wenn er Strom hat bzw. er noch batteryLevel > 0 ist und eingeschalten ist, also „isActive = true“.

### Recharching State

Innerhalb vom Recharging State wird zuerst das batteryLevel geprüft. Wenn batteryLevel > criticalBatteryLevel zutrifft, wird mit getNextPowerSupply() die Position der nächsten PowerSupply gesucht. Wird keine Position gefunden, wartet der Roboter ab und versucht es im nächsten Update() erneut.

Wird eine Position gefunden, nähert der Roboter sich der Station mit der Funktion goToPositioin() aus der „Actor“ Klasse. Sobald der Roboter mit seinem Interaktionsradius (InteractionRange) den Interaktionsradius der Station erreicht, lädt er sich auf. Das heisst batteryLevel erhöht sich langsam. Diese Aufladegeschwindigkeit wird von rechargeRate der PowerSupply und der maxRecharchingSpeed vom Roboter ermittelt.

Sobald maximumBatteryLevel erreicht wird, wird von Recharching State zu Patrolling State gewechselt.

### Patrolling

Sobald der Roboter sich in der Patrolling State befindet, wird immer zuerst powerDrain vom batteryLevel abgezogen weil der Roboter jetzt von der Batterie betrieben wird.

Mit dem getNextWaypoint() wird zuerst die nächste Position gesucht, zu der sich der Roboter bewegen soll.

Sobald dieser Punkt erreicht wird, wird die nächste Position im wayPointList-Array abgefragt und als neuer Zielpunkt verwendet. Ist der letzte Punkt von wayPointList-Array erreicht, wird wieder mit dem ersten Eintrag begonnen.

Zu Beginn wird immer batteryLevel <= criticalBatteryLevel geprüft. Trifft die Bedingung zu, wird der State zu Recharching gewechselt.

### Betätigung Schalter im Detail

Schalter wird durch den Spieler aktiviert. Dies aktiviert die PowerSupply und den Roboter im ganzen Level.MSG: POWER\_ON und isActive wird an alle Roboter und PowerSupply-Stationen verschickt.   
Empfänger sind in diesem Szenario alle Actors des Levels.

#### Versand MSG POWER\_ON und isActive im Detail

SupplyStation und Robot erhalten die MSG POWER\_ON und isActive - Nachricht und gibt sie weiter an seine State Machine.  
Die State Machine von Robot verwendet ihre handleMessage Methode um vom currentState und

vom RobotEnvironment eine Rückmeldung zu erhalten.

RobotEnvironment gibt true zurück, da es sein onPower Attribut auf true gesetzt hat.  
CurrentState ( Recharching ) gibt false zurück, da es keinen momentanen Statuswechsel gibt.

Nachdem die Message verarbeitet wurde, führt die State Machine von Robot1 ihre update() Methode aus, was momentan ‚nur‘ zur Ausführung von execute() vom State führt.

Unabhängig von der update() Methode der SM von Robot1 führt das Objekt *RobotEnvironment*

mit jedem Frame seinen Code ([Pseudocode](#_Pseudocode_RoomBEnvironment)) aus.