

Seminararbeit im
Fachbereich Informationstechnik
zum Thema

UNREAL EDITOR 3.0



von Andreas Janda und Tim Grothe

Kurs: TIT01DGR
Bearbeitungszeitraum: 7. April 2003 bis 30. Mai 2003

Inhaltsverzeichnis

1	Das Interface des UED 3.0	1
1.1	Der Aufbau des Editors	1
1.2	Die Navigation im Editor	2
1.2.1	Bewegen der Kamera	3
1.2.2	Bewegen von Objekten	3
1.3	Der Actor Browser	3
1.3.1	Texturen	5
1.3.2	Waffen, Medipacks und Power-Ups	5
1.3.3	Static Meshes	5
2	Die wichtigsten Funktionen für das Erstellen einer Map	6
2.1	BSP Brushes	6
2.2	Static Meshes und Movers	6
2.3	Einfügen in die Karte	8
3	Abschließende Schritte	9
4	Fazit	9
A	Quellenverzeichnis	10

Abbildungsverzeichnis

1	Veranschaulichung der Viewports	1
2	Screenshot des Editors	2
3	Screenshot des Actor Browsers	4
4	Icons der Primitives im Editor	7
5	Die Editor Icons für die CSG Operationen	8

Am 30. September 2003 wurde Unreal Tournament 2003, die Fortsetzung des beliebten 3D-Shooters Unreal Tournament offiziell für den Markt freigegeben. Wie bei der Vorgängerversion wurde auch diesmal der Unreal Editor mitgeliefert, diesmal in der Version 3.0. Der Editor ermöglicht es den Entwicklern aber auch den Spielern, dem Originalspiel neue Karten hinzuzufügen. Karten werden im Spielejargon auch als Maps bezeichnet.

Diese Seminararbeit bietet einen Überblick über die Fähigkeiten des Editors sowie der wichtigsten Funktionen, die für die Erstellung einer Karte nötig sind.

1 Das Interface des UED 3.0

1.1 Der Aufbau des Editors

Der UED3.0 besteht aus einem Haupt- und einem Kindfenster. Auf der linken Seite des Hauptfensters befindet sich eine aufklappbare Leiste mit mehreren Knöpfen, die für die Erstellung von Objekten in einer Map benötigt werden. Diese umfassen Quader, Kugeln, Treppen, aber auch die Möglichkeit mit Hilfe von 2D-Figuren neue Schablonen für 3D-Objekte anzulegen oder komplexe Videos in die Map einzufügen. Eine Erläuterung der wichtigsten Werkzeuge dieser Leiste befindet sich in Kapitel 2 auf Seite 6.

Der größte Teil des Hauptfensters wird von den sog. Viewports eingenommen. Hier erhält man standardmäßig eine 3D-Voransicht des Levels sowie drei weitere 2D Ansichten. Diese umfassen den Blick von oben auf die Map herab (Topview), den Blick von vorne auf die Map (Frontview) und die Sicht von einer Seite (Sideview). Abbildung 1 verdeutlicht dies nochmals.

In den 2D Sichten sieht man die in die Map eingebauten 3D-Objekte quasi zweidimensional zerlegt. Die Viewports verfügen ferner über ein Zoom-, sowie

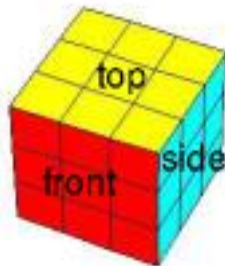


Abbildung 1: Veranschaulichung der Viewports

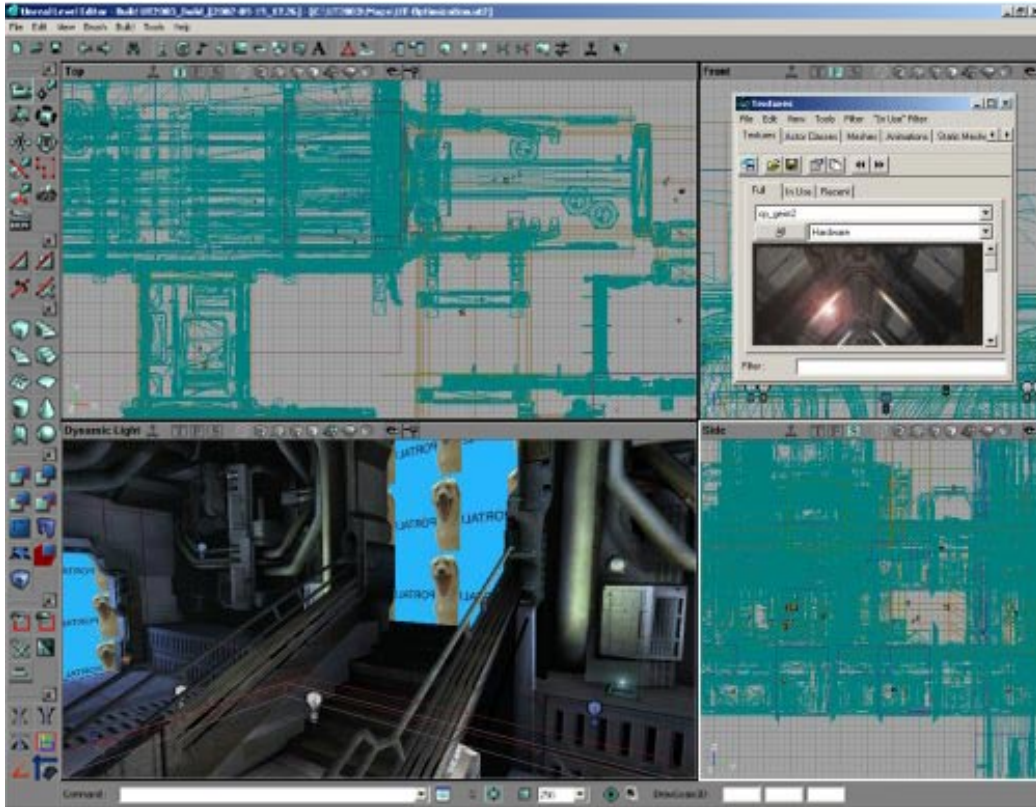


Abbildung 2: Screenshot des Editors

ein Gittersystem auf Basis der Zahl zwei. Mit Hilfe des Zooms kann man einen Kartenausschnitt vergrößern um Objekte genauer platzieren zu können. Als weitere Hilfe kann ein Objekt am Gittersystem ausgerichtet werden. Der 3D-Viewport bietet eine Voransicht des Levels, welche jedoch nicht wie im richtigen Spiel gerendert wird. Daher kann nur mittels Probespiel einer Karte, die genaue Darstellung beobachtet werden. Auf der anderen Seite kann die Voransicht, im Gegensatz zur Version im Spiel, auch so eingestellt werden, dass beispielsweise nur dynamische Lichter, das Wireframe oder die Komplexität der Karte angezeigt. Abbildung 2 zeigt einen Screenshot der Oberfläche.

1.2 Die Navigation im Editor

Der Unreal Editor unterstützt zwei verschiedene Modi der Bewegung: Erstens die Bewegung der Kamera und zweitens die Bewegung eines Objekts. Bei beiden Navigationsarten muss unterschieden werden, ob die Bewegung in einem der 2D-

oder im 3D Viewport ausgeführt werden.

1.2.1 Bewegen der Kamera

Soll die Position der Kamera im 3D-Viewport geändert werden, so muss dort die linke Maustaste gedrückt und gehalten werden. Dann bewegt sich die Kamera auf der horizontalen. Wird anstatt der linken die rechte Maustaste verwendet, so bleibt die Kamera stationär und nur der Blickwinkel verändert sich. Werden beide Maustasten gedrückt, so bewegt sie die Kamera auf der vertikalen.

Soll die Sicht in den 2D-Viewports geändert werden, so kann sowohl die linke als auch die rechte Maustaste gedrückt werden um in alle vier Richtungen zu navigieren. Drückt man beide Maustasten gleichzeitig wird bei gleichzeitiger Bewegung der Maus nach oben oder unten in die Map hinein- oder herausgezoomt. Alternativ kann die Zoomfunktion auch von einem Mousrad übernommen werden.

1.2.2 Bewegen von Objekten

Das Bewegen von Objekten funktioniert ähnlich wie die Bewegung der Kamera. Wiederum muss zwischen den Viewports unterschieden werden. Zusätzlich werden die Tasten Shift und Strg benötigt.

Hält man die Strg und die linke Maustaste gedrückt, so wird das Objekt verschoben. Verwendete man anstattdessen die Shift-Taste so wird die Kamera mit dem Objekt mitbewegt. Die rechte Maustaste lässt ein Objekt rotieren. Das Verschieben und das Rotieren von Objekten lässt sich am besten in den 2D-Viewports bewerkstelligen, da die Ansicht dort on-the-fly dargestellt wird, während der 3D-Viewport erst nach loslassen der Maustasten aktualisiert wird. Ferner können in den 2D Bereichen das Grid und der Zoom zum genauen Platzieren der Objekte in der Welt zur Unterstützung verwendet werden. Das ist sehr wichtig, da andernfalls leicht Fehler in der Map auftauchen, die die Spielbarkeit der Karte extrem einschränken können.

1.3 Der Actor Browser

Im Unreal Editor werden alle Objekte, sei es ein Brush, eine Waffe im Spiel oder die Texturen auf einer Wand, als Actor bezeichnet. Damit Actors effektiv bearbeitet werden können, gibt es den Actor Browser. Dies ist das Child-Fenster im Editor, der die Verwaltung der Actors durch verschiedene Tabs ermöglicht. Hier werden lediglich die wichtigsten Elemente vorgestellt. Abbildung 3 zeigt ein Bild des Actor Browser.

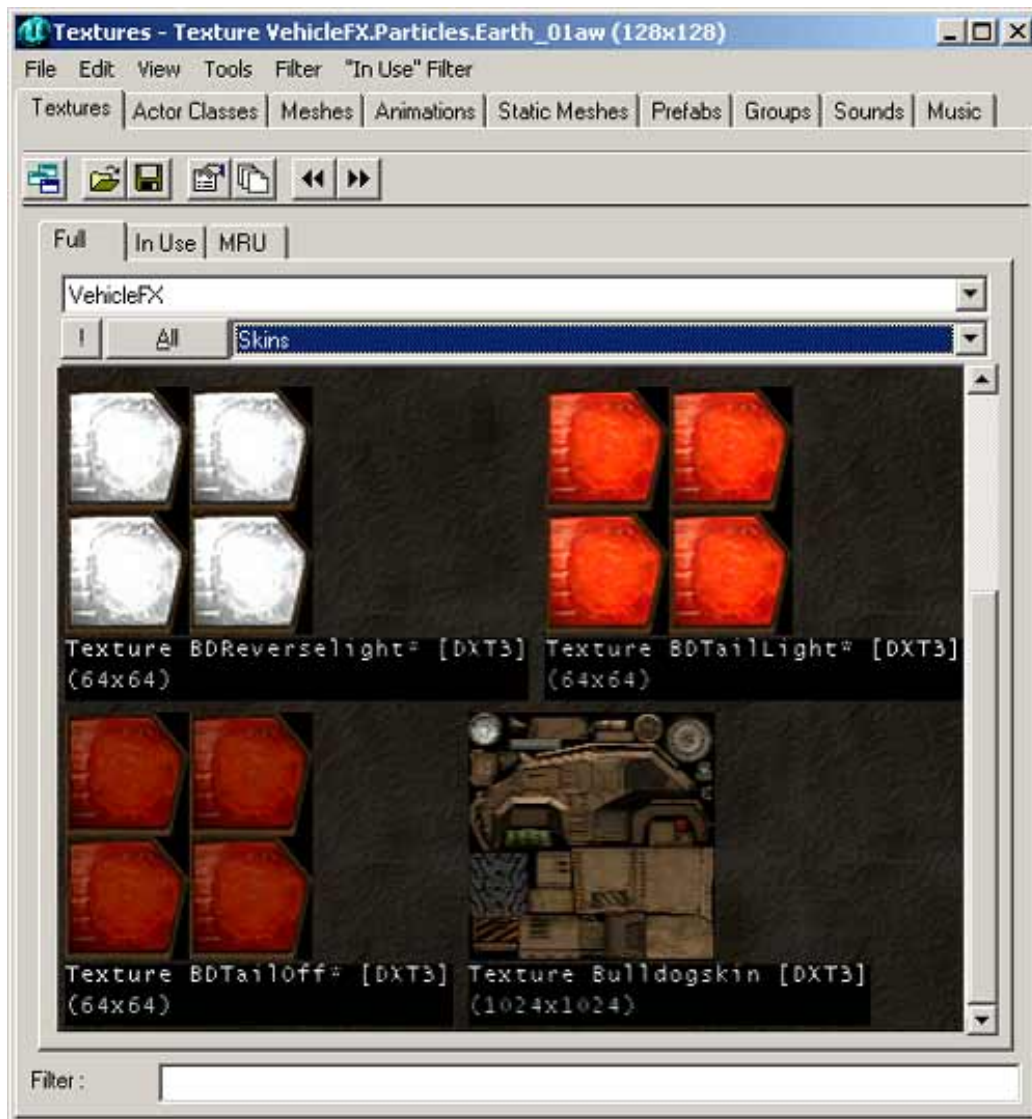


Abbildung 3: Screenshot des Actor Browsers

1.3.1 Texturen

Der Textures Tab erlaubt es, ein CSG Brush [s. S. 2.3] mit einer Textur zu versehen. Dazu können verschiedene Texturpakete geladen werden. Die meisten Pakete enthalten Texturen für Wände, Böden und Decken, die in entsprechenden Gruppen sortiert und verwaltet werden können. Wurde ein Texturpaket geöffnet, so zeigt der Textures Browser eine Vorschau der Textur sowie die Größe. Wurde eine Textur platziert, so kann diese skaliert, gedreht und verschoben werden um so an den jeweiligen CSG Brush angepasst zu werden.

1.3.2 Waffen, Medipacks und Power-Ups

Für diese Elemente, aber auch für Trigger (Auslöser für Ereignisse) und Emitter (spezielle Lichteffekte, z.B. für Funken), gibt es einen generellen Actor Tab. Dieser ermöglicht es zu definieren, welches dieser Objekte eingefügt werden soll. Das Einfügen ansich geschieht dann über ein dynamisches Kontextmenü in den jeweiligen Viewports. Diese Objekte können nach dem Einfügen, ähnlich zu den Texturen, über ein baum-strukturiertes Optionen Menü weiter angepasst werden.

1.3.3 Static Meshes

In diesem Tab werden die sog. Static Meshes verwaltet (mehr dazu in Kapitel 2.2). Static Meshes können nachgeladen und in einer 3D Voransicht betrachtet werden. Weiterhin können Angaben zur Kollisionsabfrage gemacht werden. Pakete bestehend aus mehren Static Meshes werden ähnlich zu den Texturen in Gruppen verwaltet. Static Meshes werden nach Auswahl über den Add-to-world oder je nach Situation über den Add-Mover-Brush Button eingefügt.

2 Die wichtigsten Funktionen für das Erstellen einer Map

Hier werden die wichtigsten Funktionen des Unreal Editors vorgestellt, die zur Erstellung einer einfachen Map nötig sind. Diese Werkzeuge werden auch als Primitives bezeichnet. Eine genauere Erläuterung aller Primitives kann unter [\[1\]](#) nachgelesen werden.

2.1 BSP Brushes

Die einfachste Form eines Objektes sind die BSP Brushes. BSP steht für Binary Space Partitioning und beschreibt eine Datenstruktur zum Organisieren von Objekten in einer Karte. Das eigentliche Objekt wird für die Unreal Engine durch CSG (Constructive Solid Geometry) beschrieben. Brushes werden benötigt um die Mapgrenzen sowie die Grundform der Map festzulegen. Die wichtigsten Elemente sind der Cube-, der Cylinder-, der Stairs- und der Moverbrush.

Der Cubebrush wird benötigt um Würfel und Quader anzulegen. Dazu muss man die Höhe, Länge und Breite des Cubes angeben. Ferner kann festgelegt werden, ob der Cube hohl oder solide sein soll.

Der Cylinderbrush erzeugt einen Zylinder. Die Parameter sind hier der innere und äußere Radius, die Anzahl der Zwischenschritte, die für die Rundung verwendet werden sollen sowie die Höhe des Zylinders.

Der Stairsbrush kann für einfache Treppen verwendet werden. Standardmäßig können gerade, spiral- oder kurvenförmige Treppen erstellt werden. Als Parameter nimmt dieser Brush wieder einen inneren und äußeren Radius, sowie die Höhe in Anspruch.

Mover sind spezielle Brushes, die für Objekte benötigt werden, die sich in der Map bewegen sollen (eine Tür oder ein Aufzug beispielsweise). Dies wird im Punkt [2.2](#) genauer erläutert.

2.2 Static Meshes und Movers

Static Meshes sind fertige Objekte, die in eine Map eingefügt werden können. Sie sind der Hauptbestandteil einer Map und demonstrieren die Stärke der Un-

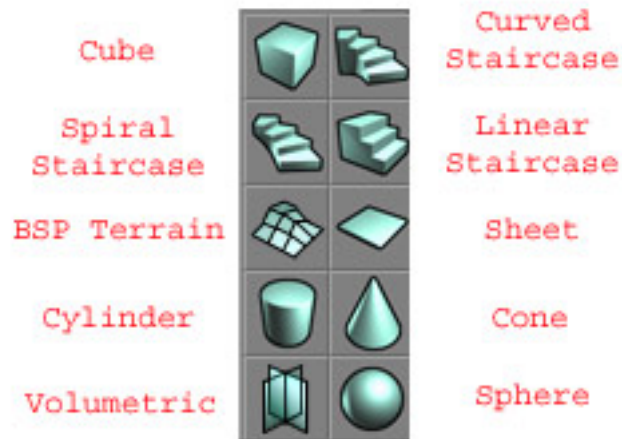


Abbildung 4: Icons der Primitives im Editor

real Engine. Static Meshes werden normalerweise mit einem externen Tool wie beispielsweise dem 3D Studio Max erstellt, können aber auch im Unreal Editor aus Brushes erstellt werden. Da bei Static Meshes die Größe, die Form und die Texturen bereits feststehen, können sie komplett im Grafikchip durch die Transform&Lighting Engine berechnet werden. Das bringt vor allem bei sehr komplexen Meshes (Anzahl der Polygone >1000) einen enormen Geschwindigkeitsvorteil. Ferner muss für jeden Static Mesh nur einmal Speicher angefordert werden. Wird dieser Mesh ein weiteres Mal eingefügt, so wird kein weiterer Speicher verbraucht, auch dann nicht, wenn der Mesh rotiert oder skaliert wird. Daher werden Static Meshes oft auch als HardwareBrushes bezeichnet.

Frühere Versionen des Unreal Editors haben erlaubt, BSP Brushes als bewegliche Objekte einzusetzen. Ab Version 3.0 geht dies nur noch mit Static Meshes. Soll beispielsweise ein Aufzug in die Map implementiert werden, so muss erst ein entsprechender Static Mesh geladen bzw. erstellt und anschließend als Mover in die Map eingefügt werden.

2.3 Einfügen in die Karte

Wenn eine neue Map erstellt wurde so ist die Welt quasi komplett gefüllt. Wird dann ein BSP Brush in die Welt eingefügt, muss man diesen BSP Brush quasi aus der Welt herausschneiden. In bereits herausgeschnittene Brushes können andere Brushes wieder hinzugefügt werden (beispielsweise eine Trennwand). Um das zu bewerkstelligen werden die Werkzeuge Add und Subtract benötigt. Wird eines dieser Werkzeuge angewandt, so wird der BSP Brush zu einem CSG Brush konvertiert. Dieser enthält dann Werte für die Kollisionsabfrage der Engine.

Neben diesen beiden Werkzeugen stehen zusätzlich Intersect und De-Intersect zur Verfügung. Intersect passt einen erstellten Brush an die Materie an (ein schon erstellter Brush wird quasi von außen umfahren), während das De-Intersect Werkzeug einen Brush von innen umfährt. Mit Hilfe von diesen beiden Werkzeugen können kompliziertere Brushes als die Standard Typen erstellt werden.

Static Meshes können nicht mit diesen Werkzeugen verwendet werden. Für sie gibt es einen speziellen Button im Static Mesh Actor Browser. Soll ein Static Mesh als Mover eingefügt werden, so benötigt man das Mover Werkzeug.

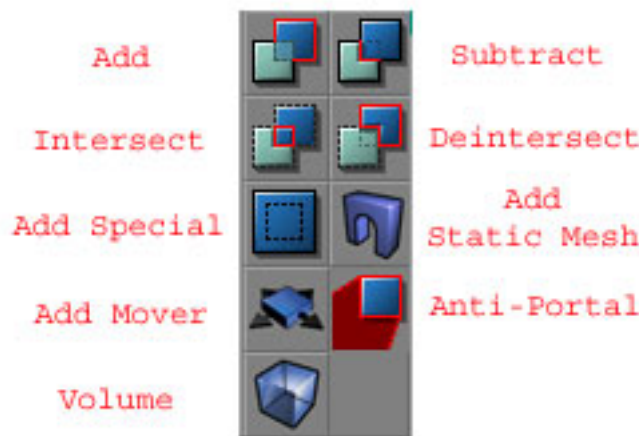


Abbildung 5: Die Editor Icons für die CSG Operationen

3 Abschließende Schritte

Mit Hilfe der vorgestellten Möglichkeiten und Funktionen kann bereits eine einfache Karte erstellt werden. Es müssen lediglich noch ein Startpunkt für die Spieler sowie Lichter eingebaut werden, da die Karte ansonsten völlig schwarz ist. Dann kann die Karte kompiliert werden. Dabei werden Informationen über alle eingefügten Actors gesammelt und in einem Format gespeichert, welches von der Engine interpretiert werden kann. Treten während des Kompilierens Fehler auf, so erscheint eine entsprechende Fehlermeldung.

Ein wichtiges Feature, welches noch nicht behandelt wurde, umfasst den Bereich der KI. Soll in einer Map die KI des Spiels aktiv werden, muss ihr vorher gesagt werden, welche Wege in der Map möglich sind, welche Trigger welches Ereignis steuern und wo die Waffen liegen. Dazu verwendet die Unreal Engine sog. Path Nodes, die der KI anzeigen, welche Möglichkeiten eine Map bietet. Durch die Path Nodes kennt die KI schon die ganze Karte, während ein Spieler diesen Vorteil nicht erhält.

Wurden in der Map Stellen eingebaut, die eine komplexe Spielerbewegung erfordern (beispielsweise einen Aufzugjump zum Erreichen eines guten Powerups), so reichen die Path Nodes nicht aus. An dieser Stelle springt das Unreal Action Script ein. Dies ist eine prozedurale Sprache, die im Editor eingebaut ist. Sie basiert auf der C Syntax. Mit Hilfe des Action Scripts lassen sich komplexe Vorgänge für die KI aber auch für die Steuerung der Elemente einer Map modellieren.

4 Fazit

Der Unreal Editor 3.0 repräsentiert ein Tool für die Erstellung von Maps für die Unreal 3D Engine. Der Aufbau ist logisch nachvollziehbar und die Funktionen sind im Internet gut dokumentiert. Zahlreiche Tipps und Tricks stehen in Foren und Newsgroups zur Unterstützung bereit. Auf der anderen Seite ist dieses Programm so komplex, dass ein gewisser Einarbeitungsaufwand unumgänglich ist. Ferner sollten die entsprechend hohen Hardwarevoraussetzungen eingehalten werden, um vernünftiges Arbeiten zu ermöglichen. Wurden die Einstiegshürden gemeistert, so steht einem Mapdesigner ein anspruchsvolles aber auch leistungsstarkes Tool zur Verfügung, mit dessen Hilfe die Unreal Engine voll ausgereizt werden kann.

A Quellenverzeichnis

1. http://udn.epicgames.com/pub/Content/UnrealEdInterface/#Brush_Primitives
2. <http://udn.epicgames.com>
3. http://www.planetunreal.com/architectonic/first_level_part1.html
4. <http://unreal3.planet-multiplayer.de>