Computerspiele

Thomas Jung

t.jung@fhtw-berlin.de

Heute

- Historische Entwicklung
  - Demo C64
- Überblick über 3D-Spiele
  - DOOM-Rendering-Engine
  - Demos Wolfenstein3D, DOOM, Descent II
- Crystal Space

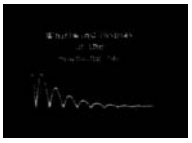
Motivation

- 3D-Spiele bilden großen Markt
- Spiele-Engines können auch für „ernsthafte" Anwendungen genutzt werden
- 3D-Spiele nutzen neueste Technologien
- Chrystal Space ist frei verfügbar

Beginn der Computerspiele

- Willy Higginbotham, Brookhaven National Laboratory
- Mitarbeiter im Atomprojekt der USA
- 1958: Oszilloskop mit regelbarem Abflugwinkel, Auslösung des Flugs per Knopf
- anlässlich der Tage der offenen Tür 1958 und 1959 im B.N.L.

MIT

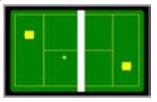


- Whirlwind Computer, 1949
  - graphische Ausgabe
  - erste Computeranimation: springender Ball

- DEC - PDP 1, 1961
  - Steve Russell entwickelt Starwar
  - zwei Schiffe mit begrenztem Treibstoff können sich beschießen
  - Programm wird mit PDP 1 (120.000$) ausgeliefert

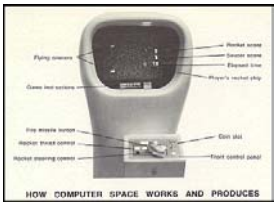
Odissey

- Ralph Baer, Sander Associates
  - Entwicklung seit 1966, Patent 1968
  - Motion Circuit für die Darstellung farbiger Balken
  - Mehrere Spiele (Ping pong, Volleyball, ...)
- Magnavox
  - verkauft kommerzielle Version seit 1972
  - 100 $, SW, Farbfolien für Fernseher

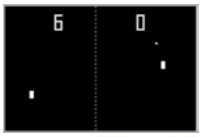


Computer Space

- Nolan Bushnell, 1971
- kommerzeielle Umsetzung von Spacewar für Firma Nuttin Associates
- Mißerfolg, zu kompliziert für Spielsalonbenutzer



Pong



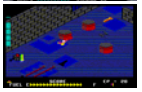
- 1972 gründet Bushnell Atari
- Entwicklung von Pong
- Name inspiriert durch Sound
- Kein Interessent zur Vermarktung
- Prototyp in Kneipe sehr erfolgreich
- Atari produziert Spieleautomaten selbst und verkauft seit 1972 8500 Geräte pro Jahr
- 1975 Heimversion „Sears Telegames"
- 1976 40 Mio $ jährlich, Übernahme durch Warner

Meilensteine

- Automaten seit Mitte der 70iger Jahre weit verbreitet
  - Breakout 1976
- Cinematronics baut seit 1977 Vektordisplays
  - Speed Freak, 1977
- Pacman 1980
  - friedliches Computerspiel
- Donkey Kong 1981

Commodore C64

- Markteinführung 1982
- In zwei Jahren 4 Mio. verkauft
- Rastergrafik: 320 x 200 Pixel, 16 Farben
- 64 KB RAM, Soundchip
- Später:
  - Atari ST: 1984
  - Commodore Amiga: 1985 →



3D - Spiele

- Merkmale
  - Subjektive Perspektive
  - Textur
- Wolfenstein 3D 1992
- Ultima Underworld 1993
- Doom 1993
- Descent II 1994

Ultima Underworld

Wolfenstein 3D

(in Deutschland verboten!)

- Id Software, 1992
- inspiriert durch 2D-Spiel für C64
- Texturierte Flächengrafik mit subjektiver Perspektive
- Statische vertikale Wände
- Dynamische Objekte (animierte Texturen) als Billboards
- Kamerarotationen, vor- und zurück bei fester Kamerahöhe
- Darstellung (optimiert für Intel 386)
  - Zellen, „Raycasting Refresh Architektur": Auswahl vertikaler Scanlines, Skalierung auf geeignete Größe anhand des z-Werts

Wolfenstein, C64

Wolfenstein3D-Demo

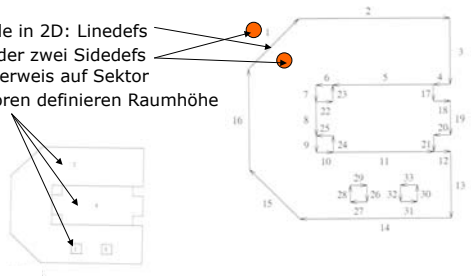
DOOM, 1993

(in Deutschland verboten!)

- Id Software 1993
- Neue Rendering Engine
  - Texturierung von Boden und Decke
  - Leichte Auf- und Abwärtsbewegungen der Kamera
- Sound

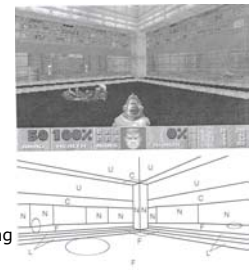
DOOM-Szenenbeschreibung

- Wände in 2D: Linedefs
- Ein oder zwei Sidedefs mit Verweis auf Sektor
- Sektoren definieren Raumhöhe



DOOM - Texturen

- SideDef beschreibt obere (U), untere (L) und mittlere (N) Textur inkl. jeweiliger Texturkoordinaten
- Nur mittlere Texturen transparent
- Decken und obere Textur in oberer, Boden und untere Textur in unterer Bildhälfte
- Affines Texturmapping für (vert.) Scanlines mit konstantem z-Wert
- Spez. Texturierung für Boden/Decke
- Bildaufbau spaltenweise, Depthculling per Spalte



Doom - Demo

Descent II

- Parallax Software, 1994
- Komplette 3D-Engine
  - Freie Kameraperspektive
  - Texturierung von schrägen Flächen
  - 3D-Objekte möglich
- Multiple User (wie bei DOOM)
- Integration von VR-Helm I-Glasses

Stand der Technik

Spielekonsolen mit Environment mapping

Psychologische Auswirkungen

- Sind umstritten
- Studie der Ruhr-Uni Bochum:
  - Ballerspiele können Gewaltbereitschaft von Jugendlichen steigern, insbesondere Empathie (Mitleid) senken
- These amerikanischer Militärpsychologen:
  - Mit dem Spielen werden Tötungsfähigkeiten trainiert

Zusammenfassung

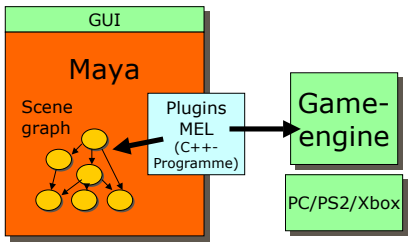
- Computerspiele seit den 60iger Jahren
- Kommerziell verfügbar seit den 70igern
- C64 1982, Wolfenstein 3D 1992
- Doom Engine noch mit eingeschränkten Kameratransformationen
- Descent II erste 6DOF-Engine 1994

Spieleentwicklung

- Klassischer Ansatz
  - Game-Engine
  - Level-Editor
  - Beispiel: Chrystall Space
- ~~Blender~~ (eingestellt März 2002)
- Maya

3D-Modellierer und Spiele

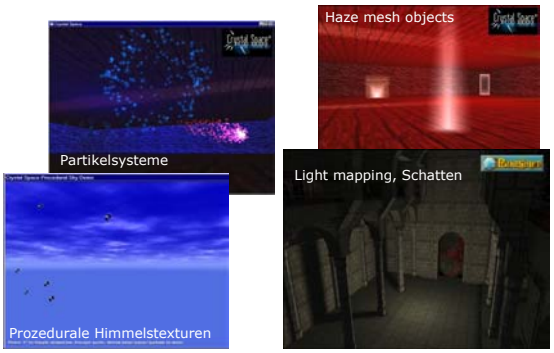
- Professionelle Modellierwerkzeuge bieten Plugin-Schnittstelle an
  - Beispiel Maya und MEL



Chrystal Space

- Begründet von Jorrit Tyberghein
- http://chrystalspace.linuxgames.com
- 6DOF-Spiele Engine
- Geschrieben in C++
- Vertrieben unter der GNU General Public License
  - Quellcode verfügbar
- Zur Zeit Beta-Status
- Für die die meisten Plattformen verfügbar
  - Win32, DOS, Linux, MacOS, BeOS, OS/2, Unix
  - Direct3D und OpenGL

Screen shots



Rendering - Funktionalitäten

- Perspektivisch korrektes Texture mapping
  - Mipmapping
  - Texturtransformationen
  - Prozedurale Texturen (Plasma, Himmel, Wasser, Feuer)
  - Transparente Texturen
- Volumetrischer Nebel
  - Halo-Effekte um Lichtquellen
- Schatten
  - Vorberechnung
  - Radiosity (noch nicht stabil ?!)
- Dynamische farbige Beleuchtung

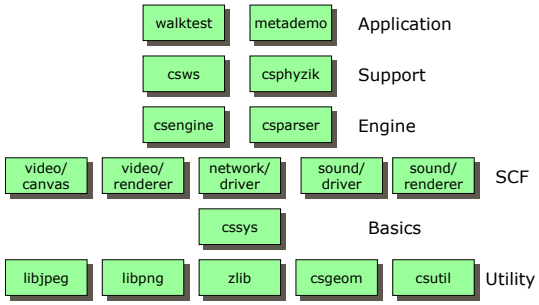
Szene - Funktionalitäten

- Landscape Engine integriert
- Moving Objects (durch Skripte steuerbar)
- Dreiecksnetze
  - Progressive Meshes
  - Dynamische Tesselierung von Bezier-Flächen
  - LOD
  - Skelett-basiert
- Partikelsysteme
- Verdeckung
  - Portale, Coverage Buffer, BSP-Trees, Octree
- Mehrere Kameras (3.Person-Modus)

Weitere Funktionalitäten

- System Driver
  - Einbindung von DLLs (.so) über abstrakte Basisklassen
  - Shared Class Facilities (SCF)
- Sound Driver
- Network Driver
- Unabhängige Geometriebibliothek
  - Vektor, Matrix, Polygon, Ebene, Quaternionen
- Scripting (Python-basiert)
- Virtuelles File System
  - Konfigurationsdateien
  - Unterstützung diverser Bildformate, Zip-Kompression

Softwarearchitektur



Welt besteht aus

- ***Sectors und Portals***
- *Mesh Objects* und *Mesh Object Factories*
  - Factories erzeugen Mesh Objects, vererben Eigenschaften
- *Light Sources*
- *Collection Objects*
  - Gruppierung von Objekten
- *Key/value Pairs*
- *Nodes*
  - Punkte im 3D-Raum

Beleuchtung

- Static
  - Nicht änderbar
  - Berechnung der Lightmaps
    - Sichtvolumen von der Lichtquelle aus
    - Berücksichtigung von Verdeckungen (Schatten)
    - Aktualisieren der Pixelintensität anhand des Abstands
- Pseudo-dynamic
  - Nur Farbe und Helligkeit änderbar
    - Mehrere Lightmaps pro Polygon werden kombiniert
- Dynamic
  - Auch Position änderbar

Transformationen

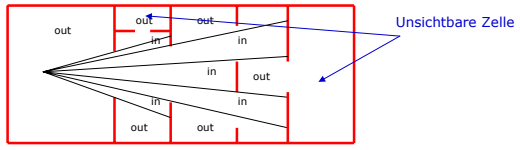
- 3x3-Kameratransformationen
  - Perspektive separat
- Warping-Transformationen für Portale
- Texturtransformationen
  - Animierbar für *Dynamic Textures*

Objekte

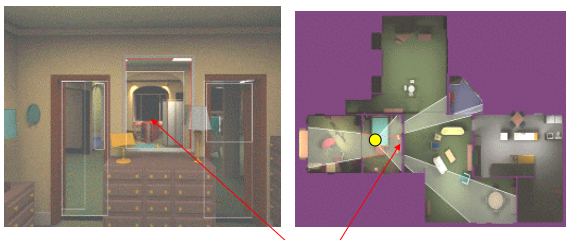
- 3D-Sprites
  - Dreieckslisten
- 2D-Sprites
  - Billboard - Polygon
- Things
  - Polygonlisten zur Definition von Wänden, Möbeln etc.
- Curved Surfaces
  - Teile von Things
- Terrain
  - muß nicht in Sektor sein

Bestimmung sichtbarer Bereiche

- Szene wird in Zellen aufgeteilt
- Zellen sind über Portale verbunden
- Nur Objekte in sichtbaren Zellen werden angezeigt
- In Gebäuden sind meist nur wenige Zellen sichtbar



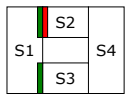
Portale: Beispiel



Portale in Chrystal Space

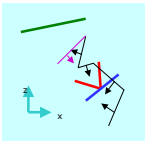
- Die Welt ist eingeteilt in *Sectors*
- *Sectors* enthalten Objekte
- Objekte bestehen aus *Polygons*
- *Things* sind Objekte die *Portals* enthalten können
- Jedes *Polygon* eines Things kann ein *Portal* sein
- Portale repräsentieren Öffnungen oder Spiegel
- Spiegel definieren Transformationen

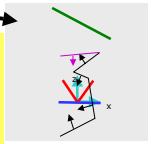
S1 besteht aus 6 normalen Polygonen und zwei Portalen

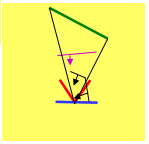


S2 besteht aus 6 Polygonen davon ist eines ein Portal

Rendering von Portalen

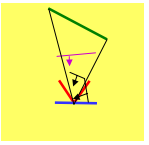




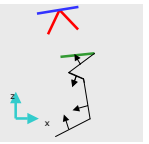


Rendere Welt (Kamera, Viewpolygon)
- Transformiere lokale *Mesh Objects* in Kameraraum
- Für alle Polygone aller *Mesh Objects*
  - Perspektivische Korrektur (/z)
  - Wenn nicht alle Ecken hinter Viewplane
    - Backface Culling
    - Wenn nicht alle Ecken vor Viewplane
      - Klippen des Polygons gegen Viewplane
    - Transformiere Texturtransformationsmatrix in Kameraraum
    - Transformiere Flächennormale in Kameraraum
    - Klippen des Polygons gegen das Viewpolygon
    - Wenn Polygon Portal ist
      - **Rendere Welt(Kamera, Polygon)**
    - Sonst
      - Rendere Polygon

Spiegelportal



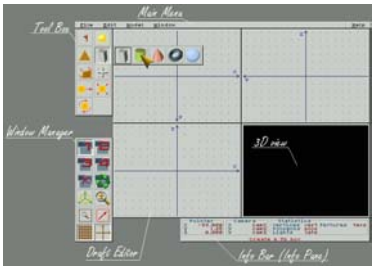
Kamera am Portal spiegeln



Rendere Welt (Kamera, Viewpolygon
…
- Wenn Polygon Portal is
  - **Rendere Welt(Kamera, Polygon)**
- Sonst
  - Rendere Polygon

Chrystal Space - MazeD

- Level Editor
  - Für alle Plattformen
  - Quellkode
  - Beta-Version
- Soll *mirrors* und *portals* unterstützen
  - Konverter für Quake-Level tun das nicht
- Quark++ wird Chrystal space unterstützen



Zusammenfassung

- Spielentwicklung mit 3D-Modellierer und speziellem Plugin
- Chrystall Space als Beispiel für heutige Spiele-Engines
- Level Editor MazeD oder Quark++
- Bestimmung sichtbarer Bereiche durch Portale