

Marcel Kuhnt & Rüdiger Hülsmann:

OMSI - Der Omnibussimulator

Software Development Kit
v. 0.1.02

2. Mai 2010

Inhaltsverzeichnis

I	Struktur von Omsi	4
1	Allgemeine Struktur	4
1.1	Karten	4
1.2	3D-Objekte	4
1.3	Spezialobjekte	6
1.4	Splines	7
1.5	Fahrzeuge	7
2	Verzeichnisstruktur	8
II	Editor	8
3	Starten des Editors	9
4	Menü des Editors	9
4.1	Map	9
4.2	Helpers	10
4.3	Extras	10
5	Sichtsteuerung	10
6	Bau-Modi/Registerkarten	10
6.1	Tile	11
6.2	Objects	11
6.3	Splines	13
III	Objektbau	14
7	Allgemeines	15
7.1	Einleitung	15
7.2	Syntax	15
7.3	Konventionen	17
7.4	Informationen zur Dokumentation	18
8	Szenerieobjekte	19
8.1	Erstellen geeigneter x- oder o3d-Dateien	19
8.2	Einfachstes Szenerieobjekt	21

8.3	Texturen mit niedriger Auflösung erstellen	22
8.4	Nacht-/Jahreszeitentexturen, einfache Variante	22
9	Splinetypen	23
9.1	Allgemeines	23
9.2	Erstellen eines neuen Straßentyps mit dem StreetCreator	23
9.2.1	Gehweg	24
9.2.2	Fahrstreifen	25
9.2.3	Grünstreifen	26

Teil I

Struktur von Omsi

1 Allgemeine Struktur

Zum Anfang sollen einige Strukturen und Eigenheiten von Omsi erklärt werden, ohne deren Kenntnis sich u.U. der Sinn einiger Befehle oder anderer Dinge nicht erschließt.

1.1 Karten

Omsi ist fähig, eine von beliebig vielen Karten zu laden.

Jede Karte (Map) besteht aus einer oder mehreren Kacheln (Tiles), welche jeweils eine Ausdehnung von 300 mal 300 Metern haben.

Mit jeder Kachel werden die auf ihr sich befindenden Szenerieobjekte, das Terrain sowie eine Untergrundtextur geladen. Zur Zeit ist eine Variation der Untergrundtextur nur im Ganzen möglich, es gibt noch keine Möglichkeit, auf einer Kachel mehrere Texturen zu überlagern.

Damit nachts eine Straßenbeleuchtung simuliert werden kann, beinhaltet jede Kachel außerdem eine sogenannte Lightmap, welche eine einfache Textur darstellt. Diese wird nachts additiv der normalen Beleuchtung überlagert, sodass der Eindruck erweckt wird, dass die Landschaft von Lichtquellen beleuchtet wird. Dieser Effekt wirkt sich auch auf Splines und Objekte sowie Fahrzeuge und Menschen aus.

1.2 3D-Objekte

3D-Objekte sind im Allgemeinen all jene Objekte, welche aus einer Reihe von Meshs bestehen, die in einem 3D-Programm (bspw. Blender) erstellt wurden.

Damit Omsi ein solches Mesh laden kann, muss es im *.x-Format oder im Omsi-eigenen *.o3d-Format vorliegen.

Sowohl Szenerieobjekte als auch Menschen und Fahrzeuge stellen jeweils ein 3D-Objekt dar.

Ein 3D-Objekt kann sich aus mehreren x-Dateien/o3d-Dateien zusammensetzen, welche sich wiederum (intern!) aus mehreren Meshs zusammensetzen können. Hierbei gilt: mindestens ein Mesh pro Textur! D.h., wird eine x/o3d-Datei erstellt, auf die mehrere Texturen gemappt werden, so kann davon ausgegangen werden, dass mindestens ebensoviele Meshs in der x/o3d-Datei vorliegen, wie Texturen gemappt wurden. Dies ist insbesondere bei einem Materialtausch wichtig.

Jedes 3D-Objekt kann über Scripts verfügen, welche zur Laufzeit von Omsi ausgeführt werden und Einfluss auf die Simulation, die Animation, die Sounds und weitere Dinge nehmen können.

Jede x/o3d-Datei kann unabhängig animiert werden. Es ist also umgekehrt notwendig, dass jeder Teil des 3D-Objektes, welcher unabhängig animiert werden soll, in eine eigene x/o3d-Datei ausgelagert wird.

Für jedes Mesh einer jeden x/o3d-Datei kann das Material nach Wunsch angepasst werden (jenachdem auch per Script dynamisch). Hierüber werden bspw. Texturtausche für die Nacht oder eine andere Jahreszeit vorgenommen.

Jedes 3D-Objekt, egal ob Fahrzeug oder Szenerieobjekt, kann über ein oder mehrere Features folgender Auflistung verfügen:

- Stringvariablen, die im Editor gesetzt werden können (z.B. Beschriftungen)
- Text-Texturen beliebiger Schriftart, welche zur Darstellung von Stringvariablen genutzt werden können
- Sounds (beliebig viele, nach Wunsch pitch-, loop- oder triggerbar)
- Paths, also Verkehrswege (üblicherweise bei Kreuzungsobjekten)
- Ampelsysteme (für Kreuzungen), welche sowohl den Verkehrsfluss als auch die entsprechenden „sichtbaren“ Ampeln steuern
- „Passagierkabine“, also Sitz-/Stehplätze und einem System von Gängen, wo Leute langgehen können; sowohl für Busse als auch für Haltestellenhäusschen u.Ä.
- Masse, Trägheitsmoment, Schwerpunkt und Kollisionseigenschaften
- Kollisions-Mesh, sobald also ein Quader nicht mehr ausreicht, um die Kollision realistisch darzustellen

- Attachpunkte, woran man nach Bedarf im Editor weitere Objekte anheften kann
- Lichtquellen für die Lightmap der Kacheln

1.3 Spezialobjekte

3D-Objekte können Spezialobjekte sein. Spezialobjekte haben eine besondere Funktion in Omsi. Im Editor erscheint ein Hinweis darauf, dass es sich um ein Spezialobjekt handelt, außerdem wandelt sich die Schaltfläche „Beschriftung“ zur Schaltfläche „Optionen“. Zur Zeit gibt es folgende Spezialobjekte:

- Bushaltestelle
- Entrypoint, wo man einen Bus platzieren kann
- Ampel (hierbei ist das Objekt gemeint, was die Ampel-Optiken selbst enthält)
- Baum
- Parkplatz

Bushaltestelle Bushaltestellen sind normalerweise nur im Editor sichtbar. Normalerweise sollte das Standardobjekt ausreichen, weil es sich nur um einen „Helfer“ handelt. Die „optische“ Bushaltestelle ist ein anderes Objekt. Das Standardobjekt befindet sich im Ordner „Omsi/Sceneryobjects/Generic/bus_stop.sco“.

Eine Bushaltestelle beinhaltet momentan die Optionen „Name“ und „Fahrgastfaktor“. Der „Fahrgastfaktor“ ist momentan die Anzahl der Fahrgäste, die hier zusteigen; dies wird aber in Zukunft nochmal anders geregelt werden. Der „Name“ ist wichtig für die Fahrpläne und dient der Erkennung der Endhaltestelle des Busses.

Entrypoint Dieser Objekttyp, dessen Standardobjekt „Omsi/Sceneryobjects/Generic/entrypoint.sco“ ist, dient der Platzierung eines neuen Busses in der Karte. Er hat lediglich die Option „Name“, unter welchem der Entrypoint in der Auswahl erscheint.

Ampel Ein Ampelobjekt besteht üblicherweise aus einer Anzahl von Optiken und muss über ein Script verfügen. Im Editor ist es dann möglich, die Option „Nr. Traffic Light“ einzustellen. Über den allgemeinen Objekt-Parameter „Var Parent“ und über die Option „Nr. Traffic Light“ erfolgt die Zuweisung, um welche Ampel

es sich dann handelt (also bspw. Hauptrichtung, Querverkehr oder Fußgänger einer der beiden Richtungen).

Baum Der Objekttyp „Baum“ wurde eingefügt, damit zum einen das Rendern von Bäumen ressourcensparender durchgeführt wird und damit es möglich ist, die Ausdehnung des Baumes in gewissen Grenzen per Zufall zu variieren. Er verfügt über folgende Optionen: „Texture“, „Height“ und „Ratio“. „Texture“ gibt die zu verwendende Textur an, welche im zugehörigen Texturordner des Baum-Objektes liegen muss. „Height“ gibt die Höhe, „Ratio“ das Verhältnis Breite/Höhe an. Die letztgenannten beiden Parameter werden variiert, wenn ein neuer Baum platziert wird. Auf diese Weise sehen alle Bäume desselben Typs dennoch immer etwas unterschiedlich aus.

Parkplatz Parkplätze sind Platzhalter für Auto-Objekte, verfügen über keine Optionen. Wenn Omsi (nicht im Editor-Modus) gestartet wird, werden automatisch mit dem in den allgemeinen Optionen angegebenen Prozentsatz statische Autos aus der Liste „parklist_p.txt“ (im Map-Verzeichnis) platziert. Standardobjekt ist „Omsi/Sceneryobjects/Generic/car_park.sco“ für Pkw.

1.4 Splines

Splines sind spezielle Objekte auf der Karte, welche mittels eines Profils definiert und entlang eines Pfades generiert werden.

Hierbei setzt sich der Pfad aus geraden und kreisgebogenen Stücken zusammen und kann über beliebige Neigungswechsel verfügen. Bei entsprechend definiertem Profil wird automatisch die Oberfläche mit definiert, sodass eine Straße, auf der Fahrzeuge fahren sollen, einfach und schnell erzeugt werden kann.

Entlang des lokalen Koordinatensystems können dann 3D-Objekte ebenfalls auf einfache Weise platziert werden.

1.5 Fahrzeuge

Jedes Fahrzeug wird zunächst ohne Antrieb und Bremsen ausgeliefert. Das einzige, was Omsi intern simuliert, ist der Wagen auf einer bestimmten Anzahl von Achsen mit Lenkung.

Es ist dem Script-Programmierer vorbehalten, für die entsprechende Antriebskraft zu sorgen. Hierdurch ist es möglich, nahezu alle denkbaren Verhaltensweisen

zu programmieren.

Einige Interaktionen, wie z.B. dass die simulierten Fahrgäste erkennen, wohin der Bus fährt oder ob die Türen offen sind, werden über einige wenige Variablen übermittelt, welche von Omsi für jedes Fahrzeug vorgegeben werden.

2 Verzeichnisstruktur

Wird das Verzeichnis von Omsi, im weiteren Verlauf einfach [Omsi] genannt, geöffnet, so finden sich eine Reihe von Unterverzeichnissen, die an dieser Stelle zunächst kurz erläutert und im weiteren Verlauf ausführlich behandelt werden.

Folgende Unterverzeichnisse sind für den Benutzer des SDKs wichtig:

- \Humans - Hierin befinden sich alle Menschen, welche in der Karte mit dem Bus und der Umgebung interagieren.
- \maps - Das Kartenverzeichnis. Alle Karten befinden sich zusammen mit ihren Boden- und Nachttexturen hier drinnen.
- \program - In diesem Verzeichnis befinden sich u.A. die globalen Variablenlisten, d.h. die Listen der Variablen, welche für alle Objekte einer Klasse gelten, wie z.B. Fahrzeuge.
- \sceneryobjects - Wie der Name bereits andeutet, befinden sich alle Szenarioobjekte in diesem Verzeichnis.
- \splines - Hier befinden sich alle Splineobjekte.
- \Vehicles - Alle Fahrzeuge befinden sich in diesem Verzeichnis.

Es gibt eine Reihe weiterer Unterverzeichnisse. Die darin enthaltenen Dateien gelten jedoch global und sollten nicht verändert werden - es handelt sich beispielsweise um die Himmelstexturen oder die grafische Benutzeroberfläche.

Teil II

Editor

Im Allgemeinen ist mit „Editor“ der Map-Editor gemeint.

3 Starten des Editors

Um den Editor zu starten, muss eine Verknüpfung zur omsi.exe erstellt werden. Auf diese Verknüpfung ist mit rechts anzuklicken und im Kontextmenü ist „Eigenschaften“ auszuwählen. Auf der Registerkarte „Verknüpfung“ befindet sich das Eingabefeld „Ziel“, wo üblicherweise der Pfad der omsi.exe angegeben ist, bspw. „c:\Programme\Omsi\Omsi.exe“. Dahinter muss nun ein „-editor“ ergänzt werden - außerdem wird die Verknüpfung umbenannt, bspw. „Omsi-Editor“.

Über die so geschaffene Verknüpfung kann ab jetzt der Editor gestartet werden.

Im Allgemeinen wird die Map geladen, die beim vorherigen Benutzen des Editors zuletzt geladen war. Handelt es sich um den ersten Start oder kommt es beim Laden der Map zu Fehlern (bspw. falls die Map nicht vorhanden ist), dann erscheint ein Dialogfeld zum Auswählen einer Map.

4 Menü des Editors

4.1 Map

Im Map-Menü finden sich folgende Einträge:

New: (bisher nicht funktionsfähig)

Change: auswählen einer anderen Map

Save: speichern der aktuellen Map (oder auch [Strg]+[S]).

General Properties: erlaubt das Einstellen von allgemeinen Karten-Optionen.

Refresh: die Map wird neu geladen.

Create Lightmap: die Lightmaps sämtlicher Tiles werden anhand der platzierten Objekte neu erstellt. Vorher wird die Karte automatisch gespeichert; der Vorgang kann je nach Map-Größe einige Zeit beanspruchen.

4.2 Helpers

Show Bounding Boxes: die sog. Bounding-Boxes, also die normalerweise nicht sichtbaren Quader um die Objekte, die zur Berechnung von Sichtbarkeit bzw. Kollisionen benötigt werden, können hiermit sichtbar gemacht werden.

Show Spline Paths: die Paths (Fahr-/Gehspuren) werden sichtbar gemacht. Dies passiert aber auch, wenn die entsprechenden Registerkarten ausgewählt werden („Traffic Rules“ oder „Tracks & Trips“).

Show Grid: blendet ein Gitter ein (statt den sonst üblichen Terrain-Texturen), damit die Ausrichtung und die Grenzen der Tiles sichtbar werden.

4.3 Extras

Debug Window: startet ein Debug-Fenster, womit einige interessante Listen angezeigt werden können. Ist hauptsächlich im Debug-Modus von Omsi relevant.

List all Object Strings: hiermit können sämtliche im Editor eingegebenen Objekt-Strings aufgelistet werden (alphabetisch sortiert). Diese Funktion ist bspw. gut geeignet, um auf einen Blick zu prüfen, ob bestimmte Objekte vorhanden sind (bspw. Bushaltestellen) bzw. ob sie korrekt geschrieben sind.

5 Sichtsteuerung

Die Sichtsteuerung im Editor erfolgt wie in Omsi selbst:

- Mittlere Maustaste/Mausrad Drag’N’Drop: schwenken um den sog. Sichtpunkt
- Rechte Maustaste Drag’N’Drop: zoomen
- Rechte Maustaste klicken: Sichtpunkt versetzen (Kamera fährt dort hin)

6 Bau-Modi/Registerkarten

Die Umschaltung der Bau-Modi erfolgt üblicherweise dadurch, dass die entsprechende Registerkarte ausgewählt wird. Im Folgenden werden sämtliche Registerkarten des Editors inkl. ihrer Funktionen beschrieben.

6.1 Tile

Hier können einige allgemeine Einstellungen an der jeweiligen Kachel vorgenommen werden, welche sich im Fokus der Sichtsteuerung befindet (worauf zuletzt mit rechts geklickt wurde).

„Create New Tile Here“ ist nur dann aktiv, wenn der Sichtpunkt auf einer Stelle liegt, wo sich noch keine Kachel befindet. Mit dieser Schaltfläche wird eine neue Kachel dort angelegt.

„Delete this Tile“ löscht die aktuelle Kachel (*funktioniert noch nicht*)

Im Textfeld darunter kann die Textur geändert werden. *Dieses System wird voraussichtlich noch verändert.*

„Create Lightmap of this Tile“ erstellt die Lightmap anhand der darauf platzierten Lichtquellen (z.B. Straßenlaternen).

„Background Image“ dient der Einrichtung eines Untergrundbildes (bspw. Karte oder Satellitenbild) über der **gesamten Map**. Es sind folgende Angaben hierfür nötig:

- Picture width: Breite des Bildes (Ost-West, im Original) in Metern
- Picture Height: Höhe des Bildes (Nord-Süd) in Metern
- Point where tile no. 0 is: hierüber wird in % der Bildbreite/-höhe angegeben, wo der Map-Ursprung (0. Kachel) liegt. Dieser Wert kann auch außerhalb des Bildes liegen.

6.2 Objects

Das ist der wichtigste Modus des Editors: hier werden die Szenerieobjekte erstellt, verschoben, gedreht, gelöscht und konfiguriert.

Sobald die Registerkarte aufgerufen wurde, lassen sich die Objekte per Mausklick selektieren. Selektierte Objekte werden rot; solange die Maus auf ein nicht selektiertes Objekt zeigt (hovern) verfärbt es sich blau. Dabei gilt: ein selektiertes Objekt lässt sich nicht gleichzeitig hovern.

Sobald ein Objekt selektiert wurde, sind dessen Eigenschaften auf der Registerkarte sichtbar und lassen sich dort editieren. Sobald eine der Eigenschaften verändert wurde, verfärbt sie deren Schrift ins Blaue. Die Veränderung muss dann mit [Enter] bestätigt oder mit [Esc] abgebrochen werden.

Wenn ein Objekt selektiert ist, kann dieses X-Y-verschoben werden, indem kurz [G] gedrückt wird. Sobald die richtige Stelle erreicht wurde, kann die Verschiebung mit [Enter] oder mit der linken Maustaste bestätigt oder wahlweise auch mit [Esc] abgebrochen werden. Auf die gleiche Weise kann das Objekt mit [Z] entlang der Z-Achse verschoben oder mit [R] rotiert werden.

Wenn das Objekt über Pfade verfügt (bspw. eine Kreuzung), dann rastet das Objekt automatisch an bestehende Pfad-Enden von Objekten oder Splines ein. Mit der Taste [F9] kann dann der Pfad des selektierten Objektes durchgeschaltet werden.

Gelöscht werden kann ein selektiertes Objekt mit der Taste [Entf].

Die Taste [N] dient dem Erstellen eines neuen Objektes. Wenn dabei bereits ein Objekt selektiert ist, dann wird das neue Objekt mit denselben Eigenschaften versehen wie das bis dahin selektierte Objekt (Ausnahme: die Dimensionen von Bäumen werden dennoch variiert). Ist zu diesem Zeitpunkt kein Objekt selektiert, dann wird ein komplett neues Objekt platziert. Der Objekttyp ist dann jeweils der aktuell in der Treeview auf der Registerkarte selektierte Typ.

In dieser Treeview sind alle bereits auf der Karte verwendeten Objekte in ihren jeweiligen Gruppen aufgeführt. Man findet dort also nur Objekttypen, die bereits in Verwendung sind. Um nun jedoch ein neu erstelltes Objekt einzufügen, ist der Button „Load...“ zu benutzen. Das neue Objekt wird dann automatisch einsortiert und kann dann verwendet werden. Wird es jedoch nicht verwendet, so wird die Liste beim nächsten Starten des Editors automatisch bereinigt; alle nicht-platzierten Objekte in der Treeview werden also dann entfernt.

Über der Treeview befindet sich eine Liste sämtlicher Objekte der aktuellen Kachel. Ist also ein Objekt „optisch verloren gegangen“, lässt es sich eventuell hierüber wieder selektieren und löschen.

Unterhalb der Treeview finden sich zahlreiche Optionen:

- X,Y,Z,rot: Platzierung und Rotation des Objektes. Insbesondere X,Y und

Z werden eher selten zur Anwendung kommen - dennoch ist ein genaues Platzieren möglich!

- Attach to Object...: hierüber kann das Objekt an ein anderes Objekt angeheftet werden, sofern das „Eltern-Objekt“ zuerst erstellt wurde und über Attach-Points verfügt. Der Vorgang ist folgender: zuerst wird auf die Schaltfläche geklickt, dann auf das „Eltern-Objekt“. Wird kein gültiges Objekt selektiert, bleibt das „Kind-Objekt“ „frei“.
- AttPntNr.: die Nummer des Attachpoints des „Eltern-Objektes“, falls dieses über mehrere Attachpoints verfügt.
- Attach to Spline...: hierüber kann, ähnlich wie oben, das selektierte Objekt an eine Spline angeheftet werden. Dies vereinfacht zunächst die Platzierung, da die Koordinaten nun relativ zur Spline laufen. Der zweite, entscheidene Vorteil ist die mögliche Vermehrfachung mit den folgenden beiden Parametern...
- Distance/Range: Ein Objekt, welches an einer Spline angeheftet ist, kann automatisch mehrfach platziert werden. Range ist dabei die Gesamtentfernung, über die die Mehrfachplatzierung laufen soll, Distance ist das Intervall, also der Abstand zwischen den jeweiligen Instanzen.
- Label.../Options...: Hierüber können bei normalen Objekten die Beschriftungen geändert werden (bspw. Straßenschilder) oder bei Spezialobjekten (Haltestelle, Ampel) die Optionen eingestellt werden.
- Var Parent: Für Ampeln relevant: hier muss der „IDCode“ (steht jeweils unter der Treeview fürs gerade selektierte Objekt) des zugehörigen Kreuzungsobjektes angegeben werden, damit die Ampel „weiß“, zu welcher Kreuzung die gehört.

6.3 Splines

Splines werden zunächst einmal grundsätzlich ähnlich wie Objekte ausgewählt und platziert. Es gibt aber ein paar Ausnahmen:

Möchte man eine neue Spline einfügen, dann muss man sich entscheiden, ob man eine *unabhängige* Spline einfügen möchte, die frei platziert werden kann oder eine *abhängige*, die an eine bestehende Spline angehängt wird.

Je nachdem muss man entweder dafür sorgen, dass vorher *keine* Spline markiert ist oder eben gerade jene Spline, an die angehängt werden soll. Zu beachten

ist aber, dass die Splines immer nur „in einer Richtung“ gebaut werden können. Soll „rückwärts“ gebaut werden, muss zunächst eine unabhängige Spline erstellt werden und diese dann mit der Einrastfunktion an die bestehende Spline manuell angehängt werden (und bleibt dann lösbar).

Bei Splines gibt es natürlich einige andere Optionen als bei Objekten:

- Length: Länge der Spline (gemessen entlang der x=0-Linie) in Metern
- Radius: Radius der Spline. 0 = gerade, negative Radian bedeuten Links-, positive Radian Rechtskurven
- Angle: Winkel der Spline. Eigentlich ist dieser bereits durch Länge und Radius definiert; gibt man etwas in dieses Feld ein, so wird die Länge so angepasst, dass sich der Winkel wie gewünscht einstellt.
- Gradient: Start- und Zielsteigung der Spline (in %)
- Cant: Start- und Zielneigung der Spline (in %), negativ = links, positiv = rechts
- Mirror: spiegelt die Spline entlang der Längsachse, damit man nicht nötigerweise „richtigrum“ bauen muss, wenn die Spline asymmetrisch ist.
- Complete to...: ermöglicht das automatische Schließen von Lücken. Hierzu muss zuerst die Schaltfläche angeklickt werden, dann das eine offene Ende, dann das andere. Eins der Enden muss eine Spline sein, die auch in die entsprechende Richtung weitergebaut werden kann.

Teil III

Objektbau

Anhand von Beispielen soll nun Stück für Stück erklärt werden, wie Objekte für Omsi erstellt werden. Zunächst aber sollte der erste Teil „Allgemeines“ durchgelesen werden, weil hier einige Grundlagen erklärt werden, auf die im weiteren Verlauf immer wieder zurückgegriffen werden wird.

7 Allgemeines

7.1 Einleitung

Als Konfigurationsdateien werden sämtliche Dateien bezeichnet, die Omsi mit externen Daten bzw. Konstanten versorgen und die keinen ausführbaren Code enthalten, also nicht zu den Scripts zählen. Sie liegen stets in Textform vor und folgen alle einer eigenen, für Omsi charakteristischen Syntax. Es finden sich an diversen Stellen in Omsi Konfigurationsdateien:

- options.cfg, envir.cfg, gamectrl.cfg und keyboard.cfg für die globalen Parameter von Omsi
- global.cfg und alle dazugehörigen *.map-Dateien der Karten
- *.sco-, *.sli-, *.bus-, *.ovh- und *.hum-Dateien, welche die allgemeinen und physikalischen Daten der Szenerieobjekte, Splines, Fahrzeuge und Menschen enthalten
- Modell-, Sound-, Texturtausch-, Kabinen- und Pfaddateien mit der Endung *.cfg, welche weitere Informationen der 3D-Objekte enthalten

Die anderen beiden großen Dateigruppen sind die Scriptdateien (wozu auch Variablen- und Konstantendefinitionen gehören) und ist die Gruppe der Mesh-, Textur- und Sounddateien.

7.2 Syntax

Die Syntax der Konfigurationsdateien folgt einem Schema, welches eigentlich recht einfach ist, aber nicht üblich, weshalb es als erstes verstanden werden sollte:

Die Grundregel lautet:

Omsi interpretiert stets nur das, was nach einem Schlüsselwort folgt!

Ein Beispiel:

```
Dieses ist ein Kommentar, weil  
hiervor kein Schlüsselwort steht.  
Es ist also völlig egal,  
was hier steht!
```

```
[keyword]
```

```
1
```

```
2
```

```
3
```

Das war ein Befehl, der durch das
Schlüsselwort [keyword]
eingeleitet wurde.

Unter der Voraussetzung, dass [keyword] ein Schlüsselwort ist (das hängt von
der Datei ab; die verschiedenen Konfigurationsdateien haben unterschiedliche Schlüsselwörter),
werden die folgenden Zeilen also von Omsi interpretiert - und auch *nur* diese.

Sobald die für das Schlüsselwort typische Anzahl der Zeilen abgearbeitet wur-
de, wird wieder „Ausschau“ nach neuen Schlüsselwörtern gehalten.

Wichtig ist aber auch, dass das Schlüsselwort am Anfang der Zeile steht und
dahinter keine weiteren Zeichen, auch keine Leerzeichen folgen!

Falsch:

```
    [keyword]
```

```
1
```

```
2
```

```
3
```

Falsch:

```
[keyword] Kommentartext oder sowas
```

```
1
```

```
2
```

```
3
```

Mit diese Erkenntnis ist es auch sehr einfach, Befehle auszukommentieren - es
muss lediglich mindestens ein Zeichen hinzugefügt werden:

Gültiger Befehl:

```
[keyword]
```

```
1
```

```
2
```

```
3
```


Auskommentierter Befehl:

[keyword]

1
2
3

Auskommentierter Befehl:

[keyword]##

1
2
3

Achtung! Nicht alle Befehle stehen zwingend in eckigen Klammern! Auch jene, die nicht eingeklammert sind, folgend aber trotzdem den obigen Konventionen!

Es gibt auch einige Befehle bzw. Schlüsselwörter, die zunächst die Anzahl der nun folgenden Einträge erwarten, sodass deren „relevante Länge“ von der angegebenen Anzahl abhängt. Beispiel hierfür sind die Befehle zum Einbinden von Scripts:

[script]

3
script\script1.osc
script\script2.osc
script\script3.osc

Hier wurde zunächst die Anzahl (3) eingegeben und darauf folgend die drei Einträge, sodass Omsi weiß, dass dahinter keine weiteren folgen.

7.3 Konventionen

3D-Koordinaten DirectX und somit auch Omsi arbeitet *intern* mit einem Linkssystem, dessen x-Achse (in Fahrtrichtung) nach rechts, y-Achse nach oben und z-Achse nach vorne zeigt.

Für alle Konfigurationsdateien von Omsi gilt jedoch **grundsätzlich**:

Rechtssystem, x-Achse nach rechts, y-Achse nach vorne und z-Achse nach oben!

Dies hängt damit zusammen, dass die gängigen 3D-Programme, wie insbesondere Blender, für gewöhnlich mit dieser Konvention arbeiten.

Ferner ist die Standardeinheit für Längen METER.

Beispielsweise befindet sich folgender Punkt zwei Meter weit vor, einen Meter links und einen halben Meter über dem Referenzpunkt:

```
2
1
0.5
```

Farben Farben werden stets im RGB-System und in dieser Reihenfolge, also Rot, Grün, Blau erwartet; es gibt einige Befehle, wie bspw. der [Light]-Befehl in den Modell-Dateien, die ganzzahlige Werte zwischen 0 und 255 erwarten, ein leuchtendes Orange lautet dann z.B.

```
255
128
0
```

Anders sieht es dagegen z.B. beim [maplight]-Befehl aus, welcher Fließkommazahlen zwischen 0 und 1 erwartet. Die gleiche Farbe würde dort lauten:

```
1.0
0.5
0
```

7.4 Informationen zur Dokumentation

Den Anfang in den nun folgenden Listen der Konfig-Befehle bzw. -Schlüsselwörter macht stets ein allgemeines Syntax-Beispiel, wie z.B.:

```
[keyword]
{parameter_a}
{parameter_b}
```

Stets in Schweifklammern befinden sich die Variablen, für die später Zahlenwerte eingesetzt werden. Im weiteren Verlauf folgen dann hierfür die Erklärungen und Erläuterungen.

In einigen Fällen kommt es vor, dass die Befehle eine variable Anzahl von Einträgen zulassen - dies wird im allgemeinen folgendermaßen kenntlich gemacht:

```
[keyword]
{num_parameter}
{parameter_1}
...
{parameter_n}
```

8 Szenerieobjekte

Zunächst soll auf Szenerieobjekte zurückgegriffen werden, weil diese meistens sehr viel einfacher aufgebaut sind als Fahrzeuge.

8.1 Erstellen geeigneter x- oder o3d-Dateien

Zunächst wird also mit Blender oder einem anderen 3D-Programm ein Objekt gebaut und mit Texturen versehen. Es können dann zwei Wege beschritten werden:

Entweder wird das Objekt als x-Datei eingebunden. Dann muss der Export unter Blender mit folgenden Optionen erfolgen: „Flip Norm“ AUS, „Swap ZY“ AN, „Flip Z“ AN. Die dann exportierte x-Datei hat dann noch einen erheblichen Makel: die Materialeigenschaften sind ungeeignet. Es sind deshalb sämtliche Abschnitte folgender Art zu suchen:

```
Material Materialname {
    1.0; 1.0; 1.0; 1.0;;
    1.0;
    1.0; 1.0; 1.0;;
    0.0; 0.0; 0.0;;
    TextureFilename {    "Texturdateiname"; }
```

Es handelt sich hierbei um die Beschreibung der Materialien der x-Datei. Die erste Zeile mit viermal „1.0“ sind die Werte für Rot, Grün und Blau sowie Alpha der diffusen Farbe, die zweite Zeile ist die „Power“ (Glanzintensität), die dritte Zeile ist die Reflexionsstärke, also die Helligkeit des Glanzpunktes und die letzte Zeile (hier nur „0.0“) sind die Rot-, Grün- und Blau-Werte, die das Eigenleuchtverhalten bestimmen.

Im Allgemeinen ist es nicht erwünscht, dass die Power 1.0 und die Reflexionsfarbe weiß (1.0, 1.0, 1.0) ist, da hierbei die gesamte beleuchtete Seite des Objektes nahezu weiß wird.

Besser ist es, hier entweder Werte für eine matte Oberfläche zu wählen:

```
Material Materialname {  
    1.0; 1.0; 1.0; 1.0;;  
    1.0;  
    0.0; 0.0; 0.0;;  
    0.0; 0.0; 0.0;;  
TextureFilename {    "Texturdateiname"; }
```

Die so präparierte x-Datei kann dann wie im nächsten Kapitel erklärt eingebunden werden.

Die zweite, bessere Variante ist das Konvertieren in das Omsi-interne o3d-Format, was jedoch u.U. nicht funktioniert, wenn die x-Datei mit einem andern 3D-Programm als Blender erstellt wurde.

Hierfür wird ebenfalls zunächst eine x-Datei mit Blender exportiert, diesmal jedoch mit den Optionen „Flip Norm“ AUS, „Swap ZY“ AUS und „Flip Z“ AUS.

Dann wird „OmsiXConv“ gestartet. Dieser verfügt über folgende Optionen (in Klammern steht die jeweilige Standardeinstellung):

1. „X-File: z up“ (an): dies besagt, dass der Konverter die x-Datei so interpretiert, dass die X-Achse nach oben zeigt. Genau so haben Sie die Datei auch nach obigen Regeln exportiert, die Option soll also an sein.
2. „Separate Files“ (aus): diese Option sorgt dafür, dass die einzelnen Objekte in Blender, die gemeinsam exportiert wurden, vom Omsi-Konverter in einzelne separate o3d-Dateien umgewandelt werden, welche nach dem Input-Dateinamen jeweils den Objektnamen angehängt bekommen. Auf diese Weise kann beispielsweise ein ganzes Fahrzeug inkl. aller animierten Teile aus Blender heraus in EINE X-Datei exportiert werden, welche dann vom Omsi-X-Konverter „fachgerecht zerlegt“ wird.
3. „Remove Specular“ (an): diese Option führt Korrektur der Materialeigenschaften automatisch durch (was oben, für x-Dateien, ja manuell gemacht werden musste).
4. „Opt. Materials“ (an): hierbei löscht der Konverter automatisch Material-/Texturdubletten

5. „All Normals up“ (aus): mit dieser Option werden alle Vertexnormalen nach oben ausgerichtet. Die Beleuchtung von Pflanzen u.Ä. wirkt in so einem Fall meist realistischer.

Mittels „Load X Object“ und „Save as o3d“ kann dann die X-Datei in eine o3d-Datei umgewandelt werden, welche dann, wie im nächsten Kapitel erklärt wird, eingebunden werden kann.

8.2 Einfachstes Szenerieobjekt

Ausgangspunkt für ein Szenerieobjekt ist die Szenerieobjekt-Datei mit der Dateiendung *.sco. Sie befindet sich in einem beliebigen Unterverzeichnis von „Omsi/Sceneryobjects“.

Im einfachsten Fall verfügt die Konfigurationsdatei eines Szenerieobjekts nur über eine Handvoll Befehle:

```
[groups]
3
Buildings
Residential
Row House

[friendlyname]
Einfaches Szenerieobjekt

[mesh]
easy.o3d

[fixed]
```

Der erste Befehl [groups] gibt an, in welche Gruppen und Untergruppen im Editor das Objekt einsortiert werden soll. Die Syntax ist recht simpel: zunächst die Anzahl der Untergruppierungen (3), dann in absteigender Reihenfolge die jeweiligen Gruppennamen. Falls diese noch nicht vorhanden sind, werden **sie erstellt**, was also bedeutet, dass man sich im Rahmen eines Projektes **an seine eigenen Gruppennamen strikt halten sollte!** Andernfalls wird es zahllose Untergruppen geben und es zu einer unübersichtlichen Struktur kommen. In diesem Fall wird das Objekt also in der Gruppe „Buildings“ zu finden sein - dort in der Untergruppe „Residential“ und dort in der Untergruppe „Row House“.

Der nächste Befehl [friendlyname] dient dazu, dem Objekt einen (etwas ausführlicheren) lesbaren Namen zu geben.

Der Befehl [mesh] weist dem Objekt eine (in diesem Fall) o3d-Datei zu (dort könnte auch eine x-Datei stehen).

Der Befehl [fixed] sorgt schließlich dafür, dass dieses Objekt, was den Gruppen zufolge ein Reihenhaushaus ist, im wahrsten Sinne des Wortes **unumstößlich**, also „fest“ ist.

Der Befehl [mesh] ist der einzige in diesem Beispiel, der mehrfach verwendet werden kann. Taucht er mehrfach (sinnvollerweise mit unterschiedlichen Dateinamen) auf, werden schlicht alle aufgezählten Meshs nacheinander (!) gerendert. Anders gesagt: über die Reihenfolge der Aufzählung der [mesh]-Befehle wird die Renderreihenfolge bestimmt! Dies spielt dann eine große Rolle, wenn Transparenzen verwendet werden; dazu an der entsprechenden Stelle mehr.

Damit das Szenerieobjekt korrekt geladen werden kann, müssen *neben* der *.sco-Datei im Allgemeinen zwei Ordner liegen mit dem nötigen Inhalt: einmal „/texture“ mit den Bitmap- und Targa-Texturen, die über Blender in das Objekt eingebunden wurden und zum Anderen im „/model“-Verzeichnis die aufgerufene *.o3d- oder *.x-Datei(en) selbst!

8.3 Texturen mit niedriger Auflösung erstellen

Damit es gleichzeitig möglich ist, die gewünschten Objekte mit hochauflösenden Texturen zu versehen (bspw. 1024x1024) und dennoch für schlechtere Systeme sparsamere Texturen vorzuhalten, kann man neben die „originalen“ Texturen (mit hoher Auflösung) jeweils eine Variante mit schlechterer Auflösung legen, welche sich durch den Postfix „_#low“ unterscheidet.

Beispiel: „meinetextur.bmp“ und „meinetextur_#low.bmp“ müssen dann nebeneinander im selben „/texture“-Ordner liegen; das Objekt wird in Blender mit der Textur „meinetextur.bmp“ versehen. Wird dann die entsprechende Option in Omsi aktiviert, greift Omsi auf die kleinere Textur „meinetextur_#low.bmp“ zurück.

8.4 Nacht-/Jahreszeitentexturen, einfache Variante

Die einfachste Variante, um Nachttexturen oder (momentaner Stand) Schneetexturen zur Anwendung zu bringen, ist das Anlegen eines Textur-Unterordners im

Bestehenden mit dem Namen „WinterSnow“ bzw. „night“. Dort wird die jeweilige Variante der Textur mit **demselben** Texturnamen wie die Originaltextur hineingelegt. Auch hier können, wie im vorherigen Kapitel erklärt, Texturen mit geringerer Auflösung „beigelegt“ werden.

Da dieses Verfahren jedoch sehr unflexibel ist, gibt es ein sehr viel umfangreicheres mit Script-Unterstützung bzw. mit frei definiertbaren Variablen. Dennoch ist es sehr hilfreich, weil es meistens vollkommen ausreicht.

9 Splinetypen

9.1 Allgemeines

Als Basis für die Splines dient ein Profil, welches in den Spline-Dateien mit der Endung *.sli hinterlegt ist. Dieses Profil wird vom Editor entlang der gewählten Richtung gerade oder verbogen extrudiert, d.h. in die Länge gezogen.

Das Erstellen und Bearbeiten von Spline-Profilen (also *.sli-Dateien) ist entweder direkt mit einem Texteditor möglich. Hierzu wird weiter unten erklärt, wie die Dateien aufgebaut sind. Zunächst aber sei auf eine einfachere Möglichkeit hingewiesen, mit der mit einer recht einfach zu erstellenden *.txt-Datei (ebenfalls im Texteditor) ein neuer Straßentyp definiert werden kann und mit einem mitgelieferten Konverter namens „StreetCreator“ in eine *.sli-Datei umgewandelt werden kann.

9.2 Erstellen eines neuen Straßentyps mit dem StreetCreator

Die Bedienung des StreetCreators ist denkbar einfach: lediglich ein Klick auf „Convert Street“, dann kann man die *.txt-Datei auswählen, welche konvertiert werden soll und schon wird unter selbem Namen (jedoch der Endung *.sli) der neue Spline-Typ angelegt, welcher lediglich in das gewünschte Verzeichnis kopiert werden muss („Omsi/Splines/[eigenes Sub-Verzeichnis]“).

Der Aufbau der *.txt-Dateien für StreetCreator entspricht dem üblichen Aufbau der Omsi-Konfigurationsdateien mit Schlüsselwörtern. Grundsätzlich beschreibt die Datei den Aufbau der Straße von links nach rechts. Für jeden „Streifen“ (Gehweg, Fahrspur, Grünstreifen) gibt es einen Block, die Reihenfolge der Blöcke ergibt

dann den Aufbau der Straße.

Die Markierung [middle] markiert die Mitte der Straße - hierrüber wird insbesondere gesteuert, welche Fahrspuren in welche Richtung verlaufen!

Für Gehweg (Sidewalk), Fahrspur (lane) und Grünstreifen (grassstrip) gibt es unterschiedliche Blöcke:

9.2.1 Gehweg

Beispiel:

```
[sidewalk]
str_side1.bmp
5
0.953125
0.1875
4
0.25
0.6
0.3
0.90625
0
```

Erläuterung:

```
[sidewalk]
{Texturname (ohne Pfad)}
{Texturlänge (m)}
{Texturkoordinate Innenkante (von 0.0 bis 1.0)}
{Texturkoordinate Außenkante (von 0.0 bis 1.0)}
{Breite (m)}
{Höhe überm Boden (m)}
{Position des Gehweges, im Verhältnis der Texturbreite}
{Breite des Gehweges, im Verhältnis der Texturbreite}
{Position Bordstein, wird nicht benötigt, 0 setzen}
{Position Radweg, wird nicht benötigt, 0 setzen}
```

Mit diesem Befehl wird ein Gehweg angelegt.

Die Textur wird ohne Pfad angegeben. Die „Texturlänge“ ist das Intervall, mit der sich die Textur in Längsrichtung wiederholt. Kleine Werte stauchen die Textur, große dehnen sie. Die Texturkoordinaten für Innen- und Außenkante werden

im Verhältnis zur Breite angegeben - 0.0 ist der linke Rand der Textur, 1.0 der rechte Rand. Ausgerechnet werden kann der Wert über die Formel „Koordinate in Pixel geteilt durch Breite in Pixel“, also z.B.: Koordinate = 128 Pixel, Breite = 256 Pixel ergibt: $128 / 256 = 0.5$! Für die senkrechte Kante des Bordsteines wird automatisch der Teil der Textur verwendet, der jenseits der Innenkante liegt. Im obigen Beispiel liegt diese bei 0.953125 - d.h., für die senkrechte Kante wird automatisch der Bereich von 0.953125 bis 1.0 der Textur verwendet.

Die Höhe überm Boden liegt bei fast allen Gehwegen unserer eigenen Splines bei 0.25m, also 25cm. Bei einer Straßenhöhe von 0.1m ergibt dies eine Bordsteinhöhe von 0.15m.

Die Position und Breite des Gehweges wird ebenfalls über die Texturkoordinaten wie oben beschrieben bestimmt; dies hat den Vorteil, dass man die Koordinaten direkt im Graphikprogramm ablesen kann (und muss sie dann aber wieder durch die Gesamtbreite teilen). Soll kein Gehweg vorhanden sein, wird die Breite auf -1 gesetzt.

Die Position von Bordstein und Radweg sind Vorleistungen und werden nicht gebraucht. Sie können bei 0 belassen werden.

9.2.2 Fahrstreifen

Beispiel:

```
[lane]
str_asphdrk_1line.bmp
6
0.995
0.005
1
3
0.1
0.5
0.9
```

Erläuterung:

```
[lane]
{Texturname (ohne Pfad)}
{Texturlänge (m)}
```

```

{Texturkoordinate rechte Texturseite (von 0.0 bis 1.0)}
{Texturkoordinate linke Texturseite (von 0.0 bis 1.0)}
{1 oder -1: Spiegelung}
{Breite (m)}
{Höhe überm Boden (m)}
{Position der Fahrspur, im Verhältnis der Texturbreite}
{Breite der Fahrspur, im Verhältnis der Texturbreite}

```

Sehr ähnlich wie der Gehweg wird hiermit eine Fahrspur angelegt. Die Texturkoordinaten laufen dabei üblicherweise über die gesamte Breite (allerdings empfiehlt es sich, wie im obigen Beispiel, nicht ganz an den Rand zu gehen, da sonst durch Mipmapping noch Reste der jeweils anderen Kante sichtbar sind).

Ein neuer Parameter hier ist die Spiegelung. Hier kann entweder „1“ oder „-1“ eingegeben werden, um zu definieren, ob die Texturkoordinaten von links nach rechts (1) oder von rechts nach links (-1) laufen sollen.

Die Fahrspur wird genauso wie beim Gehweg definiert, hier üblicherweise mit einer Breite von fast 1.0 und einer Position von 0.5 (Mitte der Spur).

Wie bereits erwähnt, verwenden wir standardmäßig eine Höhe von 0.1m überm Boden.

9.2.3 Grünstreifen

Beispiel:

```

[grassstrip]
str_gruenstreifen.bmp
5
0.9727
0.36
5
0.25
0

```

Erläuterung:

```

[grassstrip]
{Texturname (ohne Pfad)}
{Texturlänge (m)}
{Texturkoordinate Innenkante (von 0.0 bis 1.0)}
{Texturkoordinate Außenkante (von 0.0 bis 1.0)}

```

{Breite (m)}
{Höhe überm Boden (m)}
{Position Bordstein, wird nicht benötigt, 0 setzen}

Der Grünstreifen wird ähnlich dem Gehweg definiert, jedoch mit dem Unterschied, dass er stets automatisch gespiegelt wird, sodass es auf beiden Seiten einen Bordstein hat. Die Breite wird über den gesamten Grünstreifen gemessen (nicht nur über die Hälfte!). Die Höhe überm Boden ist bei unseren Standardstraßen wiederum 0.25m.

Achtung: liegt der Grünstreifen (wie üblich) in der Mitte der Straße, sollte die [middle]-Markierung *vor* dem Grünstreifen kommen.