Einführung in die Grundlagen von 3D-Engines

Teil 2: Grundlagen von Game Engines

Markus Kleffmann, Markus Muckhoff

Xpiriax Software

RPC 2009

Themenübersicht

- Wie baut man eine Game Engine?
- Was braucht man für die Erstellung einer Game Engine?
- Warum sollte man eine eigene Engine schreiben?

Wie baut man eine Game Engine?

Wie baut man eine Game Engine?

Entwicklung einer rudimentären Game Engine in fünf Schritten:

- Schritt 1: Grafik-Engine
- Schritt 2: Input
- Schritt 3: Loading (und Erweiterung der Grafik-Engine)
- Schritt 4: Physiksystem
- Schritt 5: Sound-Engine

Grafik-Engine

Grafik-Engine

- sorgt für die grafische Darstellung von 2D- und 3D-Grafik
- übernimmt Licht- und Schattenberechnung
- erzeugt grafische Effekte wie Feuer, Wasser, Schnee, Nebel (Shader)
- Performance ist hier sehr wichtig (deshalb in der Regel eine der komplexesten Komponenten einer Game Engine)
- heutzutage arbeiten fast alle Grafik-Engines in 3D (daher der Name 3D-Engine)

Grafik-Engine

- sorgt für die grafische Darstellung von 2D- und 3D-Grafik
- übernimmt Licht- und Schattenberechnung
- erzeugt grafische Effekte wie Feuer, Wasser, Schnee, Nebel (Shader)
- Performance ist hier sehr wichtig (deshalb in der Regel eine der komplexesten Komponenten einer Game Engine)
- heutzutage arbeiten fast alle Grafik-Engines in 3D (daher der Name 3D-Engine)

Grafik-Engine

- sorgt für die grafische Darstellung von 2D- und 3D-Grafik
- übernimmt Licht- und Schattenberechnung
- erzeugt grafische Effekte wie Feuer, Wasser, Schnee, Nebel (Shader)
- Performance ist hier sehr wichtig (deshalb in der Regel eine der komplexesten Komponenten einer Game Engine)
- heutzutage arbeiten fast alle Grafik-Engines in 3D (daher der Name 3D-Engine)

Grafik-Engine

- sorgt für die grafische Darstellung von 2D- und 3D-Grafik
- übernimmt Licht- und Schattenberechnung
- erzeugt grafische Effekte wie Feuer, Wasser, Schnee, Nebel (Shader)
- Performance ist hier sehr wichtig (deshalb in der Regel eine der komplexesten Komponenten einer Game Engine)
- heutzutage arbeiten fast alle Grafik-Engines in 3D (daher der Name 3D-Engine)

Grafik-Engine

- sorgt für die grafische Darstellung von 2D- und 3D-Grafik
- übernimmt Licht- und Schattenberechnung
- erzeugt grafische Effekte wie Feuer, Wasser, Schnee, Nebel (Shader)
- Performance ist hier sehr wichtig (deshalb in der Regel eine der komplexesten Komponenten einer Game Engine)
- heutzutage arbeiten fast alle Grafik-Engines in 3D (daher der Name 3D-Engine)

Auswahl der Multimedia-API

Empfehlung für Anfänger: DirectX

Gründe:

- die meisten Bücher über Spieleprogrammierung behandeln DirectX
- fast alle kommerziellen Spielefirmen verwenden DirectX

Nachteil:

- man ist nicht plattformunabhängig und muss unter Windows entwickeln

Auswahl der Multimedia-API

Empfehlung für Anfänger: DirectX

Gründe:

- die meisten Bücher über Spieleprogrammierung behandeln DirectX
- fast alle kommerziellen Spielefirmen verwenden DirectX

Nachteil:

- man ist nicht plattformunabhängig und muss unter Windows entwickeln

Auswahl der Multimedia-API

Empfehlung für Anfänger: DirectX

Gründe:

- die meisten Bücher über Spieleprogrammierung behandeln DirectX
- fast alle kommerziellen Spielefirmen verwenden DirectX

Nachteil:

- man ist nicht plattformunabhängig und muss unter Windows entwickeln

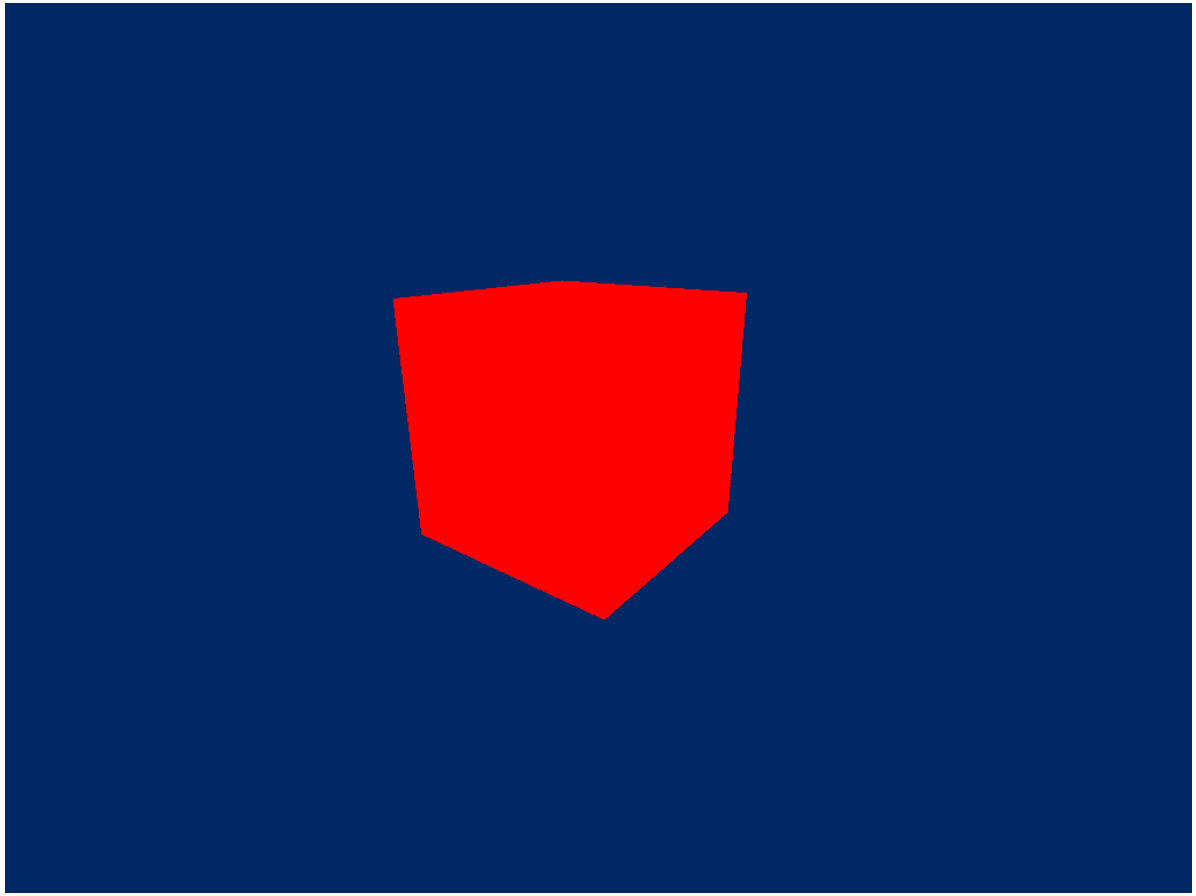
Das erste Bild

Entwicklungszeit: ca. 2 Stunden

Woher kommen die 3D-Daten?

- im ersten Schritt:
  - 3D-Daten fest in den Quellcode schreiben (Klassiker: Würfel)
  - keine Textur (nur Flat Shading)
- Vorteil: man kann sehr schnell etwas sehen
- nur für extrem einfache Objekte sinnvoll möglich

Das erste 3D-Objekt



Entwicklungszeit (Objekt fest einkodiert): ca. 1 Stunde

Input

Input

- verwaltet die diversen Eingabegeräte (Tastatur, Maus, Joystick etc.)
- ermöglicht die Steuerung des Spiels
- Komplexität ist sehr gering

Input

- verwaltet die diversen Eingabegeräte (Tastatur, Maus, Joystick etc.)
- ermöglicht die Steuerung des Spiels
- Komplexität ist sehr gering

Input

- verwaltet die diversen Eingabegeräte (Tastatur, Maus, Joystick etc.)
- ermöglicht die Steuerung des Spiels
- Komplexität ist sehr gering

Ein einfaches Input-System

- Eingabegeräte werden in bestimmten Intervallen abgefragt (z.B. alle 5 ms)
- Engine reagiert auf die Benutzereingaben (z.B. Taste „Pfeil-Nach-Oben“ bewegt den Spieler einen Schritt nach vorne)

Entwicklungszeit: ca. 2 Tage

Schritt 3

Loading

Loading

Um komplexere 3D-Objekte und andere Daten (z.B. Texturen, Sounds) zu laden, brauchen wir ein Loading-System.

Tipp

Am Anfang so wenig wie möglich selbst implementieren - statt dessen fertige Libraries verwenden. Z.B.:

- Laden von Grafiken: libpng
- Laden von Sounds: OpenAL

Loading

Um komplexere 3D-Objekte und andere Daten (z.B. Texturen, Sounds) zu laden, brauchen wir ein Loading-System.

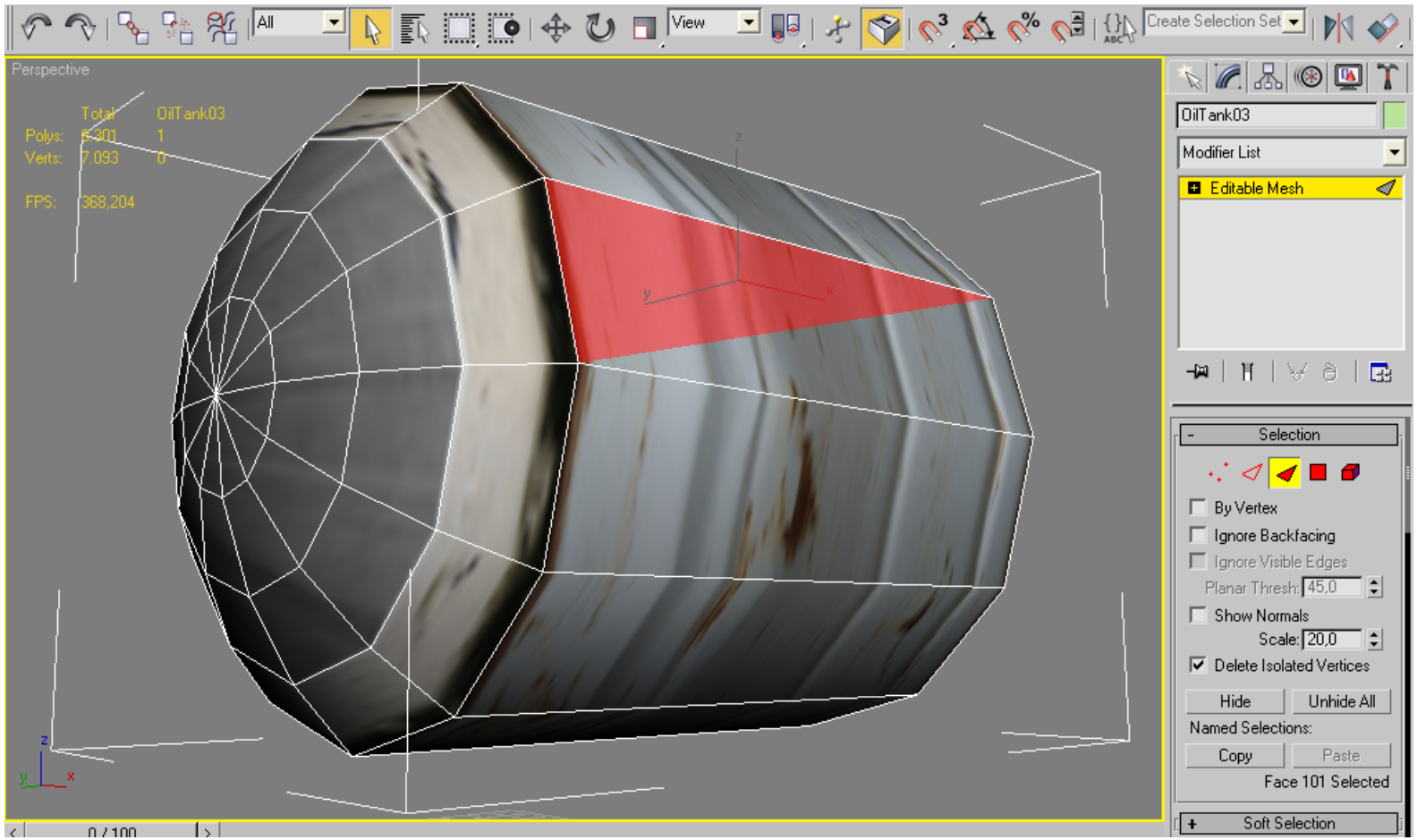
Tipp

Am Anfang so wenig wie möglich selbst implementieren - statt dessen fertige Libraries verwenden. Z.B.:

- Laden von Grafiken: libpng
- Laden von Sounds: OpenAL

Woher kommen die 3D-Daten?

- 3D-Modelle mit einem entsprechenden Programm modellieren



Woher kommen die 3D-Daten?

- Modelle exportieren
  - am einfachsten: DirectX Mesh Files, Dateiendung .x
  - am flexibelsten: eigenes Dateiformat
- Modell-Datei enthält Informationen über das 3D-Modell
  - Vertex-Daten
  - Textur-Informationen
  - Animation
- anfangs: Modell-Datei in der Engine laden und Inhalt entsprechend aufbereiten
- später: Daten *vor* dem Laden aufbereiten (Geschwindigkeits-Optimierung)

Woher kommen die 3D-Daten?

- Modelle exportieren
  - am einfachsten: DirectX Mesh Files, Dateiendung .x
  - am flexibelsten: eigenes Dateiformat
- Modell-Datei enthält Informationen über das 3D-Modell
  - Vertex-Daten
  - Textur-Informationen
  - Animation
- anfangs: Modell-Datei in der Engine laden und Inhalt entsprechend aufbereiten
- später: Daten *vor* dem Laden aufbereiten (Geschwindigkeits-Optimierung)

Woher kommen die 3D-Daten?

- Modelle exportieren
  - am einfachsten: DirectX Mesh Files, Dateiendung .x
  - am flexibelsten: eigenes Dateiformat
- Modell-Datei enthält Informationen über das 3D-Modell
  - Vertex-Daten
  - Textur-Informationen
  - Animation
- anfangs: Modell-Datei in der Engine laden und Inhalt entsprechend aufbereiten
- später: Daten *vor* dem Laden aufbereiten (Geschwindigkeits-Optimierung)

Woher kommen die 3D-Daten?

- Modelle exportieren
  - am einfachsten: DirectX Mesh Files, Dateiendung .x
  - am flexibelsten: eigenes Dateiformat
- Modell-Datei enthält Informationen über das 3D-Modell
  - Vertex-Daten
  - Textur-Informationen
  - Animation
- anfangs: Modell-Datei in der Engine laden und Inhalt entsprechend aufbereiten
- später: Daten *vor* dem Laden aufbereiten (Geschwindigkeits-Optimierung)

Das erste geladene 3D-Objekt

Entwicklungszeit (mit X-Files): ca. 1 Stunde

Entwicklungszeit (eigenes Dateiformat): ca. 2 Wochen

Eine fortgeschrittene Grafik-Engine

Entwicklungszeit: ca. 6 Monate

Physiksystem

Physiksystem

- simuliert alle physikalischen Effekte im Spiel, z.B.
  - Schwerkraft
  - Festkörperphysik, sog. Rigid-Body-Physik
- ist für die Kollisionserkennung zuständig (Collision Detection und Collision Response)
- fast alle Effekte werden gefaked, da eine korrekte physikalische Berechnung
  - meist zu aufwändig ist (z.B. Simulation von Flüssigkeiten)
  - oft unnötig ist (z.B. Lichtberechnung)
  - den Spielfluss zerstören kann (z.B. Sound im Weltraum)

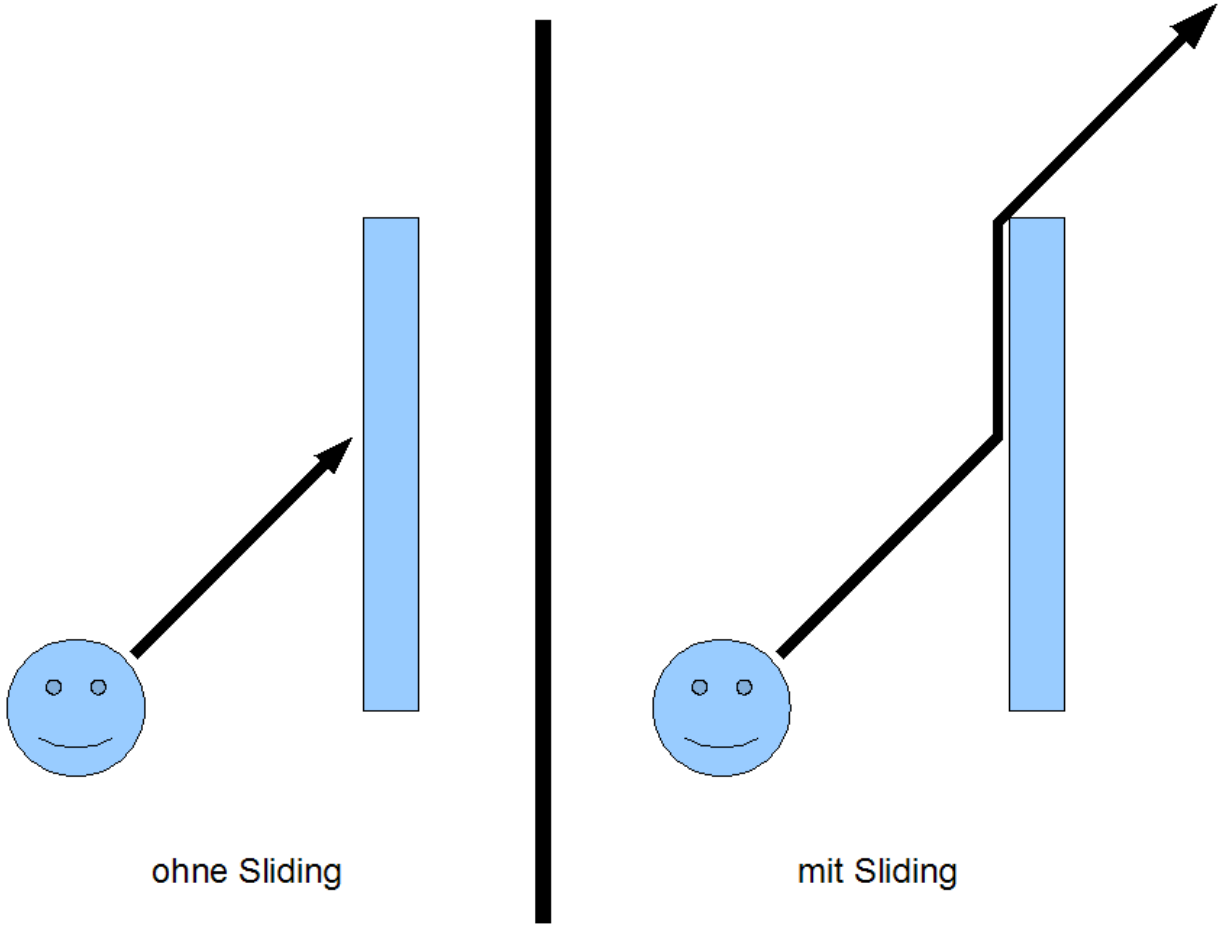
Physiksystem

- simuliert alle physikalischen Effekte im Spiel, z.B.
  - Schwerkraft
  - Festkörperphysik, sog. Rigid-Body-Physik
- ist für die Kollisionserkennung zuständig (Collision Detection und Collision Response)
- fast alle Effekte werden gefaked, da eine korrekte physikalische Berechnung
  - meist zu aufwändig ist (z.B. Simulation von Flüssigkeiten)
  - oft unnötig ist (z.B. Lichtberechnung)
  - den Spielfluss zerstören kann (z.B. Sound im Weltraum)

Physiksystem

- simuliert alle physikalischen Effekte im Spiel, z.B.
  - Schwerkraft
  - Festkörperphysik, sog. Rigid-Body-Physik
- ist für die Kollisionserkennung zuständig (Collision Detection und Collision Response)
- fast alle Effekte werden gefaked, da eine korrekte physikalische Berechnung
  - meist zu aufwändig ist (z.B. Simulation von Flüssigkeiten)
  - oft unnötig ist (z.B. Lichtberechnung)
  - den Spielfluss zerstören kann (z.B. Sound im Weltraum)

Beispiel: Sliding ist physikalisch nicht korrekt



Rudimentäres Physiksystem

Entwicklungszeit: ca. 6 Monate

Sound-Engine

Sound-Engine

- Aufgaben:
  - atmosphärische Hintergrundmusik abspielen
  - zur Szene passende Soundeffekte erzeugen (z.B. Vogelgezwitscher im Wald, Rauschen eines Flusses)
  - Soundeffekte zu bestimmten Aktionen abspielen (z.B. Trittgeräusche beim Laufen, Schussgeräusche)
- besitzt eine sehr geringe Komplexität
- ist trotzdem enorm wichtig, da Sound sehr viel zur Atmosphäre beiträgt - mit Sound wirkt eine Szene lebendiger und realistischer

Sound-Engine

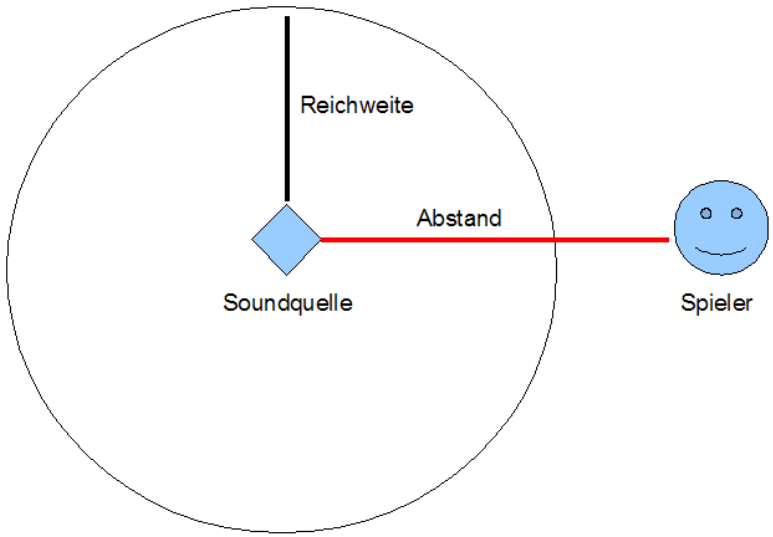
- Aufgaben:
  - atmosphärische Hintergrundmusik abspielen
  - zur Szene passende Soundeffekte erzeugen (z.B. Vogelgezwitscher im Wald, Rauschen eines Flusses)
  - Soundeffekte zu bestimmten Aktionen abspielen (z.B. Trittgeräusche beim Laufen, Schussgeräusche)
- besitzt eine sehr geringe Komplexität
- ist trotzdem enorm wichtig, da Sound sehr viel zur Atmosphäre beiträgt - mit Sound wirkt eine Szene lebendiger und realistischer

Sound-Engine

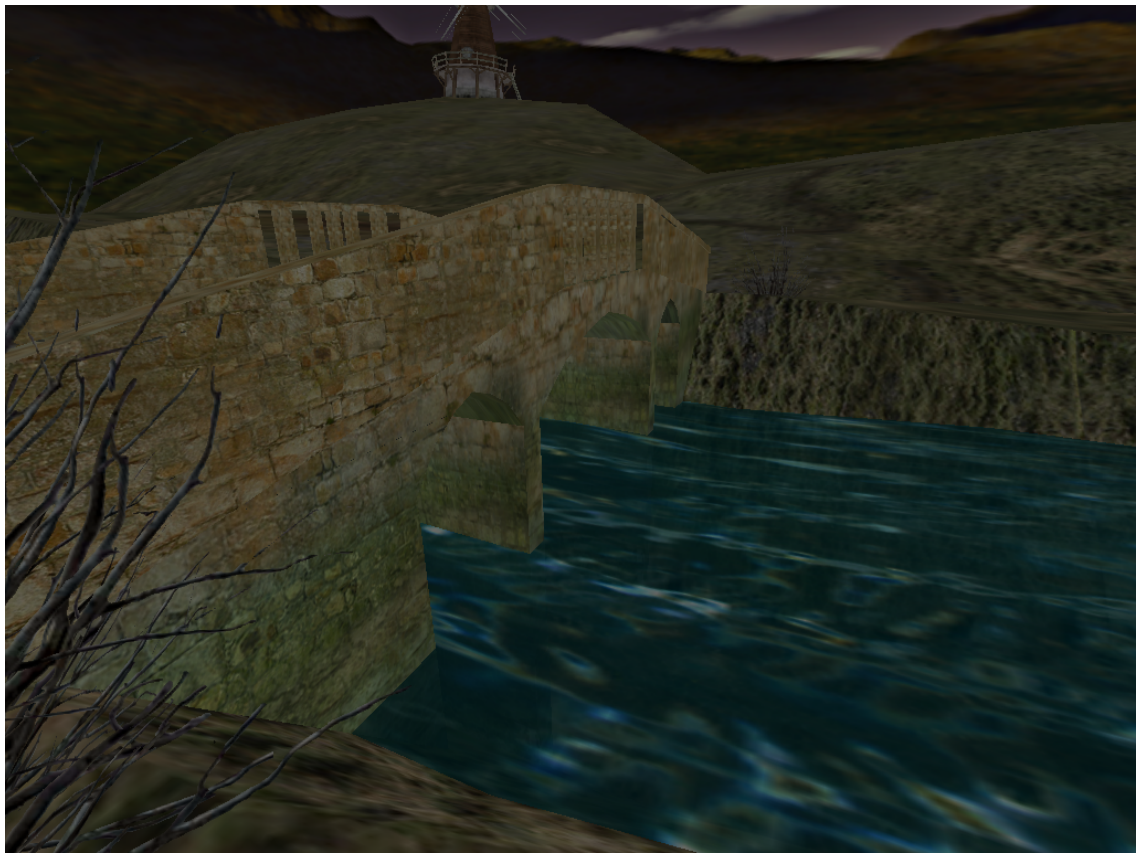
- Aufgaben:
  - atmosphärische Hintergrundmusik abspielen
  - zur Szene passende Soundeffekte erzeugen (z.B. Vogelgezwitscher im Wald, Rauschen eines Flusses)
  - Soundeffekte zu bestimmten Aktionen abspielen (z.B. Trittgeräusche beim Laufen, Schussgeräusche)
- besitzt eine sehr geringe Komplexität
- ist trotzdem enorm wichtig, da Sound sehr viel zur Atmosphäre beiträgt - mit Sound wirkt eine Szene lebendiger und realistischer

Lautstärke von Soundquellen

- alle Soundquellen haben eine Reichweite
- Lautstärke von Soundquellen wird anhand der Entfernung zum Spieler festgelegt
- Regel: Wenn im Wald ein Baum umfällt und keiner ist da... dann gibt es *kein* Geräusch



Fertige Sound-Engine



Entwicklungszeit: ca. 2 Wochen

Fazit

Mit dem aktuellen Stand können wir bereits

- ein Level laden und anzeigen
- uns pseudo-physikalisch korrekt innerhalb dieses Levels bewegen
- Hintergrundmusik und Soundeffekte abspielen

Das reicht für ein erstes, einfaches Spiel!

Entwicklungszeit

Benötigte Entwicklungszeit mit einem 2-Mann-Team:

- Grafik-Engine (inkl. Loading): ca. 6 Monate
- Input: ca. 2 Tage
- Physiksystem: ca. 6 Monate
- Sound-Engine: ca. 2 Wochen

Insgesamt ca. 12 Monate, also 1 Jahr Entwicklungszeit!

Wichtige Merkregel

Meistens dauert die Entwicklung länger als geplant!

Entwicklungszeit

Benötigte Entwicklungszeit mit einem 2-Mann-Team:

- Grafik-Engine (inkl. Loading): ca. 6 Monate
- Input: ca. 2 Tage
- Physiksystem: ca. 6 Monate
- Sound-Engine: ca. 2 Wochen

Insgesamt ca. 12 Monate, also 1 Jahr Entwicklungszeit!

Wichtige Merkregel

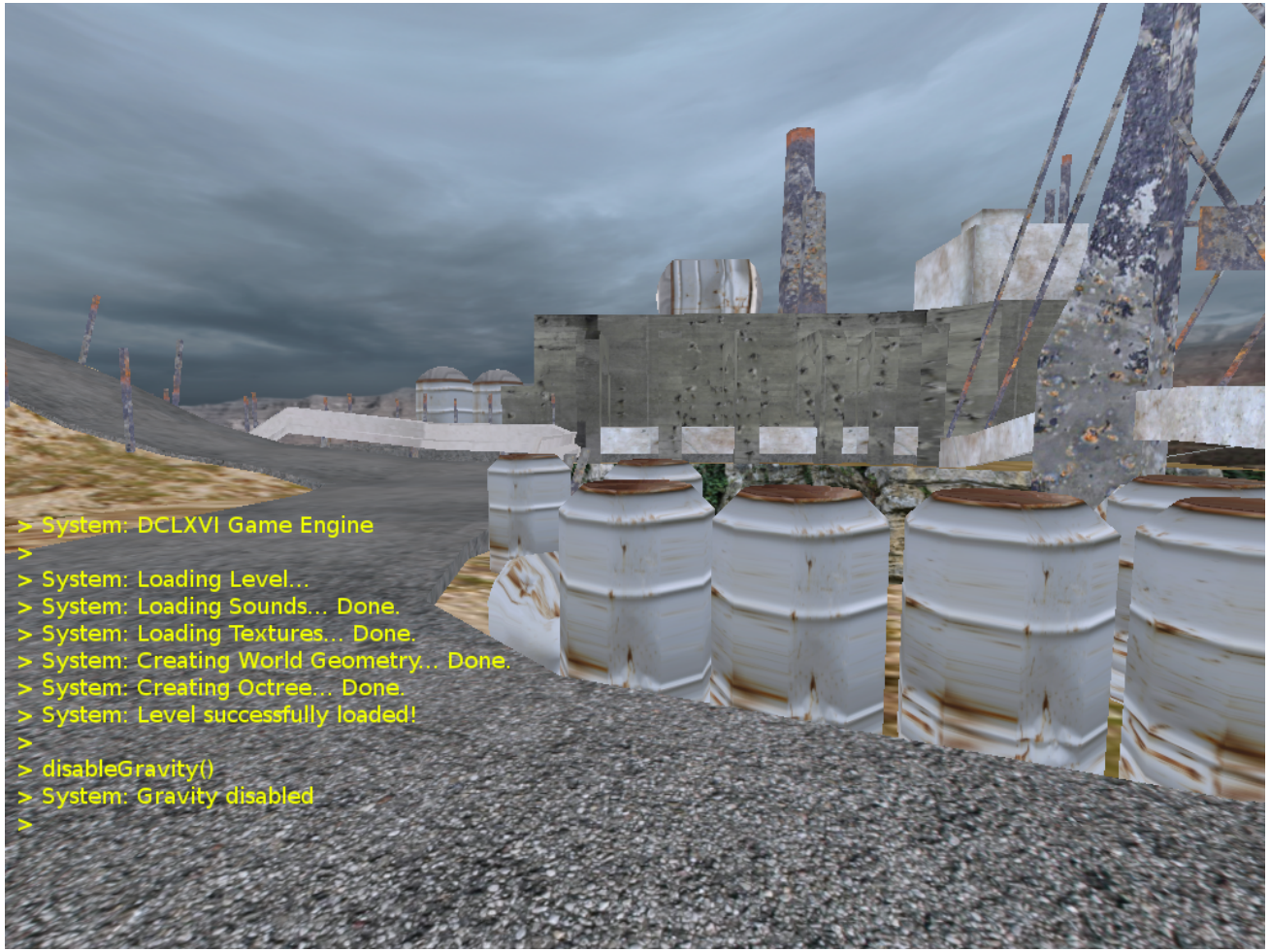
Meistens dauert die Entwicklung länger als geplant!

Mögliche Erweiterungen

Konsole

- Vorbild: „Quake“
- Hauptaufgaben:
  - Daten modifizieren, ohne das Spiel neu starten zu müssen
  - Ausgabe von Debug-Informationen zur Laufzeit

Eine einfache Konsole

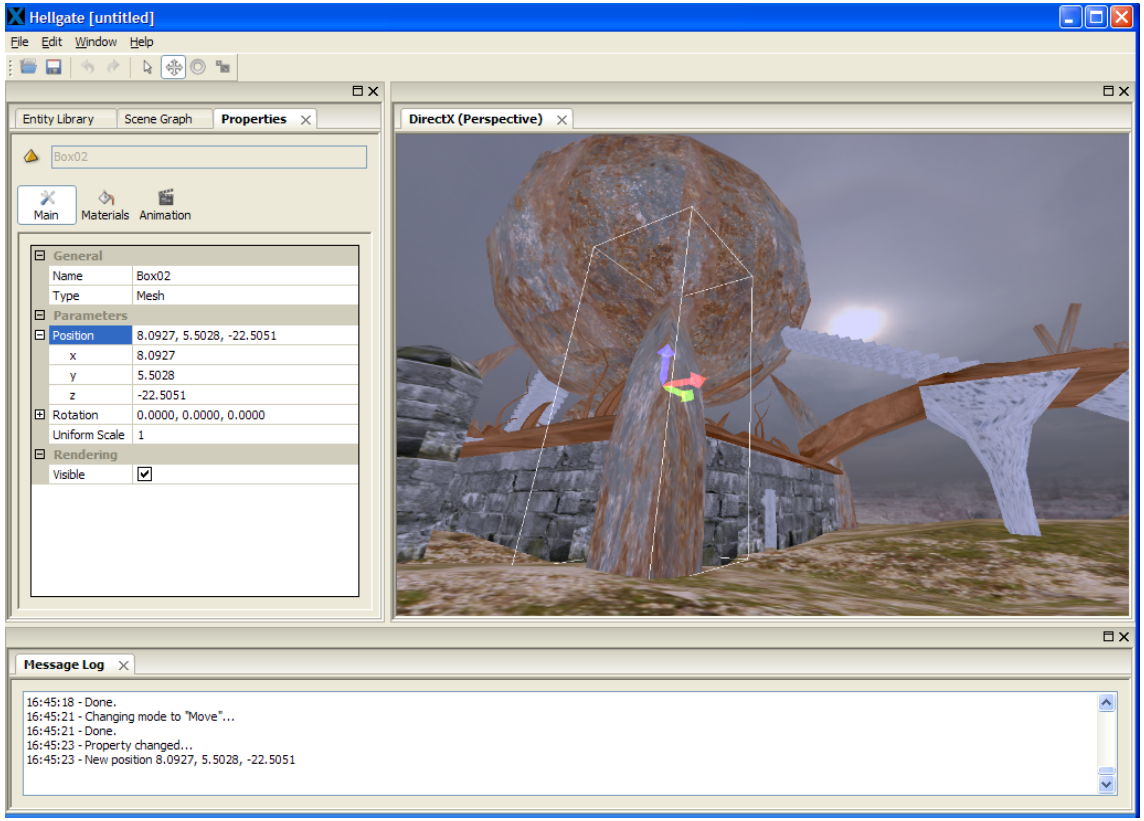


Entwicklungszeit: ca. 2 Wochen

Level- / Map-Editor

- fast jedes Spiel besitzt heutzutage einen Level- oder Map-Editor
- Vorteil für die Entwicklung:
  - Erstellung von Content ist nicht auf Designer beschränkt
- Vorteil für das fertige Spiel:
  - bessere Chance auf Langlebigkeit

Hellgate-Editor



Entwicklungszeit: $>$ 6 Monate

Sonstige Erweiterungen

- weitere mögliche Erweiterungen sind
  - Künstliche Intelligenz
  - Netzwerk
  - Persistenz
  - Partikelsysteme
  - Scripting
  - ...
- Trennung zwischen Engine und Spiel ist hierbei oft schwierig

Voraussetzungen

Was braucht man, um eine eigene Game Engine zu entwickeln?

Was braucht man, um eine eigene Game Engine zu entwickeln?

- Fähigkeiten
- Entwicklungswerkzeuge
- Team

Fähigkeiten

Benötigt werden

- gute Programmierkenntnisse (C / C++)
- grundlegendes mathematisches Wissen (insbesondere Lineare Algebra)
- gute Englisch-Kenntnisse

Hilfreich sind

- Basiswissen im Bereich Software-Engineering
- rudimentäre Kenntnisse der Physik (Kollisionsberechnung, Rigid-Bodies etc.)
- weiterführendes mathematisches Wissen (für fast alle Bereiche wichtig)
- Spezialwissen (künstliche Intelligenz, Netzwerktechnik etc.)

Fähigkeiten

Benötigt werden

- gute Programmierkenntnisse (C / C++)
- grundlegendes mathematisches Wissen (insbesondere Lineare Algebra)
- gute Englisch-Kenntnisse

Hilfreich sind

- Basiswissen im Bereich Software-Engineering
- rudimentäre Kenntnisse der Physik (Kollisionsberechnung, Rigid-Bodies etc.)
- weiterführendes mathematisches Wissen (für fast alle Bereiche wichtig)
- Spezialwissen (künstliche Intelligenz, Netzwerktechnik etc.)

Das nötige Werkzeug

Benötigt werden

- Programmierumgebung - Eclipse, Visual Studio
- Compiler - GCC, Microsoft C++ Compiler

Hilfreich sind

- Versionierungssystem - Subversion, CVS
- Bugtracker - Mantis, Bugzilla
- unzählige weitere Tools

Für zusätzlichen Content

- 2D-Grafik-Programme - Photoshop, GIMP
- 3D-Grafik-Programme - 3ds Max, Blender, Maya
- Sound-Programme - Cubase, Logic, Audacity

Das nötige Werkzeug

Benötigt werden

- Programmierumgebung - Eclipse, Visual Studio
- Compiler - GCC, Microsoft C++ Compiler

Hilfreich sind

- Versionierungssystem - Subversion, CVS
- Bugtracker - Mantis, Bugzilla
- unzählige weitere Tools

Für zusätzlichen Content

- 2D-Grafik-Programme - Photoshop, GIMP
- 3D-Grafik-Programme - 3ds Max, Blender, Maya
- Sound-Programme - Cubase, Logic, Audacity

Das nötige Werkzeug

Benötigt werden

- Programmierumgebung - Eclipse, Visual Studio
- Compiler - GCC, Microsoft C++ Compiler

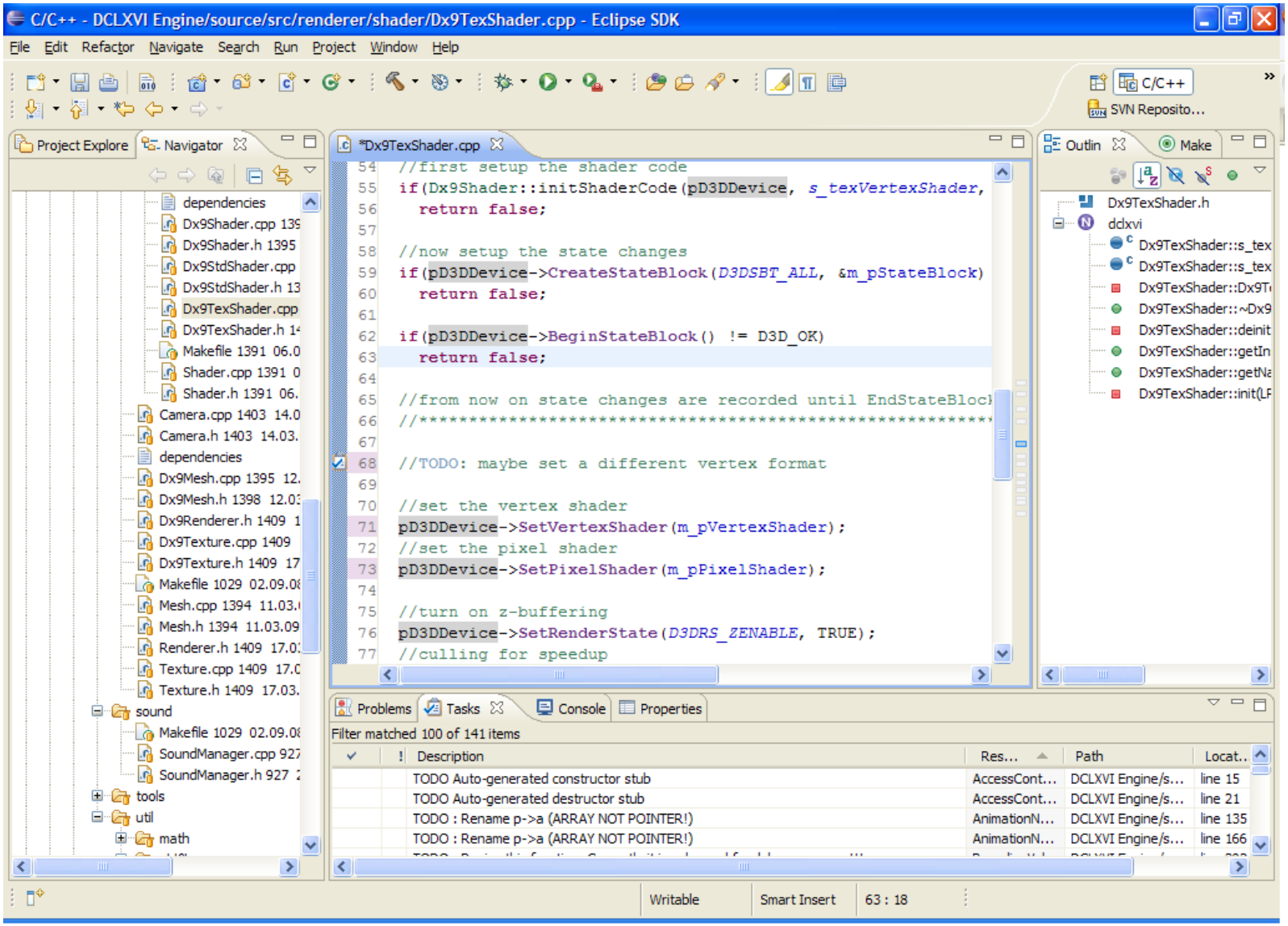
Hilfreich sind

- Versionierungssystem - Subversion, CVS
- Bugtracker - Mantis, Bugzilla
- unzählige weitere Tools

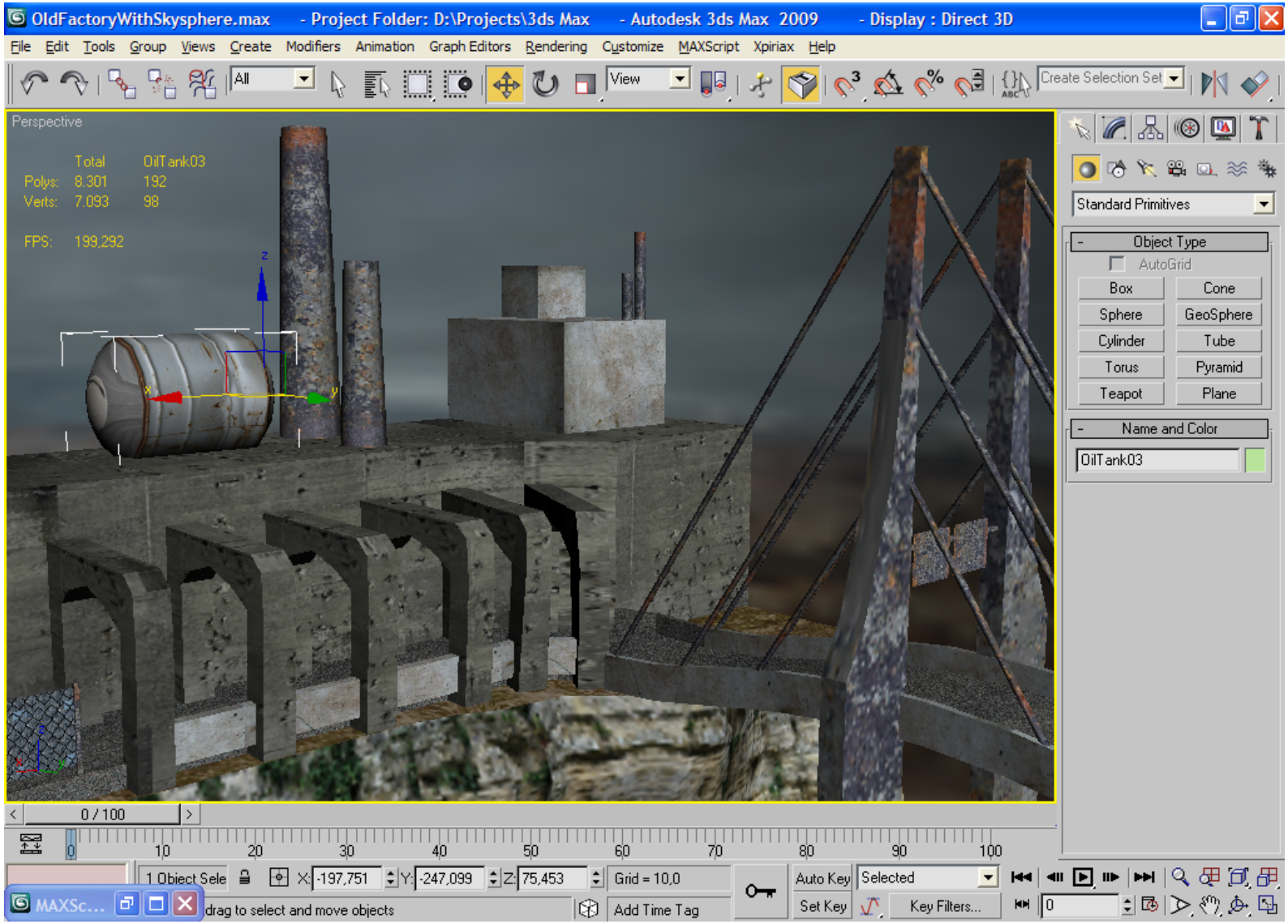
Für zusätzlichen Content

- 2D-Grafik-Programme - Photoshop, GIMP
- 3D-Grafik-Programme - 3ds Max, Blender, Maya
- Sound-Programme - Cubase, Logic, Audacity

Eclipse IDE


C/C++ - DCLXVI Engine/source/src/renderer/shader/Dx9TexShader.cpp - Eclipse SDK
File Edit Refactor Navigate Search Run Project Window Help
Project Explore
Navigator
*Dx9TexShader.cpp
Outlin
Make
Problems
Tasks
Console
Properties
Filter matched 100 of 141 items
Description
Res...
Path
Locat..
TODO Auto-generated constructor stub
TODO Auto-generated destructor stub
TODO : Rename p->a (ARRAY NOT POINTER!)
TODO : Rename p->a (ARRAY NOT POINTER!)
Writable
Smart Insert
63 : 18

3ds Max


OldFactoryWithSkysphere.max - Project Folder: D:\Projects\3ds Max - Autodesk 3ds Max 2009 - Display : Direct 3D
File Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Customize MAXScript Xpiriax Help
Perspective
Total OilTank03
Polys: 8.301 192
Verts: 7.093 98
FPS: 199,292
Standard Primitives
Object Type
AutoGrid
Box Cone
Sphere GeoSphere
Cylinder Tube
Torus Pyramid
Teapot Plane
Name and Color
OilTank03
0 / 100
1 Object Sele
X: -197,751 Y: -247,099 Z: 75,453
Grid = 10,0
Auto Key Selected
Set Key Key Filters...
drag to select and move objects
Add Time Tag

Team

Frage: Kann man eine Game Engine auch alleine entwickeln?

Antwort: Prinzipiell ja, aber...

- fehlende Fähigkeiten - niemand ist in allen Bereichen gut
- Zeit - die Entwicklung einer Game Engine ist sehr zeitaufwändig
- Motivation - es gibt viele demotivierende Phasen

Team

Frage: Kann man eine Game Engine auch alleine entwickeln?

Antwort: Prinzipiell ja, aber...

- fehlende Fähigkeiten - niemand ist in allen Bereichen gut
- Zeit - die Entwicklung einer Game Engine ist sehr zeitaufwändig
- Motivation - es gibt viele demotivierende Phasen

Wie sollte das Team aussehen?

Wichtige Aufgabenbereiche bei der Computerspieleentwicklung

- Programmierer
  - Grafikprogrammierung
  - Künstliche Intelligenz (KI)
  - Physik (z.B. Kollision, Schwerkraft)
  - ...
- Grafiker & Designer
  - 3D-Modelle
  - Texturen
  - Leveldesign
  - ...
- Sound-Artists
  - Musik
  - Soundeffekte

Wie sollte das Team aussehen?

Wichtige Aufgabenbereiche bei der Computerspieleentwicklung

- Programmierer
  - Grafikprogrammierung
  - Künstliche Intelligenz (KI)
  - Physik (z.B. Kollision, Schwerkraft)
  - ...
- Grafiker & Designer
  - 3D-Modelle
  - Texturen
  - Leveldesign
  - ...
- Sound-Artists
  - Musik
  - Soundeffekte

Wie sollte das Team aussehen?

Wichtige Aufgabenbereiche bei der Computerspieleentwicklung

- Programmierer
  - Grafikprogrammierung
  - Künstliche Intelligenz (KI)
  - Physik (z.B. Kollision, Schwerkraft)
  - ...
- Grafiker & Designer
  - 3D-Modelle
  - Texturen
  - Leveldesign
  - ...
- Sound-Artists
  - Musik
  - Soundeffekte

Fazit

Warum sollte man eine eigene Engine schreiben?

Warum sollte man eine eigene Engine schreiben?

Nachteile einer eigenen Engine:

- Entwicklungszeit - es braucht in der Regel mehrere Jahre, bis die Engine so weit ist, dass man damit das erste Spiel entwickeln kann
- Anforderungen - es wird erheblich mehr Know-How vorausgesetzt, als bei der Verwendung einer fertigen Engine
- Motivation - am Anfang eine lange Durststrecke, in der man nicht viel sieht

Warum sollte man eine eigene Engine schreiben?

Nachteile einer eigenen Engine:

- Entwicklungszeit - es braucht in der Regel mehrere Jahre, bis die Engine so weit ist, dass man damit das erste Spiel entwickeln kann
- Anforderungen - es wird erheblich mehr Know-How vorausgesetzt, als bei der Verwendung einer fertigen Engine
- Motivation - am Anfang eine lange Durststrecke, in der man nicht viel sieht

Warum sollte man eine eigene Engine schreiben?

Nachteile einer eigenen Engine:

- Entwicklungszeit - es braucht in der Regel mehrere Jahre, bis die Engine so weit ist, dass man damit das erste Spiel entwickeln kann
- Anforderungen - es wird erheblich mehr Know-How vorausgesetzt, als bei der Verwendung einer fertigen Engine
- Motivation - am Anfang eine lange Durststrecke, in der man nicht viel sieht

Warum sollte man eine eigene Engine schreiben?

Vorteile einer eigenen Engine:

- Kompatibilität - man kann die Engine genau an seine Bedürfnisse bzw. an die Anforderungen des zu entwickelnden Spiels anpassen
- Flexibilität - man kann während der Entwicklung des Spiels sehr einfach Änderungen an der Engine vornehmen
- Lizenzgebühren - man muss keine Lizenzgebühren für eine fremde Engine zahlen und kann gleichzeitig versuchen, Lizenzen für seine eigene Engine zu verkaufen (bestes Beispiel: id-Software)
- Verständnis - man erhält einen Blick hinter die Kulissen anstatt eine Engine einfach nur als BlackBox zu verwenden

Warum sollte man eine eigene Engine schreiben?

Vorteile einer eigenen Engine:

- Kompatibilität - man kann die Engine genau an seine Bedürfnisse bzw. an die Anforderungen des zu entwickelnden Spiels anpassen
- Flexibilität - man kann während der Entwicklung des Spiels sehr einfach Änderungen an der Engine vornehmen
- Lizenzgebühren - man muss keine Lizenzgebühren für eine fremde Engine zahlen und kann gleichzeitig versuchen, Lizenzen für seine eigene Engine zu verkaufen (bestes Beispiel: id-Software)
- Verständnis - man erhält einen Blick hinter die Kulissen anstatt eine Engine einfach nur als BlackBox zu verwenden

Warum sollte man eine eigene Engine schreiben?

Vorteile einer eigenen Engine:

- Kompatibilität - man kann die Engine genau an seine Bedürfnisse bzw. an die Anforderungen des zu entwickelnden Spiels anpassen
- Flexibilität - man kann während der Entwicklung des Spiels sehr einfach Änderungen an der Engine vornehmen
- Lizenzgebühren - man muss keine Lizenzgebühren für eine fremde Engine zahlen und kann gleichzeitig versuchen, Lizenzen für seine eigene Engine zu verkaufen (bestes Beispiel: id-Software)
- Verständnis - man erhält einen Blick hinter die Kulissen anstatt eine Engine einfach nur als BlackBox zu verwenden

Warum sollte man eine eigene Engine schreiben?

Vorteile einer eigenen Engine:

- Kompatibilität - man kann die Engine genau an seine Bedürfnisse bzw. an die Anforderungen des zu entwickelnden Spiels anpassen
- Flexibilität - man kann während der Entwicklung des Spiels sehr einfach Änderungen an der Engine vornehmen
- Lizenzgebühren - man muss keine Lizenzgebühren für eine fremde Engine zahlen und kann gleichzeitig versuchen, Lizenzen für seine eigene Engine zu verkaufen (bestes Beispiel: id-Software)
- Verständnis - man erhält einen Blick hinter die Kulissen anstatt eine Engine einfach nur als BlackBox zu verwenden

Ende

Vielen Dank für die Aufmerksamkeit!