Beispielszenario für die Modellierung von Räumen

**Beschrieb der Situation:**

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

**Raum A:** **Raum B:**

**Umgebung:** **Umgebung:**

Bei Eintritt dunkel, nach Betätigung des Bei Eintritt dunkel

Schalters A beleuchtet

**Gegenstände: Gegenstände:**

- Schalter 1, Schalter 2, Schalter 3 - Tür als Ein- und Ausgang

- Lampe für die Beleuchtung

- Tür als Ein- und Ausgang

**Handlungsstrang:**

**RoomA *.globalState.getPowerState-> onEmergency***

**; //deshalb hat man Zutritt**

**RoomA *.currentState = [Accessible, NonVisited];***

**RoomB *.globalState.getPowerState-> powerless***

**; //deshalb *NonAccessible***

**RoomB*.currentState = [NonAccessible, NonVisited];***

Der Spieler betritt Raum A, welcher durch dank der Notstromversorgung zugänglich ist. Alles in Raum A wird vom Notstromgenerator (nicht modelliert) versorgt, deshalb sind auch die Schalter in Raum A verfügbar.

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

Die Betätigung des Schalters wird getriggert und die Attribute des Raumes verändert:

RoomA.*switch1 = true;*

RoomA.*lamp = true;*

**?** Für eine Aktion wie das Einschalten der Beleuchtung, ist kein Statewechsel notwendig, oder?

Dass der Raum nun erhellt ist, könnte folgende Auswirkungen haben:  
*- Roboter sieht in den Raum und sieht, wer sich darin befindet -> globalState.isEnlighted()*

*- Stromverbrauch wird erhöht -> globalState: while(lights) {electricityBill++};*

- .....

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

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

Die Betätigung des Schalters wird getriggert und beim Einschalten wird eine Telegramm an die Verteilerliste: *DL\_powerConsumer* geschickt.

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 (in buckland als zweiter, ist das ein Problem) und ein true zurückliefert.**

RoomB.*globalState.setPowerState() -> onPower.*

Raum B war deshalb inaccessible, weil kein Strom zur Verfügung stand. Der inaccessible State 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 RaumB vor dem Stromzusammenbruch einen eingeschalteten Lichtschalter besass, ist das *lampSwitch* true:

Trotzdem leuchtete sie nicht, weil sie durch ihre update() Methode konstant den PowerLevel des Raumes anfragt:

*while(lampSwitch == true && parentRoom.getGlobalState().onPower==true) {*

*lamp.light = true;*

*}*

*lamp.light = false;*

ODER: Alle Lampen erhalten die Message an *powerConsumer\_DL* auch und prüfen, ob sie eingeschaltet sind – wenn ja, beginnen Sie wieder zu leuchten.

**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. und zudem an die eigene State-Machine, welche ihren globalState anpasst. 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.