HANSER

Matthias Melzer

Second Life-Programmierung mit der Linden Scripting Language

ISBN-10: 3-446-41349-9
ISBN-13: 978-3-446-41349-8

Leseprobe

Weitere Informationen oder Bestellungen unter
http://www.hanser.de/978-3-446-41349-8
sowie im Buchhandel

4.4 Skriptfunktionen

Sie können Skripte im laufenden Betrieb austauschen. Damit nicht jeder beliebige Spieler ein Skript übergeben kann, wird das Skript, das ausgetauscht werden soll, mit einer *PIN* (Persönliche Identifikationsnummer) versehen. Dies erreichen Sie durch die Funktion:

```
llSetRemoteScriptAccessPin(integer pin)
```

Dieser Funktion übergeben Sie einen beliebigen `integer`-Wert, der als persönliche Geheimzahl angesehen wird. Beachten Sie jedoch hierbei, dass die Geheimzahl für das ganze Objekt zählt und somit bei Einrichtung eine Eigenschaft des Objektes ist und nicht nur eine Variable im setzenden Skript. Wenn Sie also mehrere Skripte in einem Objekt haben, welche die PIN ändern, ist die PIN, mit der Sie darauf zugreifen können, immer die letzte gesetzte Geheimzahl. Das Setzen einer Geheimzahl verzögert den weiteren Ablauf des Skriptes um 0,2 Sekunden.

Den Zugriff auf ein auszutauschendes Objekt erhalten Sie über die Funktion:

```
llRemoteLoadScriptPin(key target, string name, integer pin, integer run-
ning, integer start_param)
```

Ihr übergeben Sie die UUID des Objektes und den Namen des Skriptes innerhalb des Objektinventars, das ausgetauscht werden soll. Das auszutauschende Skript wird ohne weitere Warnung überschrieben. Der dritte Parameter ist die Geheimzahl. Ist sie nicht identisch mit der definierten, wird die Funktion abgebrochen, und Sie erhalten eine Fehlermeldung:

```
Task Object trying to illegally load script onto task Other_Object!
```

Über den Parameter `running` können Sie bestimmen, ob das Skript aktiviert sein soll oder nicht. Vielleicht möchten Sie das Skript erst zu einem bestimmten Zeitpunkt starten, dann wäre es sinnvoll, dies in einem anderen Skript über die Funktion

```
llSetScriptState(string name, integer run)
```

ein- bzw. auszuschalten. In beiden Fällen erreichen Sie durch eine Übergabe von `TRUE` ein Aktivieren oder durch Übergabe von `FALSE` ein Deaktivieren eines Skriptes. Der letzte Parameter der Funktion `llRemoteLoadScriptPin` ist ein `integer`-Wert, den Sie als Startwert übergeben können. **Listing 4.8**, **Listing 4.9** und **Listing 4.10** sollen Ihnen den Vorgang eines Austauschens einmal näher erläutern. Sie brauchen dazu zwei Prims, in einem platzieren Sie das Skript aus **Listing 4.8**. Es beinhaltet das Skript, das ersetzt werden soll. In dem anderen Prim hinterlegen Sie die zwei Skripte aus **Listing 4.9** und **Listing 4.10**. Berühren Sie das Prim mit den zwei Skripten, und beobachten Sie den Chat.

Listing 4.8 Skript-Update: zu ersetzendes Skript

```
integer pin = 1000;
integer CHANNEL = 1000;

default
{
```

```
    state_entry()
    {
        llSetRemoteScriptAccessPin(pin);
        llListen(CHANNEL, "", NULL_KEY, "");
        llSetTimerEvent(1.0);
    }

    timer()
    {
        llOwnerSay("LindenScript ist toll");
    }

    listen(integer channel, string name, key id, string message)
    {
        if (channel == CHANNEL && message == "getPin")
        {
            llSay(CHANNEL, (string) pin);
        }
    }
}
```

Listing 4.9 Skript-Update: das neue Skript

```
default
{
    state_entry()
    {
        llSetTimerEvent(1.0);
    }

    touch_start(integer num_detected)
    {
        llSetTimerEvent(0.0);
    }

    timer()
    {
        llOwnerSay("Update erfolgreich");
    }
}
```

Listing 4.10 Skript-Update: das Updater-Skript

```
integer CHANNEL = 1000;

default
{
    touch_start(integer num_detected)
    {
        llListen(CHANNEL, "", NULL_KEY, "");
        llSay(CHANNEL, "getPin");
    }

    listen(integer channel, string name, key id, string message)
    {
        if(channel == CHANNEL)
        {
            llRemoteLoadScriptPin( id,
                                   "script1",
                                   (integer) message,
                                    TRUE,
                                    0
                                 );
        }
    }
}
```

Wie Sie bereits richtig vermuten, können Sie nicht nur Skripte mit dem Einsatz dieser beiden Funktionen austauschen. Sie können auch Objekte um Funktionalitäten erweitern, indem Sie gänzlich neue Skripte übergeben. Jedoch müssen Sie der Besitzer des Objektes sein, das ein Update erfahren soll. Zudem müssen Sie sicherstellen, dass Sie die Rechte haben, das Objekt zu verändern. Das Objekt, das upgedatet werden soll, muss gerezzt sein, darf sich also nicht im Objektinventar befinden und muss zudem auf derselben Sim sein. Wenn eine dieser Voraussetzungen nicht erfüllt ist, schlägt Ihr Vorhaben fehl. Sollten Sie eine nicht zu identifizierende UUID übergeben haben, erhalten Sie die Fehlermeldung:

```
Unable to add item!
```

Der Einsatz von `llRemoteLoadScriptPin` verzögert den weiteren Ablauf des Skriptes um drei Sekunden.

Als letzte Neuerung wollen wir uns nun ansehen, wie man Landmarks auslesen kann.

4.4.1 Auslesen von Landmarks

Es gibt Funktionen, deren Namen viel versprechender klingen als das, was sie letzten Endes hergeben. Eine dieser Funktion ist:

```
key llRequestInventoryData(string name)
```

Eine Funktion mit solch einem Namen lädt uns ein zu fantasieren. Doch alles, was Sie momentan mit dieser Funktion anstellen können, ist das Auslesen von Landmarks. Vielleicht wird eines Tages auch noch die ein oder andere Funktionalität der Funktion zugeordnet werden.

Nachdem Sie den Namen einer Landmark übergeben haben, erhalten Sie einen Schlüssel zurück. Mit diesem ist es Ihnen möglich, im `dataserver`-Event auf die Koordinaten der Landmark zuzugreifen. Sehen Sie sich hierzu **Listing 4.11** an.

Listing 4.11 Auslesen von Landmarks

```
string LANDMARK_NAME = "Name der Landmark";
key landmarkId;

default
{
    touch_start(integer num_detected)
    {
        landmarkId = llRequestInventoryData(LANDMARK_NAME);
    }

    dataserver(key requestID, string data)
    {
        if (requestID == landmarkId) {
            landmarkId = NULL_KEY;
            vector landmarkCoords = (vector) data + llGetRegionCorner();
            vector objectPos = llGetPos() + llGetRegionCorner();
            float distance = llVecDist(objectPos, landmarkCoords);
            llOwnerSay( LANDMARK_NAME + " ist " +
                       (string) distance + " Meter entfernt."
                      );
        }
    }
}
```

In **Listing 4.11** wird die Entfernung zwischen dem Objekt und einem durch eine Landmark definierten Punkt ermittelt. Da die Rückgabe der Koordinaten als lokale Regionskoordinaten erfolgt, werden die globalen Koordinaten über die Addition mit dem Rückgabewert der Funktion `llGetRegionCorner` ermittelt. Über die `vector`-Funktion `llVecDist` wird dann letzten Endes die Distanz ermittelt.

Hiermit haben Sie einen Überblick erhalten, was alles in Bezug auf das Inventar eines Objektes möglich ist. Ungewöhnlich ist mit Sicherheit die Ausgabe der Fehlermeldungen im Chat- oder Debug-Fenster. Daher soll Ihnen Tabelle 4.2 auf der nächsten Seite noch einmal alle Fehlermeldungen im Schnellüberblick liefern. So können Sie immer einfach ermitteln, welche Funktion eine Fehlermeldung abgesetzt hat.

Im nächsten Kapitel werden wir uns die Kommunikation zwischen Objekten und ihrer Umgebung ansehen. Dies beinhaltet sowohl die Interaktion zwischen einem Spieler und einem Objekt als auch die Möglichkeiten der Kontaktaufnahme zu einem Objekt durch das Versenden einer E-Mail oder das Abrufen von Daten durch ein Objekt von einer Seite im Internet.

Tabelle 4.2 Fehlermeldungen der Objektinventar-Funktionen

Fehlermeldung	Funktion	Erklärung
Not permitted to edit this	`llAllowInventoryDrop`	Ein Spieler oder Objekt hat versucht, ein Skript zu übergeben.
No item named [Name]	`llGetInventoryCreator` `llGetInventoryPermMask`	Das Objekt existiert nicht im Objektinventar.
Couldn't find notecard [Name]	`llGetNotecardLine` `llGetNumberOfNotecard-Lines`	Die Notiz existiert nicht im Objektinventar oder ist leer.
Unable to give inventory list: No items passed filter	`llGiveInventoryList`	Objektinventar ist leer, oder Objektbesitzer hat keine Kopiererlaubnis.
Unable to add item!	`llRemoteLoadScriptPin`	Kopieren fehlgeschlagen
Task Object trying to illegally load script onto task Other_Object!	`llRemoteLoadScriptPin`	Übergabe der falschen PIN
Missing inventory item [Name]	`llRemoveInventory`	Das Objekt existiert nicht im Objektinventar.