Use Cases für die Realisierung der Räume in Rocket

[1 Modellierung von Räumen und deren Verhalten 2](#_Toc402977890)

[1.1 Modellierung durch eine State Machine 2](#_Toc402977891)

[1.2 Beschrieb der Situation: 2](#_Toc402977892)

[1.3 Elemente der Sequenz 3](#_Toc402977893)

[1.4 Betreten von Raum A 4](#_Toc402977894)

[1.5 Licht einschalten 4](#_Toc402977895)

[1.5.1 Allgemein: Beleuchtung in den Räumen 4](#_Toc402977896)

[1.6 Spieler geht zum Kontrollpult 5](#_Toc402977897)

[1.6.1 Betätigung Schalter2 5](#_Toc402977898)

[1.6.2 Alternative: Schalter 3 ist per default OFF 6](#_Toc402977899)

[2 Anhang 6](#_Toc402977900)

[2.1 Pseudocode RoomBEnvironment 6](#_Toc402977901)

# Modellierung von Räumen und deren Verhalten

## Modellierung durch eine State Machine

Passives Element

Gründe für die Verwendung einer State Machine  
Gründe die gegen eine Verwendung einer State Machine sprechen

## Beschrieb der Situation:

Das Spiel wurde neu gestartet :

Die Hauptstromversorgung ist unterbrochen, da der Hauptschalter in Raum A nicht auf ON ist.

Dies bedeutet, dass zu Beginn des Spiels das Attribut powerState der einzelnen Environment Objekte, mit Ausnahme jenes in Raum A auf false geschaltet sind.

Es funktionieren somit nur jene elektronische Anlagen, die an der Notstromversorgung angeschlossen sind, was im Moment nur für Raum A zutrifft. Weitere Räume, die an der Notversorgung angeschlossen sind, sind nicht geplant. Falls diese doch realisiert werden sollten, käme am einfachsten ein weiteres Attribut: emergencyPower zur Environment Klasse hinzu.

**Die Räumlichkeiten und ihre Status**

|  |  |  |
| --- | --- | --- |
|  | **Room A** | **Room B** |
| Beschrieb | Kontrollraum | Beliebiger Raum |
| Stromversorgung | Notstrom, deshalb  powerState = true | Primärstrom, deshalb  powerState = false |
| Architektur | Haupteingang (elektrisch) | Haupteingang (elektrisch) |
| **Items** | Schalter1: Licht  **Schalter2: Primärstrom Level**  **Schalter3: Stromversorgung Raum B**  Lampe  Türe (nicht in Modell implementiert) | Schalter1: Licht  Lampe  Türe (nicht in Modell implementiert) |

## Elemente der Sequenz

Alle zum Startzeitpunkt aktiven Zustände und die Attribute der Räume A und B sind ebenfalls im GenMyModel aufgezeichnet.

|  |  |  |
| --- | --- | --- |
|  | **Room A** | **Room B** |
| **Zustand** | [active, empty] | [active, empty] |
| **Attribute** |  | |
| isVisited | false | false |
| accessible | true | false (door is locked due to power state) |

Weitere Objekte, die instanziert wurden und deren Attribute vor dem Eingreifen des Spielers:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **RoomBEnvironment** | **RoomAEnvironment** |
| onPower | false | true |
|  |  |  |
| **Methoden** |  | |
| getPowerState()  setPowerState() |  | |

### Schaltpult

|  |  |  |  |
| --- | --- | --- | --- |
|  | Schalter1 (Licht in A) | Schalter2 (Power All) | Schalter3 (Power Room B) |
| **Attribute** |  |  |  |
| switch | off | off | on / off (tbd) |
|  |  |  |  |
|  |  |  |  |
|  |  | |  |

## Betreten von Raum A

Der Spieler betritt Raum A (Kontrollraum). Eine andere Wahl hat er nicht, da Raum B auf Grund der stromlosen elektrischen Tür verschlossen ist. Raum A befindet sich zu Beginn des Spiels im Zustand [Idle]. Beim Eintreten wechselt der Status zu [active, hasPlayer].

Raum A ist dunkel, dennoch sind (an der Wand) drei Schalter zu erkennen:

1. Ein Schalter um in A die Beleuchtung anzuschalten
2. Ein Schalter um auf allen Ebenen, die Stromversorgung ein- und auszuschalten
3. Ein Schalter um in Raum B die Stromversorgung selektiv zu de-/aktivieren

Raum A ist beim Eintreten dunkel oder dank der Notstromversorgung nur schwach beleuchtet.

An der Seite befindet sich der Lichtschalter.

Durch das Betreten des Raumes, wird ein Zustandswechsel ausgeführt.

Der SphereCollider Trigger wird aktiv und Raum A gibt eine MSG: PLAYER\_ENTERS an seine StateMachine weiter, welche vom currentState ein true zurückerhält. Der currentState ( [active, empty] ) verlangt von der StateMachine einen Wechsel in den Status [active, holdsPlayer].

## Licht einschalten

Der Spieler aktiviert den Schalter 1, mit dem die Lampe (=Lamp1) eingeschaltet wird. Die Interaktion mit dem GameObjekt „Schalter1“ aktiviert die Light-Komponente der Lampe unter der Voraussetzung dass auch der powerState true ist.

(Umsetzung: Interaktionsradius um Schalter, Triggern der Aktivitätstaste und wenn diese aktiviert wird, wird das Event an das Objekt weitergegeben. Binärer Status des Schalters: 0-1)

### Allgemein: Beleuchtung in den Räumen

Item Lamp1 (Lampe in Raum A):

*update() {*

*if ((Schalter1)==true && myRoom.sm.environment.getPowerState() ) light: on;*

*}*

*else light: off;*

Nach Betätigen des Schalters befindet sich die Lampe in folgendem Zustand:

|  |  |  |
| --- | --- | --- |
| **Attribut** | **Value** | **Beschrieb** |
| myRoom | Room | Raum, in dem sich das statische Objekt befindet |
| light | true | Licht schaltet ein (light value increases from 0 to x) |
|  |  |  |

## Spieler geht zum Kontrollpult

Der Spieler hat Schalter2(Primärstrom) und Schalter3(Strom für Raum B) vor sich.  
Schalter3 ist inaktiv – das heisst, der Aktionsradius ist deaktiviert (da Level unpowered).

### Betätigung Schalter2

Schalter2 wird durch den Spieler aktiviert. Dies reaktiviert die Stromversorgung für das ganze Level.

MSG: POWER\_ON wird an alle Räume verschickt, deren zugehörige Schalter auf ON sind:  
Empfänger sind in diesem Szenario einmal alle Räumlichkeiten des Levels, weil angenommen wird, dass alle nachfolgenden Schalter (hier nur Schalter3) per default ON sind. [Alternative hier.](#_Betätigung_Schalter3)

#### Versand MSG POWER\_ON

Raum B erhält die POWER\_ON Nachricht und gibt sie weiter an seine State Machine.  
Die State Machine von Raum B verwendet ihre handleMessage Methode um vom currentState und

vom RoomEnvironment eine Rückmeldung zu erhalten.

RoomEnvironment gibt true zurück, da es sein onPower Attribut auf true gesetzt hat.  
CurrentState ( [active, empty] ) gibt false zurück, da es keinen momentanen Statuswechsel gibt.

Nachdem die Message verarbeitet wurde, führt die State Machine von Raum B ihre update() Methode aus, was momentan ‚nur‘ zur Ausführung vom State [active, empty] führt.

Unabhängig von der update() Methode der SM von Raum B führt das Objekt *RoomBEnvironment*

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

#### Auswirkungen in Raum B: Türe und Licht

*Annahme:* *Schalter3 ist per default auf true, d.h. Raum B wird beim Einschalten von Schalter 2 mit Strom versorgt.*  
Zwei Items werden von der aktivierten Stromversorgung beeinflusst:   
Die Türe und die Beleuchtung.

Item Lamp2 (Lampe in Raum B): Analog zur Lampe in Raum A, müssen auch hier die beiden Bedingungen erfüllt sein, so dass im Raum B Licht brennen kann.

*update() {*

*if ((Schalter1)==true && myRoom.sm.environment.getPowerState() ) light: on;*

*}*

Item Türe in Raum B

Bisher war die Türe in Raum B geschlossen, weil der Strom fehlte. Durch die Änderung im *RoomBEnvironment* öffnet sie sich. Die Implementation sieht gleich aus wie bei der Lampe.

Alternative:

Wenn wir die Türe in die Verteilerliste für elektrische Geräte hinzufügen, und sie auf einen Stromunterbruch / Stromwiedereinschaltvorgang reagieren lassen wollen, muss beachtet werden, dass der Strom auch einzeln, für Raum B ausgeschaltet werden kann. Deshalb ist eine Prüfung des Environments des ParentRooms zu bevorzugen.

### Alternative: Schalter 3 ist per default OFF

Es wurde davon ausgegangen, dass Schalter3 per default ON ist, und Raum B gleich nach dem Einschalten des Schalter2 wieder zu Strom kommt. Wäre dies nicht der Fall, bliebe die Türe verschlossen und die Lampe in B würde nicht zu brennen beginnen (siehe ihre update() Methoden).  
Die Betätigung von Schalter3 müsste also noch zusätzlich gemacht werden.

Durch das Aktivieren von Schalter 2 ist er für die Interaktion freigegeben.

Betätigen des Schalter3 löst eine Nachricht POWER\_ON aus, die gezielt an RaumB geschickt wird.  
Der weitere Verlauf ist analog Punkt [Versand MSG POWER\_ON](#_Versand_MSG_POWER_ON) zu verstehen.

# Anhang

## Pseudocode RoomBEnvironment

onMessage(MSG) { if MSG=POWER\_ON -> powerOn: true

if MSG=POWER\_OFF ->powerOn: false }

update() { if powerON -> electricityBill++ }