Inhalt

[2 Modellierung von Räumen und deren Verhalten 2](#_Toc402091143)

[2.1 Rahmenbedingungen der Sequenz 3](#_Toc402091144)

[2.2 Betreten von Raum A 3](#_Toc402091145)

[2.3 Licht einschalten 3](#_Toc402091146)

[2.4 Spieler geht zum Kontrollpult 4](#_Toc402091147)

[2.4.1 Betätigung Schalter2 4](#_Toc402091148)

[2.4.2 Betätigung Schalter 3 5](#_Toc402091149)

[2.4.3 Betätigung Schalter3 6](#_Toc402091150)

# Modellierung von Räumen und deren Verhalten

* States, Stateswechsel
* Abhängigkeiten
* Messaging
* Interaktivität, Einfluss durch Spieler

**Beschrieb der Situation:**

Umgebungsbedingungen

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

Es funktionieren somit nur jede elektrische Anlagen, die an der Notstromversorgung angeschlossen sind. So u.A. Raum A.

**Die Räumlichkeiten und ihre Status**

|  |  |  |
| --- | --- | --- |
|  | Room A | Room B |
| Beschrieb | Kontrollraum |  |
| Stromversorgung | Notstrom | Primärstrom |
| Architektur | Haupteingang (elektrisch) | Haupteingang (elektrisch) |
| **Items** | Schalter1: Licht  **Schalter2: Primärstrom Level**  **Schalter3: Stromversorgung Raum B**  Lampe  Türe | Schalter1: Licht  Lampe  Türe |

Der Spieler betritt Raum A (Kontrollraum). 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

Alle bis zum jetzigen Zeitpunkt möglichen Zustände und die Attribute der Räume A und B sind im GenMyModel aufgezeichnet.

## Rahmenbedingungen der Sequenz

Beim Start des Spiels befinden sich die Räume in folgenden Zuständen:

|  |  |  |
| --- | --- | --- |
|  | 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 |
|  |  |  |
| burning |  |  |
|  |  |  |
| **Methoden** |  |  |
| update() (unity) | Auführung bei jedem Frame: | |

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

## Betreten von Raum A

Der Spieler betritt Raum A. Eine andere Wahl hat er nicht, da Raum B auf Grund der stromlosen elektrischen Tür verschlossen ist.

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

An der Seite befindet sich der Lichtschalter.

## Licht einschalten

Der Spieler aktiviert den Schalter 1, mit dem die Lampe (=Lamp1) eingeschaltet wird. Die Interaktion mit dem GameObjekt „Schalter1“ setzt das Zustands-Attribut der Lampe auf true.

(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)

Item Lamp1 (Lampe in Raum A):

*update() {*

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

*}*

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 ist stromversorgt*  
  
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.

### Betätigung Schalter 3

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 nun auch 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++ }

Beispielszenario für die Modellierung von Räumen Entwurf1

17.10.2014

Handlungsstrang:

Initialsituation:

RoomA .globalState.getPowerState-> onEmergency

; //deshalb hat man Zutritt

RoomA .currentState = [Accessible, NonVisited];

RoomB .globalState.getPowerState-> powerless

; //deshalb NonAccessible

RoomB.currentState = [NonAccessible, NonVisited];

Handlung:

Der Spieler betätigt Schalter 1, welcher die Beleuchtung in A aktiviert.

Die Betätigung des Schalters wird getriggert und die Attribute des Raumes resp. der Items im Raum verändert:

RoomA.switch1 = true; RoomA.lamp = true;

Nach Betätigung des Schalters 1 wird Raum A durch die Beleuchtung erhellt.

Der Spieler betätigt den Schalter 2, welcher auf dem ganzen Level die Stromzufuhr einschaltet.

Objekt Schalter2:  
boolean switch: on

if(pushed && playerinrange)

send Msg an Verteilerliste: *DL\_powerConsumer*.

In dieser Liste befinden sich alle Komponenten / Entities, die auf eine konstante Stromversorgung angewiesen sind, also nicht mit Batterie laufen.

Diese Entities, die sich beim Instanzieren in *die DL\_powerConsumer*-Liste eingetragen haben, erhalten die Nachricht GOT\_POWER. Sie akzeptieren diese und leiten Sie an die StateMachine weiter.

Die StateMachines, erhalten diese Message und geben sie an den GlobalState weiter, der diese als Erster handelt. Bei uns managt der GlobalState Dinge wie „Stromunterbruch“, „Sprinkleranlage an“ etc. Die onMessage() Methode des GlobalState – hier am Beispiel des bisher stromlosen Raum B’s - liefert somit ein *true* zurück.

onMessage() {

if(powerON) {

switch(hasStateX) {

case0: ..

case1: ..

default: FSM.saveCurrentState(), FSM.changeState(**accessible**) }

}

}

RoomB.globalState.setPowerState() -> onPower.

Raum B war deshalb inaccessible, weil kein Strom zur Verfügung stand.

Durch das Handling der Message durch den GlobalState, ist der Raum nun zugänglich -> Siehe Code als nächstes. Falls der Stromunterbruch während dem State [Accessible, Visited] geschah und der Raum auf diese Weise auf inaccessible geschaltet wurde, wird zurück zu diesem Zustand gewechselt der als previousState bekannt ist

Der inaccessible-State nimmt die Message entgegen und führt nun (aufgerufen durch das Update der State-Machine) die execute Methode aus, in der geprüft wird  
(triggering), ob Strom vorhanden ist. Wenn ja – und wenn ev. noch andere Bedingungen erfüllt werden, wechselt er in den entsprechenden Status:  
if(parentRoom.*globalState.getPowerState()==onPower && door is unlocked) {*

SM.changeState(Accessible);

}

Falls Raum B vor dem Stromzusammenbruch einen eingeschalteten Lichtschalter besass, ist das *lampSwitch Attribut des Raumes* true & wenn wir davon ausgehen, dass die Lampe zum einen auf den Switch prüft und zum anderen entweder:

1. ebenfalls auf den GlobalState zugreift, um den Powerstand abzufragen oder
2. als Empfänger der der Message GOT\_POWER über den wieder erlangten Saft informiert wird.

Dann sollte sie wieder zu leuchten beginnen. Das Handling der Items ist noch Gegenstand der Diskussion. Grundsätzlich finden wir es sinnvoll, wenn alle elektrischen betriebenen, vom Strom abhängigen Items in der DL\_powerConsumer Liste sind. Dann könnte man auch von der Abhängigkeit des GlobalStates wegkommen.

Der Spieler betätigt Schalter 3, mit dem in Raum B, einzeln der Strom ein- und ausgeschaltet wird.

Strom wird ausgeschaltet (im Moment ON) und an Raum B wird die Message: NO\_POWER geschickt.

Die Entity RaumB gibt dies weiter an die im Raum vorhandenen Entities wie Lampe, Maschine1, etc. (oder eben alternativ sind alle Geräte Empfänger der Verteilerliste).

Die eigene State-Machine, wird vom GlobalState veranlasst, den Zustand anzupassen. Der aktuelle Status „Accessible“ wird verlassen, auf die gleiche Weise wie er vorhin eintrat und es wird in den Inaccessible Zustand gewechselt.

Beim Wechsel wird der PreviousState abgespeichert, damit man weiss, ob der Raum besucht wurde, oder nicht.