



Destruction Racer™

-- *d!racer* --

Clemens Brandorff (9526005)
Andreas Maierhofer (0027599)
Roland Sattler (9525932)

Spielzusammenfassung

Spielbeschreibung

Destruction Racer fällt in die Klasse der Autorennspiele, wobei es darum geht, einen in freier Natur angesiedelten Renn-Parcour zu durchqueren. In der Landschaft finden sich verschiedene Checkpoints, die der Spieler in gewisser Zeitvorgabe mit seinem Vehikel durchqueren sollte. Die Steuerung des Fahrzeugs obliegt vollständig dem Spieler, wobei es keine Vorgaben gibt, auf welchem Weg die diversen Checkpoints erreicht werden müssen, solange die richtige Checkpoint-Reihenfolge eingehalten wird. Werden alle Checkpoints in richtiger Reihenfolge und entsprechender Zeit durchquert so hat der Spieler das Ziel erreicht.

Als Entwicklungsplattform wird sowohl *Windows* als auch *Linux* verwendet, wobei die Plattformunabhängigkeit durch Verwendung von *OpenGL* und *SDL* gewährleistet wird. Für Spielobjekte werden hauptsächlich 3D-Studio Objekte eingesetzt werden. Weiters wird zur Teamunterstützung eine *CVS-Umgebung* verwendet um die gemeinsame Entwicklung besser zu koordinieren.

3D-Objekte – Texturierung, Beleuchtung, ...

Im Spiel treten folgende 3D-Objekte auf:

- **Landschaft** (terrain)

Das Renngelände wird durch eine *heightmap* definiert und mittels eines *binary-triangle-tree* als Datenstruktur verwaltet bzw. gerendert. Als Texturierung der Spielwelt wird *prozedurale Texturierung* je nach Höheninformation verwendet (*2-pass multitexturing*). Die Beleuchtung der Spielwelt wird vom OpenGL übernommen, wobei die Spielwelt als gekrümmte Fläche angesehen wird. Die Landschaft wird gegen das Kamera-Frustum geclipped.

- **Vehikel**

Dabei handelt es sich um vom Spieler steuerbare Objekte in Form eines Fahrzeugs, die durch das Spielgelände frei bewegt werden können. Sie werden durch 3ds-max Objekte repräsentiert, mit verschiedenen Texturen versehen und gerendert. Die Beleuchtung übernimmt OpenGL, wobei angenommen wird, dass das Vehikel kantig ist (d.h. ein Normalvektor pro Polygon, da die Basisklasse, die für das

- **Checkpointobjekte**

Laden der Objekte zuständig ist, nicht in der Lage ist „smoothing groups“ aus dem 3dsmax zu exportieren, und das gesamte Vehikel als gekrümmte Fläche anzunehmen nicht gut aussieht.)

Die Checkpoints werden statisch in der Spiellandschaft platziert und entsprechen leicht erkennbaren 3D-Objekten, die vom Spieler zu durchqueren sind. (Repräsentation siehe Vehikel). Jeder Checkpoint verfügt über eine Nummer, die leicht ersichtlich am Checkpoint erkennbar ist.

- **Himmel (sky)**

Der Himmel ist mittels einer animierten *skydome* gestaltet.

- **Landschaftsobjekte**

In der Weltmitte findet sich eine Steinskulptur, die ursprünglich einen anderen Verwendungszweck haben sollte ☺

Spielsteuerung – 3D-Ansicht, bewegte Objekte...

Als Kameramodelle stehen eine Verfolgerkamera (default), die dem Auto folgt, und eine frei bewegliche Kamera zur Verfügung.

W / S	Kamera Vor / Zurück
A / D	Kamera Seitlich
Q / E	Kamera Roll
UP / DOWN	Kamera Pitch
LEFT / RIGHT	Kamera Heading
U	Vehikel Gas
J	Vehikel Bremse
H / K	Vehikel Links / Rechts
F10 bzw. ^	Menükonsole
T	Speed'o'meter ein/aus
M	Minimap ein/aus
F1	Verfolger-Kamera ein/aus
F2	Framerate-Anzeige ein/aus
F3	Wireframe/Filled Modus
F5	Umschalten der MipMap-Filter
F7	Display Lists ein/aus
F8	View frustum culling ein/aus
ESC	Spiel verlassen

Menübefehle

quit	Spielende
reset	Spielneustart
wf	Wireframe Modus
fill	Filled Modus
sky on	Himmel ein
sky off	Himmel aus
tacho on	Tacho ein
tacho off	Tacho aus
map on	Minikarte ein
map off	Minikarte aus

Verwendete Bibliotheken / zusätzlicher Code

• Open GL	http://www.opengl.org
• SDL	http://www.libsdl.org Application Window
• SDL_ttf	http://www.libsdl.org/projects/SDL_ttf/ Einlesen von TrueType Fonts
• 3dsFileLoader	http://www.tsrevolution.com/ Basisklasse zum Einlesen von 3ds Dateien
• 3dsFileFormat	http://www.tsrevolution.com/ Basisklasse zum Einlesen von 3ds Dateien

Neuerungen seit der zweiten Abgabe

• Spielsystem	Es ist nun möglich einen Rennparcour mit Zeitmessung zu durchqueren. Werden die Checkpoints in gegebener Reihenfolge durchquert, hat man „gewonnen“ ☺
• Kamera	Zusätzlich zur getrennten Steuerung von Auto und Kamera, also zum freien Kameramodell, gibt es nun auch noch eine Verfolgerkamera, die sich am Auto orientiert.

• Steuerung	Zusätzliche Tastatur- und Menübefehle erlauben mehr Eingriffsmöglichkeiten ins Rendering bzw. Gameplay.
• Autophysik	Das Auto verhält sich realistischer vor allem unter Gravitationseinwirkung, so dass nun auch z.B. Sprünge möglich sind. Weiters sollte es zu weniger Clipping-Fehlern mit der Landschaft kommen.
• Landschaft	Die prozedurale Texturierung durch <i>two-pass rendering</i> und Texturblendung ermöglicht eine wesentlich realistischere Landschaftsdarstellung. Ein bewegter Himmel (<i>skydome</i>) rundet die Welt Darstellung ab.
• Zusätzliche Objekte	Neu im Spiel sind texturierte Checkpoints, eine Minimap, die die Rennstrecke mit den darauf befindlichen Objekten beschreibt, bewegter Himmel, ...
• View frustum culling	View frustum culling funktioniert nun, ohne dass Dreiecke am Rand abgeschnitten werden. Weiters werden Objekte (Checkpoints, Landschaftsobjekte, ...), die sich nicht im view frustum befinden, nicht gezeichnet.
• Nebeleffekt	In Bodennähe wird mittels <i>volumetric fog</i> die Illusion von Bodennebel erzeugt. (GL_EXT_fog_coord)

Wir haben das Spiel auf mehreren Windows Systemen und unter Linux getestet und hoffen daher, dass es ohne Schwierigkeiten start- und lauffähig ist. Das Paket enthält alle Quelldateien (*./src*) und zusätzlichen libraries (*./src/lib*) sowie include Dateien (*./src/include*), die für ein Kompilieren des Projektes unter Windows notwendig sind.

Viel Vergnügen!

Clemens Brandorff (cthulhu@chello.at)
 Roland Sattler (ludi@netpratique.fr)
 Andreas Maierhofer (am@antitachyon.com)