http://canyonshooter.tu-bs.de

Teamprojekt
WS07/08

CANYON SHOOTER

http://canyonshooter.tu-bs.de

Vorwort

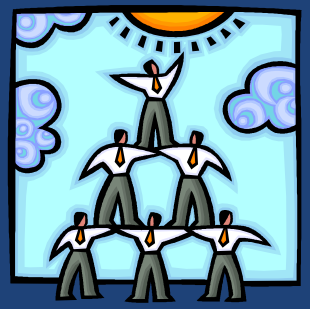
- Die Präsentation ist in 3 wesentliche Bestandteile aufgeteilt
  - Gesamtübersicht des Projekts
  - Präsentation der einzelnen Module
  - Ausblick
- Die Präsentation dauert ca. 2 Stunden

Das Projekt im Überblick

EINLEITUNG

Aktueller Stand der Entwicklung

Das Team



Florian Mätschke
Teamleiter

Christian Woizischke
Architectural Project Manager

Manuel Rodriguez
Multimedia Content Manager

Markus Lorenz

Malte Mauritz

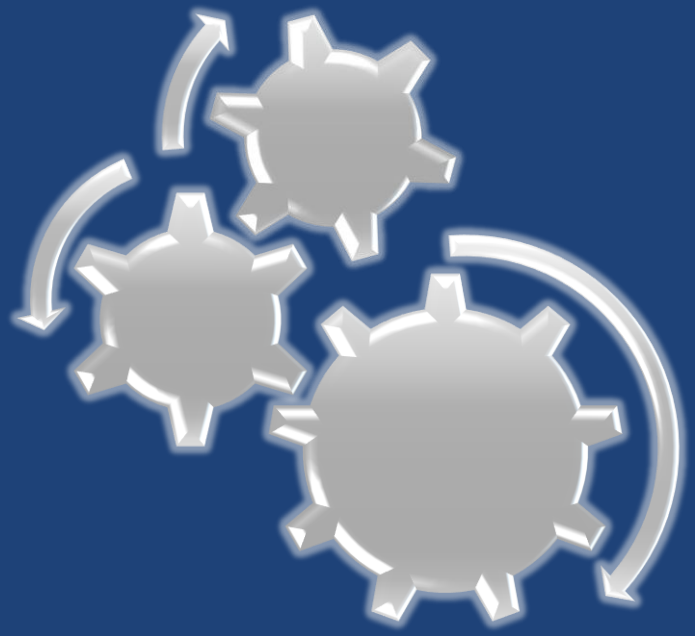
Martin Fiebig

Thorben Schulze

Sascha Lity

Richard Wolfer

Danny Melching



Präsentation der einzelnen Module

Gliederung der Module

- Grafik
  - Grafik-Engine,
  - Partikel-Effekte + Live-Demo
- Physik
- Gameplay
  - Story
  - Player
  - Enemies & AI State Machine
  - Weapon & Items
- Canyon-Generierung & Level Editor
- User Interface
  - Menu, Profile, Highscore
  - HUD
  - Console
- Sound & Content
  - Sound System
  - Sounds
  - 3D Modelle + Animationsprototypen
  - Texturen und Grafiken
  - Website

Christian Woizischke, Florian Mätschke

GRAFIK

Christian Woizischke

Grafik-Engine

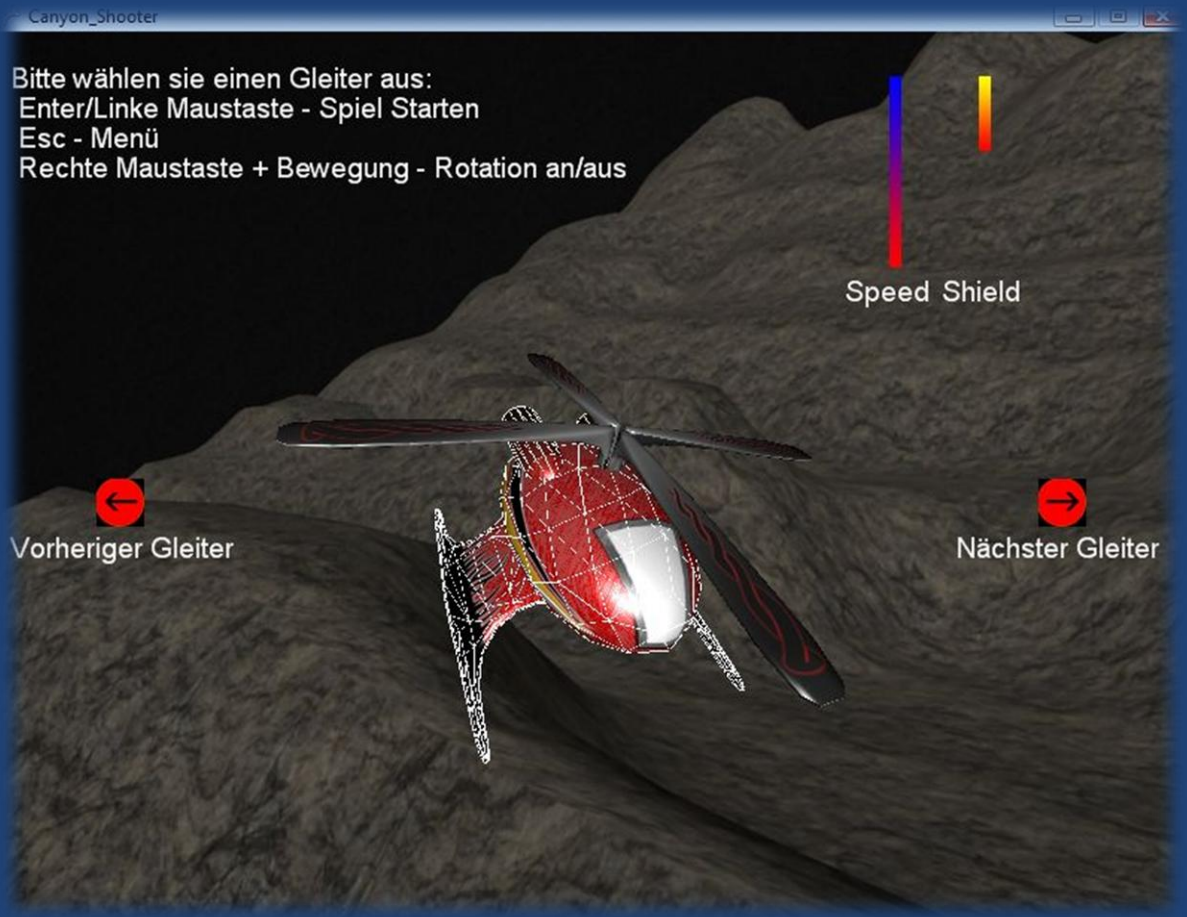
Grafik: Kameras

- Perspektivische Kamera hinter dem Spieler
  - Veränderbares Field-of-View (Öffnungswinkel)
    - Geschwindigkeitsgefühl
- Freie perspektivische Kamera zu Debugging-Zwecken
  - Kann im Spiel mit der Taste „C“ ausgewählt werden.

Grafik: Modelle

- Modelle
  - Statisch (Aus FBX-Dateien oder dynamisch generierte)
  - Animation: Rotation (pro Modellteil)
    - Beispiel: Rotor vom Helikopter
  - Innerhalb eines Modells verschiedenes Aussehen
    - Modellteile können verschiedene Texturen, Beleuchtung, etc. haben.
  - Einfache Beschreibung in XML-Dateien für den Entwicklungsprozess
- Oberflächen
  - Farbe / Textur
  - Beleuchtet, Selbstleuchtend, Schatten
  - Glänzend / Matt
  - Drahtgittermodell
    - Debugging
  - Einfache Beschreibung in XML-Dateien für den Entwicklungsprozess

Grafik: Material pro Mesh

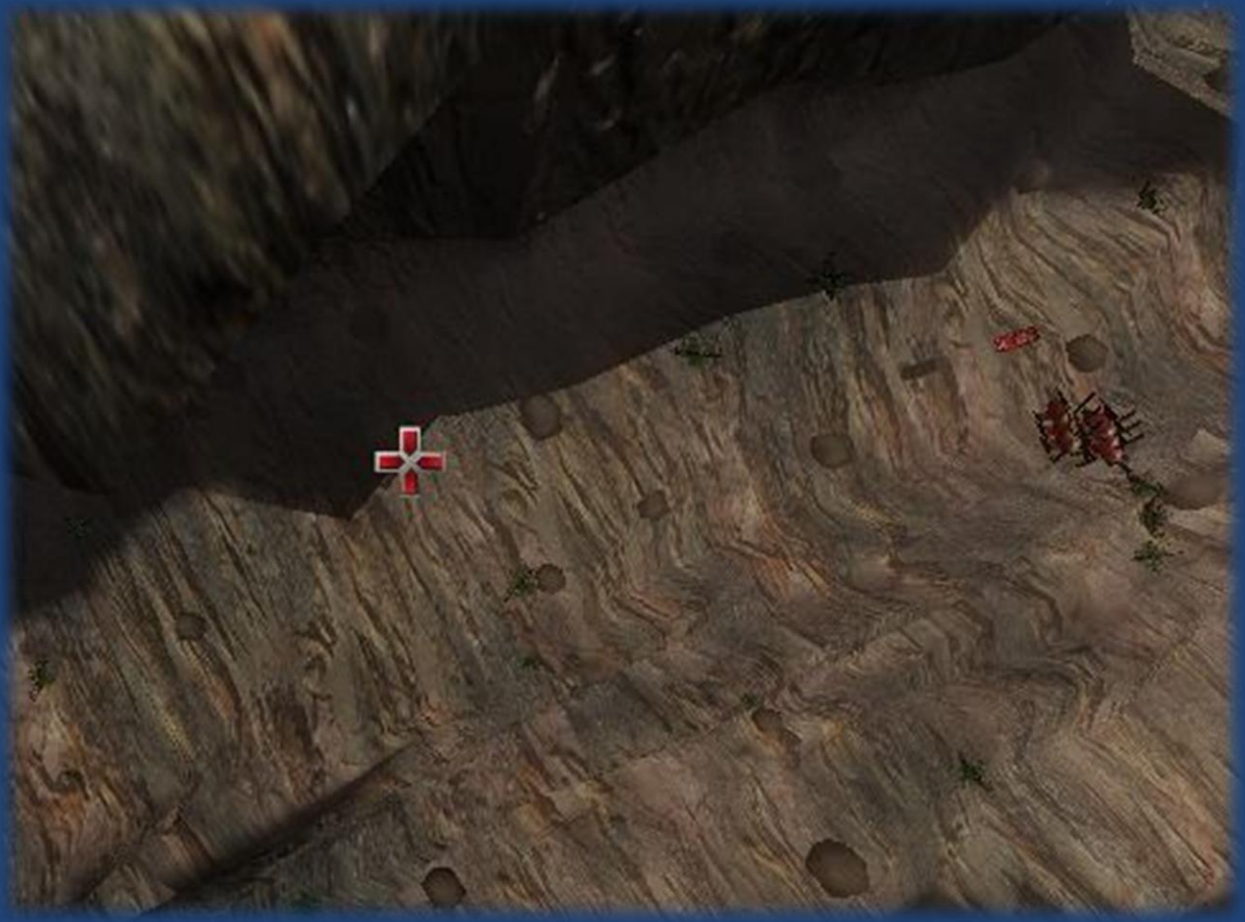


Grafik: Licht und Schatten

- Sonnenlicht
  - Parallele Strahlen
  - Schatten:
    - Shadow Mapping
    - Low-Detail für große Entfernungen
    - High-Detail in der Nähe des Spielers
    - Weicher Übergang zwischen den Detailstufen
    - Benötigt Shader Model 3.0
- Bis zu 5 Punktlichtquellen
  - Strahlen in alle Richtungen bis zu einer bestimmten Distanz
  - Keine Schatten
- Ambientes Licht
- Beleuchtungsmodell: Blinn-Phong
- Normal-Mapping
  - Benötigt Shader Model 3.0

Grafik: Lichtquellen

Grafik: Shadow Maps



Grafik: Shadow Mapping

Grafik: Normal Mapping

Grafik: Normal Mapping

Grafik: Post-Processing

- Bloom-Filter über das gesamte Bild
  - Aus dem Beispiel von XNA
  - „Weiches Bild“
  - Mehrere Stärken möglich
    - Zur Zeit eher dezent

Grafik: Dekoration

- Statische Objekte ohne Kollision
  - Steine, Grasbüschel auf dem Boden des Canyons
  - Zufällige Rotation
  - Automatische Positionierung auf dem Canyon
    - Optional: Nur auf Flächen mit einer bestimmten Normalen
      - z.B. auf dem Boden (Normale: 0,1,0)
  - Instancing-Verfahren → Große Anzahl an Objekten möglich
    - Ein Aufruf an Direct3D zeichnet mehrere Instanzen eines Objekts

Grafik: Skybox und Statics

Grafik: Nebel und Statics

Grafik: Nebel

Grafik: Hardwarevoraussetzungen

- Einige Features benötigen Shader-Model 3.0
  - Schatten
  - Nebel
  - Fragment Light (Beleuchtung auf Pixel-Ebene)
  - Normal Mapping
- Aber: Unterstützung von Shader-Model 2.0
  - Weniger Details
    - z.B.: Vertex Light

Grafik: Einstellungen

- Auflösung
- Bildwiederholrate
- Vertikale Synchronisierung
- Multi-Sampling

- Zur Zeit nur im Quellcode auswählbar

Florian Mätschke

Partikel-Effekte

Partikel-Effekte

- 6 verschiedene Partikel-Effekte zur Auswahl:
  - Rauch, Raketenqualm, Explosionen, …
- Effekte können an Objekte angehängt werden.
- Jeder Effekt kann einzeln getimed werden.

Partikel-Effekte

- Partikel-Systeme
  - konfigurier- und erweiterbar per XML-Definition
    - Wind, Gravitation, Größe,
    - Wachstum, Rotation, Partikel-Sprite, …
  - werden native auf der Grafikkarte als Shader ausgeführt

Partikel Effekt-Definition

explosion.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<XnaContent>
  <Asset Type="DescriptionLibs.ParticleEffect.ParticleEffectDescription">
      <!--Specifies the EffectType to use and the assigned class.-->
      <EffectType>EXPLOSION</EffectType>
      <Settings>
          <TextureName>Content/Textures/Particles/explosion3</TextureName>
          <MaxParticles>30</MaxParticles>
          <!-- Particle's Duration in Seconds -->
          <Duration>2</Duration>
          <DurationRandomness>1</DurationRandomness>
          <EmitterVelocitySensitivity>1</EmitterVelocitySensitivity>
          <MinHorizontalVelocity>3.5</MinHorizontalVelocity>
          <MaxHorizontalVelocity>4</MaxHorizontalVelocity>
          <MinVerticalVelocity>-5</MinVerticalVelocity>
          <MaxVerticalVelocity>5</MaxVerticalVelocity>

          <Gravity>0 0 -3</Gravity>
          <EndVelocity>6</EndVelocity>
          <MinColor>FFFFFF66</MinColor>
          <MaxColor>FFFFFFFF</MaxColor>
          <MinRotateSpeed>-1</MinRotateSpeed>
          <MaxRotateSpeed>1</MaxRotateSpeed>
          <MinStartSize>10</MinStartSize>
          <MaxStartSize>20</MaxStartSize>
          <MinEndSize>50</MinEndSize>
          <MaxEndSize>100</MaxEndSize>
          <SourceBlend>SourceAlpha</SourceBlend>
          <DestinationBlend>InverseSourceAlpha</DestinationBlend>
      </Settings>

  </Asset>
</XnaContent>
```

Live Demo

Partikel Effekte

Christian Woizischke, Florian Mätschke

PHYSIK

Physik

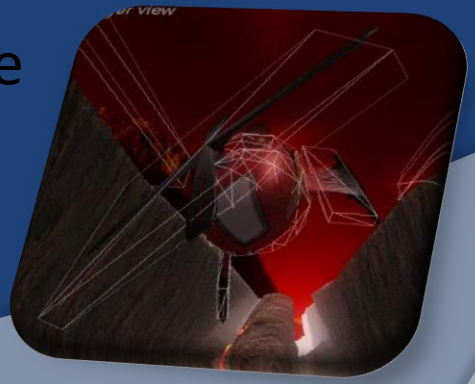
- Open-Source Physik-Engine XPA von XNADev.ru
  - Wrapper zur nativen Open-Dynamics-Engine (ODE)
  - Keine eingebaute Multi-Threading Unterstützung
  - Bugs
- Wrapper-Klassen als Schnittstelle zwischen CS und XPA
  - Multi-Threading auf Multi-Core CPUs
- Fast jedes bewegliche Objekt nutzt die Physik-Engine
  - oder ist in einer Hierarchie mit einem Physikobjekt.
    - Beispiel: Rakete ist ein Physikobjekt
    - Licht an der Rakete ist kein Physikobjekt

Physik

- Statische Objekte mit Kollisionsmöglichkeit
  - Bestandteil des Levels
  - Jedes 3D-Modell ist nutzbar
- Wrackteile bei Zerstörung von Objekten
  - Wrackteile können sich nach zufälliger Zeit selbst zerstören.
  - Rekursiv
  - Jedes 3D-Modell ist nutzbar
  - Zur Zeit werden noch Test-Modelle (Steine) verwendet.
- Druckwelle bei Explosionen
  - Zur Zeit nur im Single-Thread Modus

Physik

- Geröll
  - Beim Einschlag einer Rakete in den Canyon
  - Kollision mit dem Spieler, den Gegnern und den Projektilen
  - Mehrere Emitter-Typen
    - Erzeugung innerhalb eines Volumens (Kugel oder OOBB)
    - Erzeugung an einem Punkt
      - Geschwindigkeit in eine Richtung
      - Geschwindigkeit in eine zufällige Richtung innerhalb eines Kegels oder einer Kugel
- Visualisieren der Kollisionsgeometrie
  - Drahtgittermodell
  - Debugging



Richard Wolfer, Florian Mätschke,
Sascha Lity, Markus Lorenz, Christian Woizischke

GAMEPLAY

Sascha Lity

Story

Story

- 3 Story-Vorschläge
- Der Dritte wurde gewählt
- Das Rennen:

”Jedes Jahr findet das große Canyonballrennen auf der Erde statt. Es hat schon vor 100 Jahren, als die Gleiter entwickelt wurden, das illegale Straßenrennen Canonball abgelöst und genießt große Beliebtheit bei der Bevölkerung. Die Piloten kommen von überall um den besten Canyonfighter zu ermitteln. Es gilt ”gegnerische“ ferngesteuerte Gleiter abzuschießen, während man mit irrem Tempo durch den Canyon jagt und irre Manöver fliegen muss, um nicht an den Wänden zu zerschellen.

Du bist einer dieser Piloten. . .

Stell dein Können unter Beweis und hol dir den Titel!!!! “

Richard Wolfer, Manuel Rodriguez,
Florian Mätschke, Christian Woizischke

Player

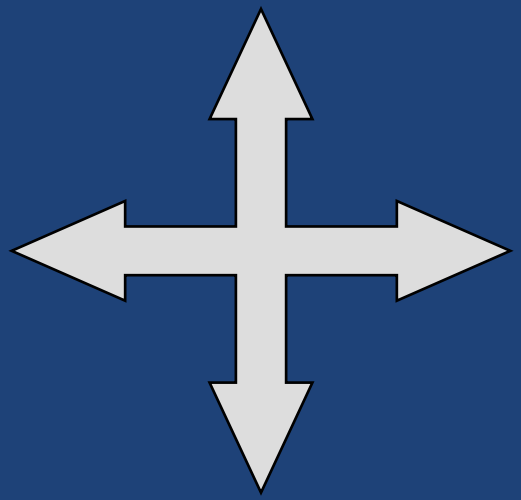
Player

Aus Krankheitsgründen von Richard Wolfer fehlt dieser Teil der Präsentation!

Player2

- Third-Person Perspektive
- Camera und Model banking
- Pan / Drift

Player2 Steuerung



Globale Rotation

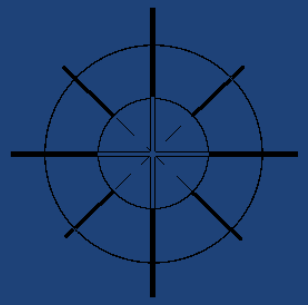
Live Demo

Player2

Florian Mätschke

Enemies & AI-State Machine

Enemies



- Jeder Enemy-Typ ist per XML konfigurierbar
  - Model, Hitpoints, Geschwindigkeit, Bewaffnung,….
- Es können Enemy-Formations gebildet werden.
  - Enemies fliegen in Gruppen.
- Intelligente Enemies mit verschiedenen Enemy-AIs

Murmel3.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<XnaContent>

    <Asset Type="DescriptionLibs.EnemyType.EnemyTypeDescription">

        <Model>Drone</Model>
        <MaxHitpoints>100</MaxHitpoints>
        <Speed>250.0</Speed>
        <Path>0 0 0</Path>
        <Weapons>
            <Item>ULTRA_PHASER</Item>
        </Weapons>
        <Formation>-60 0 0 60 0 0 0 90 -60 </Formation>
    </Asset>

</XnaContent>
```

AI-State Machine

- Die Canyon Shooter Engine verfügt über eine eigene AI-State Machine.
- Sie bietet eine hohe Flexibilität und sorgt gleichzeitig für einen wartbaren AI-Programmcode.
- Verwendung z.B. für Enemies, Waffen, Special-Effects, … möglich

AI-State Machine

- Jeder AIState ist als Klasse realisiert und verfügt über eigene Events wie:
  - **OnInit()** – Einmaliger Aufruf beim erstellen.
  - **OnEnter()** – Aufruf beim Wechsel in diesen Gamestate
  - **OnUpdate()** – Wiederholender Aufruf, wenn aktiv
  - **OnExit()** – Aufruf beim verlassen des States.

AI-State Machine

- Die EnemyAI2 zeigt momentan folgendes Verhalten
  - **Patrol** – Ausschau nach dem Spieler halten
  - **FlyToWaypoint** – Zum nächsten Wegpunkt fliegen
  - **FlyInFormation** – In Formation fliegen
  - **FlyToPlayer** – Zum Spieler fliegen
  - **AttackPlayer** – Vor dem Spieler herfliegen und angreifen

- Die einzelnen States können leicht erweitert oder gegen andere ausgetaucht werden.
  - Ein besonderes Feature ist,
    dass sie erst zur Laufzeit zugewiesen werden

Florian Mätschke, Christian Woizischke,
Markus Lorenz, Martin Fiebig

Weapons & Items

Weapons

- 4 Waffen stehen bisher zur Auswahl
  - UltraPhaser
  - Stinger Rockets
  - Minigun (2x)
  - PlasmaGun (von Markus Lorenz)

- Alle Waffen
  - sind per XML konfigurierbar
  - basieren auf einer BaseWeapon Klasse
  - und werden von einem WeaponManager verwaltet.

Weapon Manager

- Der Weapon Manager verwaltet
  - die Bewaffnung,
  - Munition,
  - und Auswahl der Primär- und Sekundärwaffen.
  - Er kümmert sich um die Positionierung der Waffen am Spieler-Model.
- Er bietet ein einfaches Interface zum Zugriff auf die vorhanden Waffen.
- Der Weapon Manager kann jedem GameObject zugewiesen werden.
  - Zum Beispiel dem Spieler, Enemies oder sogar einem Projektil, wenn man möchte.

Items

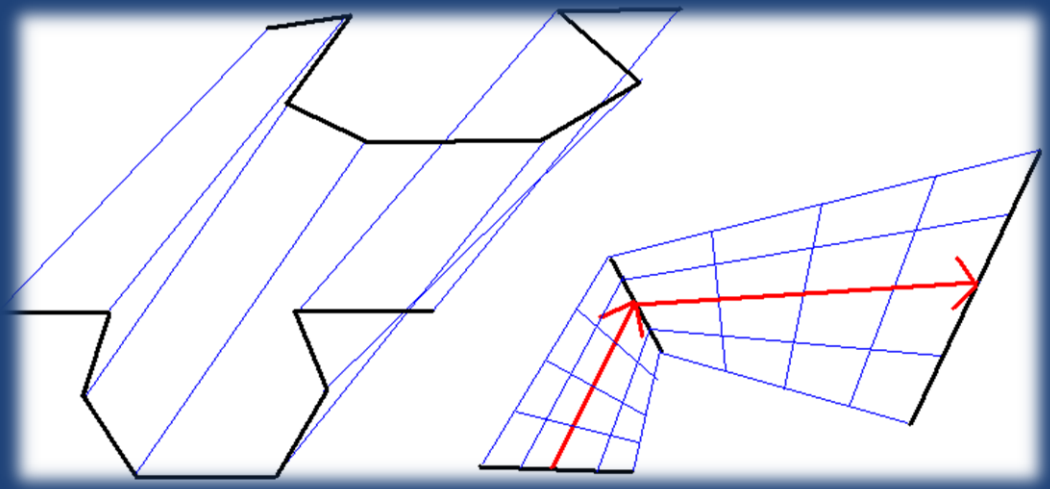
- Typ bestimmt das Modell
- Hervorheben vom Modell durch Bewegung (auf und ab)

Thorben Schulze, Richard Wolfer

CANYON GENERIERUNG & LEVELEDITOR

Canyon

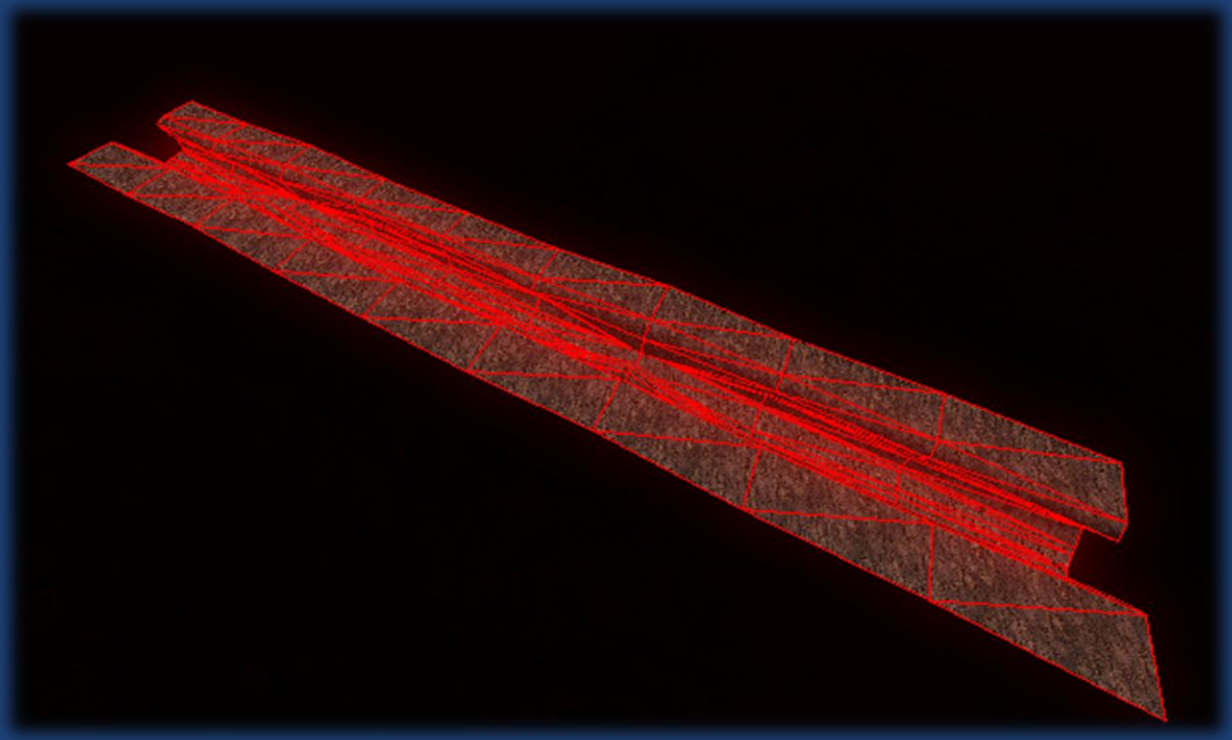
- Genieren eines Canyons anhand von Profilen
- Richtung wird angegeben durch Vektoren



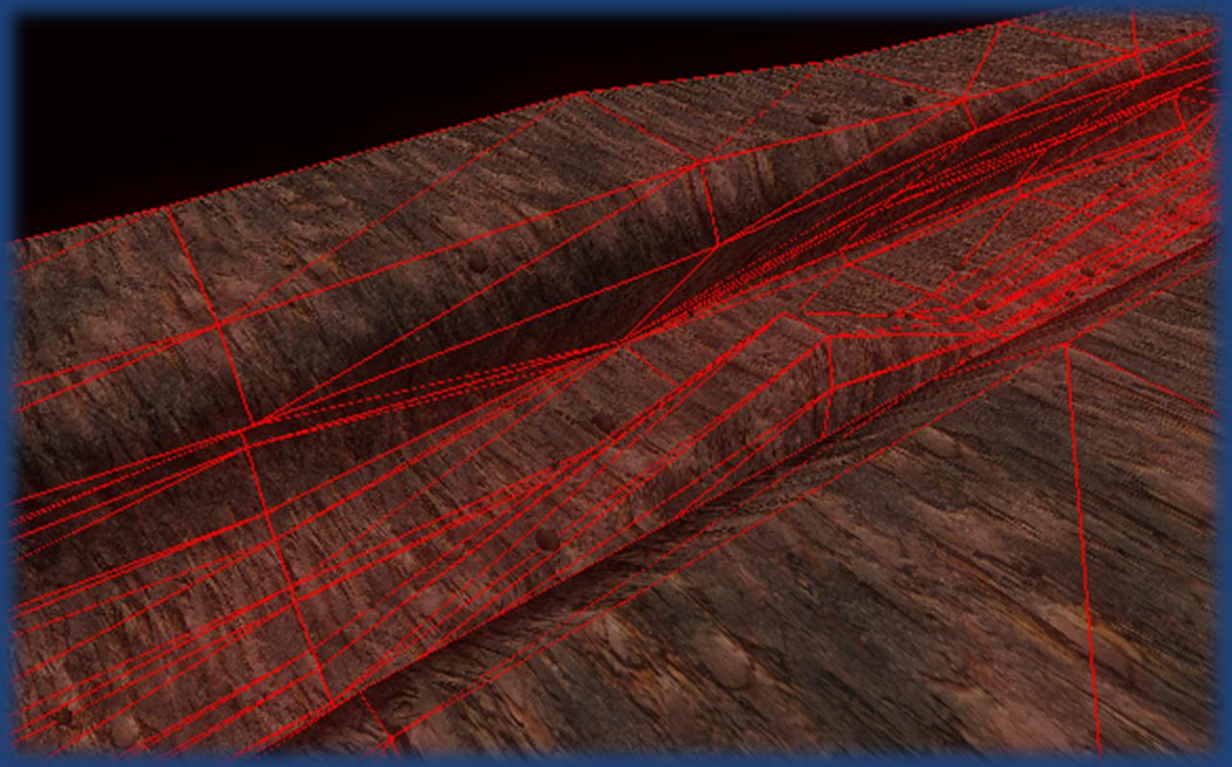
Canyon

- Berechnung von Normalen anhand der umgebenen Dreiecke
- Berechnung von u,v-Werten für die Textur
- Anzeigen eines bestimmten Canyonabschnitts
- Löschen alter Segmente

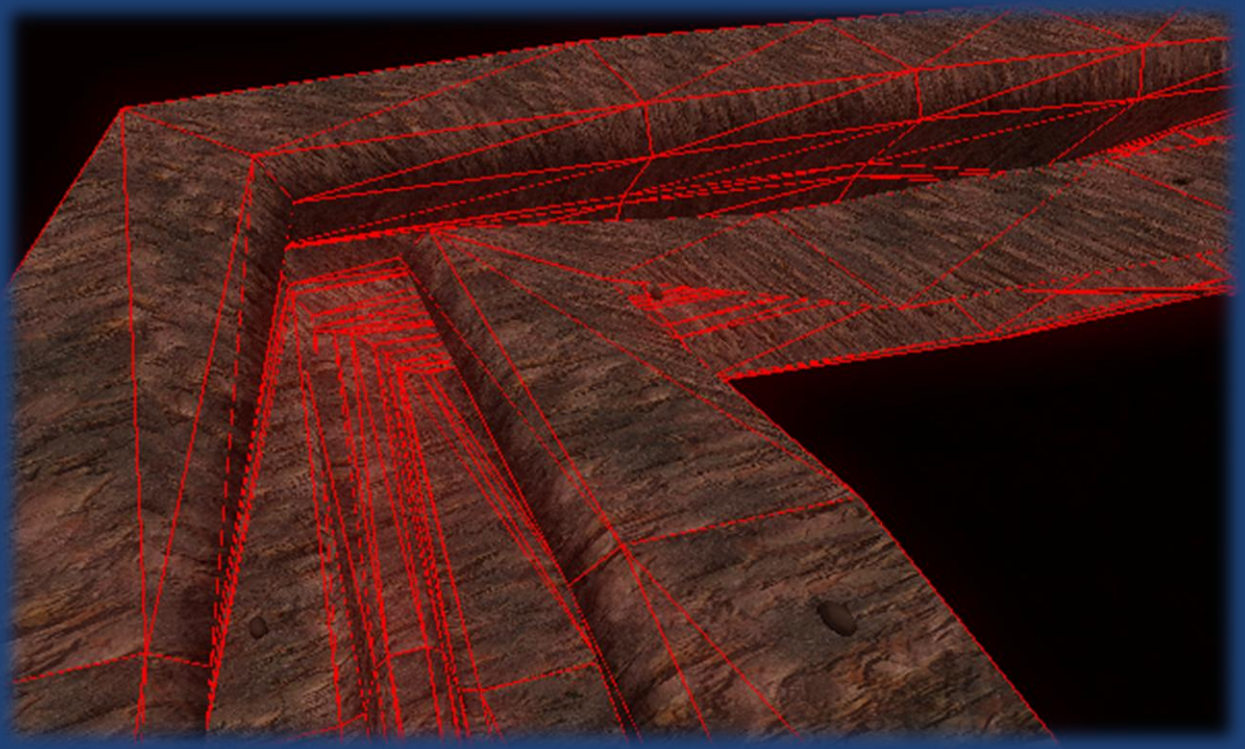
Canyon



Canyon

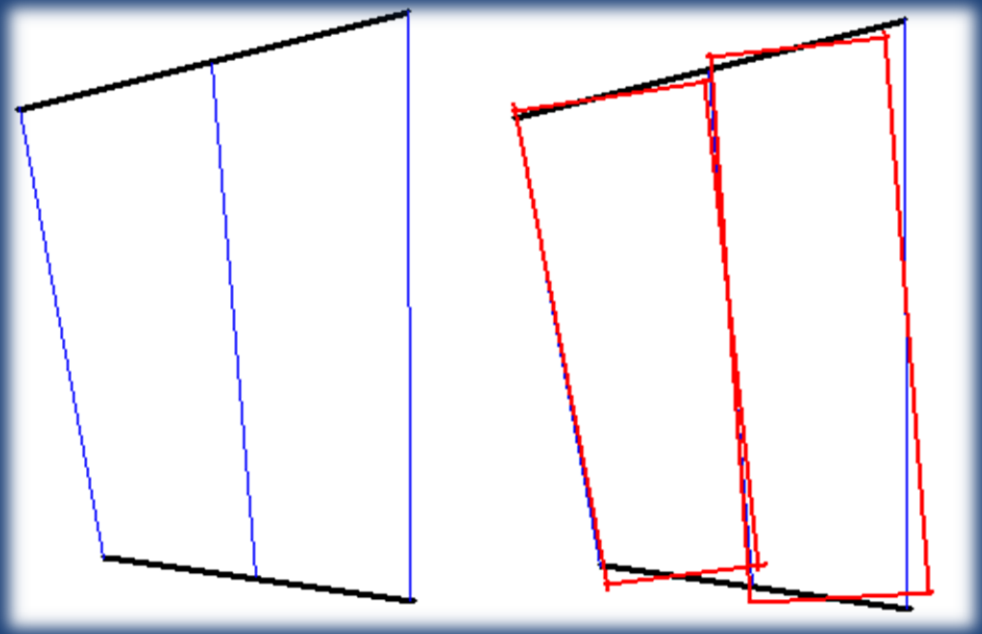


Canyon



Canyon

- Erstellen von Bounding-Boxes



Level Editor

Aus Krankheitsgründen von Richard Wolfer fehlt dieser Teil der Präsentation!

Sascha Lity, Malte Mauritz
Martin Fiebig, Markus Lorenz

USER-INTERFACE

Sascha Lity & Malte Mauritz

Menu, Profile, Highscore

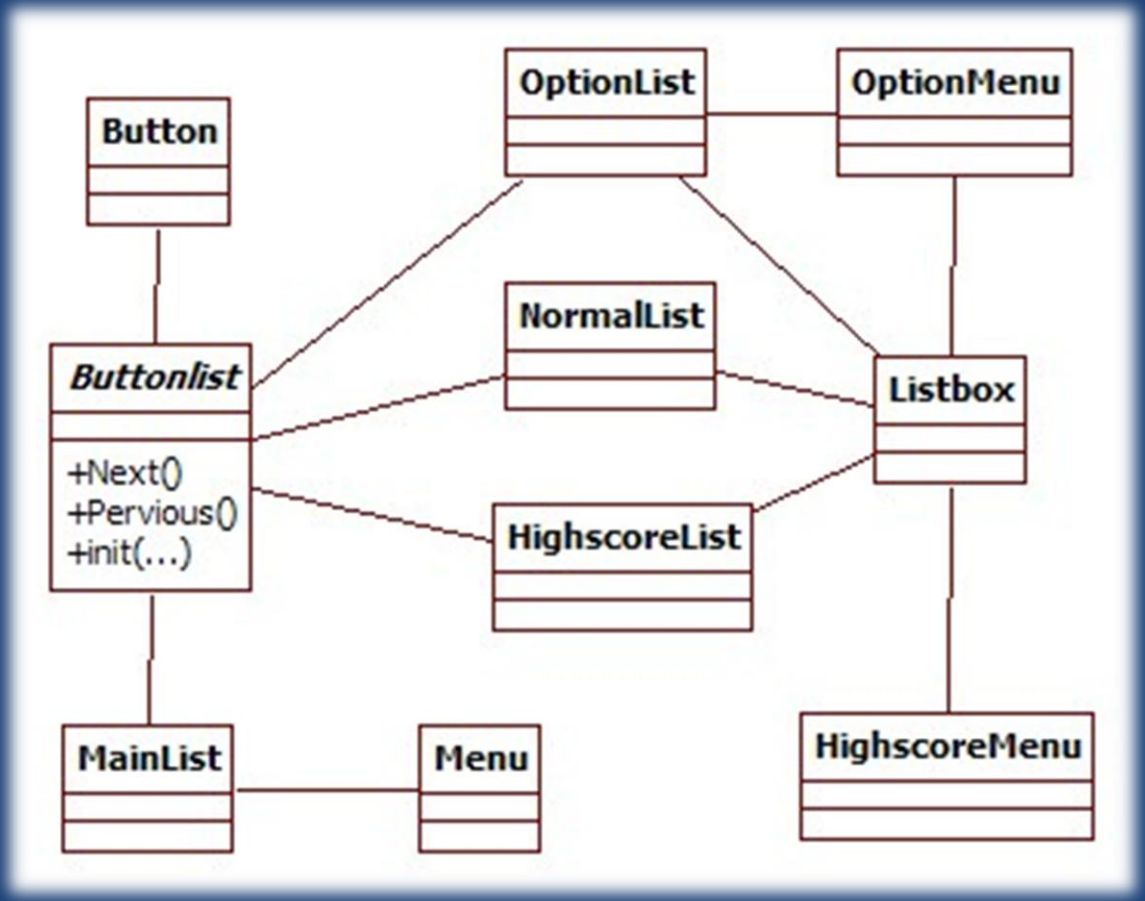
Struktur Menü

- Menü besteht aus:
  - Hauptmenü
  - Optionsmenü
  - Highscoremenü
  - Gleiteruahl
  - Profil
- Verlinkung über Buttons
- Interaktion in Untermenüs

Struktur Menü

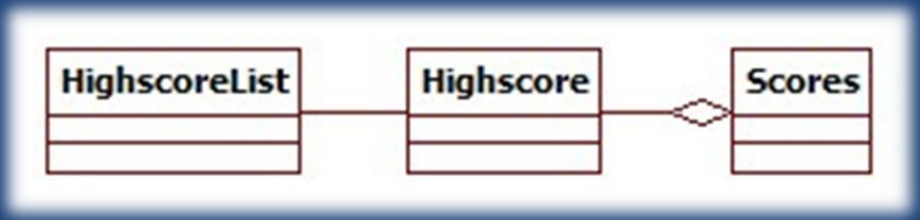
- ButtonList als Hauptelement des Menüs
  - Vereinfachung der Steuerungsprogrammierung
  - Sammlung der Buttons
- Buttonsteuerung auch per Eventsystem möglich
- Prompt zur Interaktionsaufforderung vorhanden
  - Einsatz mit Tastenbelegung
- Keine Templates für Menüelemente in XNA

Struktur Menü



Struktur Highscore

- Verwaltung der Scores
  - Doppeltgeschachtelte Liste
    - Schwierigkeit
  - Ausgabe nach Schwierigkeit
  - Einfügen
  - Reset
- Zufällige Liste beim ersten Start
  - Fehlervermeidung



Profil: Einstellungen

- Profil ermöglicht Speicherung eigener Einstellungen
- Möglichkeit 5 Profile anzulegen
- Bestehende Profile nicht veränderbar
  - nur löschbar
- Spiel kann nicht ohne aktives Profil gestartet werden

Profil

- Profilauswahl wird bei einem bestehen Profil übersprungen
  - über Menü kann man zur Profilauswahl zurückkehren
- Steuerbar per Maus oder Tastatur
- Grafiken von Manuel (Hintergrund) und Danny (Restliche Grafiken)
- Button werden dynamisch durch eine XML-Datei erzeugt (verantwortlich Malte)

Menü

- Steuerung, um durch die einzelnen Menüs zu Navigieren oder das Spiel zu starten
- Möglichkeit per Maus sowie per Tastatur

Gleiterauswahl

- Spielgleiter wird gewählt und das Spiel kann gestartet werden
- Maus und Tastatur stellen Steuerungsmöglichkeit dar
- Per Mausbewegung kann man den Gleiter rotieren lassen
- 5 verschiedene Gleiter zur Auswahl
- Auswahl und starten des Spiels

Input

- Bindingmöglichkeit für Maustasten

- Funktionen zur Abfrage, ob vorher gebindete Tasten gedrückt bzw. wieder losgelassen wurden vorhanden

- Diese Abfrage findet für Tastatur sowie Maus statt
  - basiert auf dem Beispielcode, angepasst an den Binding-Mechanismus

Live Demo

Intro, Profil, Menusteuerung
und Gleiterauswahl

Martin Fiebig

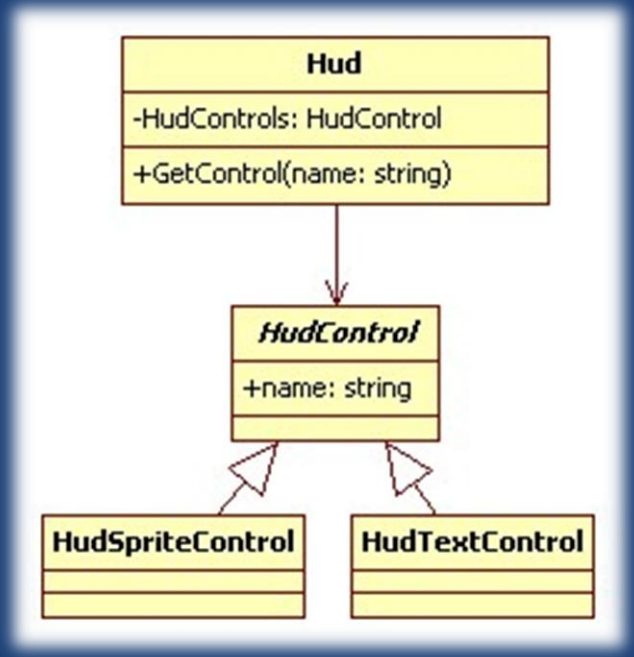
HUD

HUD - Head Up Display

- HUD-Elemente
  - 2D Grafiken
  - Text
  - Beeinflussbar
    - Text
    - Position
    - Verhalten

HUD: Elemente

- Scrollen
  (Von Position zu Position)
- Effekte
  - Grow (Wachsen)
  - Shrink (Schrumpfen)
  - Shakkle (Wackeln)
  - Pulse (Pulsieren)
  - FadeIn (Einfaden)
  - FadeOut (Ausfaden)
- Ein- und Ausblenden



Markus Lorenz, Martin Fiebig

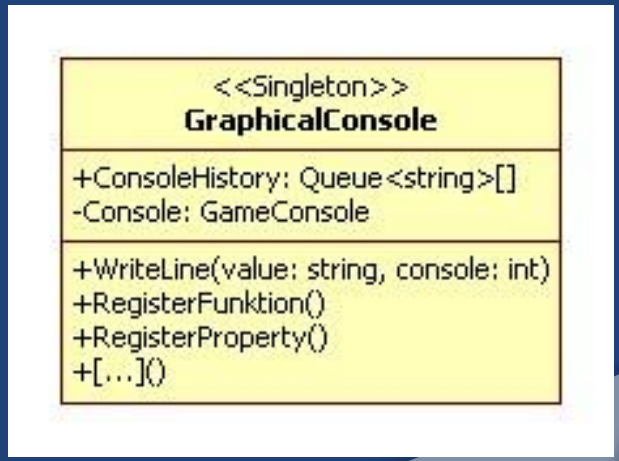
Console

Console

- 2 Konsolen
  - „Debug“-Console (Quake like)
    - Anzeigen bzw. Ausblenden über „^“
    - Eingabe von Kommandos (siehe Markus)
    - Ausgabe von Debug Informationen
  - „Life“-Console (UT like)
    - Schmaler streifen (derzeit) am oberen Teil des Bildschirms
    - Für „Chat“ und Detailinformationen für den Spieler gedacht. (Anspornende Nachrichten „Multikill“ „Ultrakill“, besondere Ereignisse)

UT-Like

- 5 angezeigte Zeilen
- Im gesammten Programm aufrufbar



Quake-Like

- Ausgabe von Umfangreichen Debug Informationen über „WriteLine“
- „Auto-Vervollständigung“ bei der Kommandoeingabe
  - Vorschläge können mit **Tab** durchgeschaltet werden
- Scrollbar

Console

- Über eine Konsole können durch Texteingabe
  - Funktionen aufgerufen werden
  - Eigenschaften geändert werden
- Einfache Registrierung
  - GameConsole.RegisterObjectFunction(
    “MeinObjekt“, myObject, “FunktionA“);
  - GameConsole.RegisterObjectProperty(
    “MeinObjekt“, myObject, “EigenschaftA“);
  - GameConsole.RegisterStaticFunction(
    “MeineKlasse“, typeof(MyClass), “FunktionB“);
  - GameConsole.RegisterStaticProperty(
    “MeineKlasse“, typeof(MyClass), “EigenschaftB“);

Console

- Einfache Nutzung
  - Funktion mit Parametern aufrufen
    - “MeinObjekt.FunktionA 'Parameter String' 34 -2,34“
    - Rückgabewert wird ausgegeben
  - Wert einer Eigenschaft abfragen
    - “MeinObjekt.MeineEigenschaft“
    - Rückgabewert wird ausgegeben
  - Wert einer Eigenschaft setzen
    - “MeinObjekt.Meine Eigenschaft 'Neue String'“

Console

- Möglichkeit, Befehle zu vervollständigen
  - string[ ] GameConsole.GetSuggestions(
    string commandStart)
- Möglichkeit Informationen abzufragen
  - Liste aller registrierten Objekte und Klassen
  - Registrierte Funktionen und Eigenschaften zu einem Objekt
  - Parametertypen zu einer Funktion
  - Rückgabewert zu einer Funktion oder Eigenschaft

Console

- Das Parsen der Textparameter geschieht automatisch für Int, Double, Float, Bool und Enumerations
- Zusätzlich können TypeParser registriert werden, die die abstrakte Klasse GameConsoleTypeParser implementieren
- Die GameConsole wurde für das Balancing und die Bug-Suche geschrieben

Features im Überblick

SOUND & CONTENT

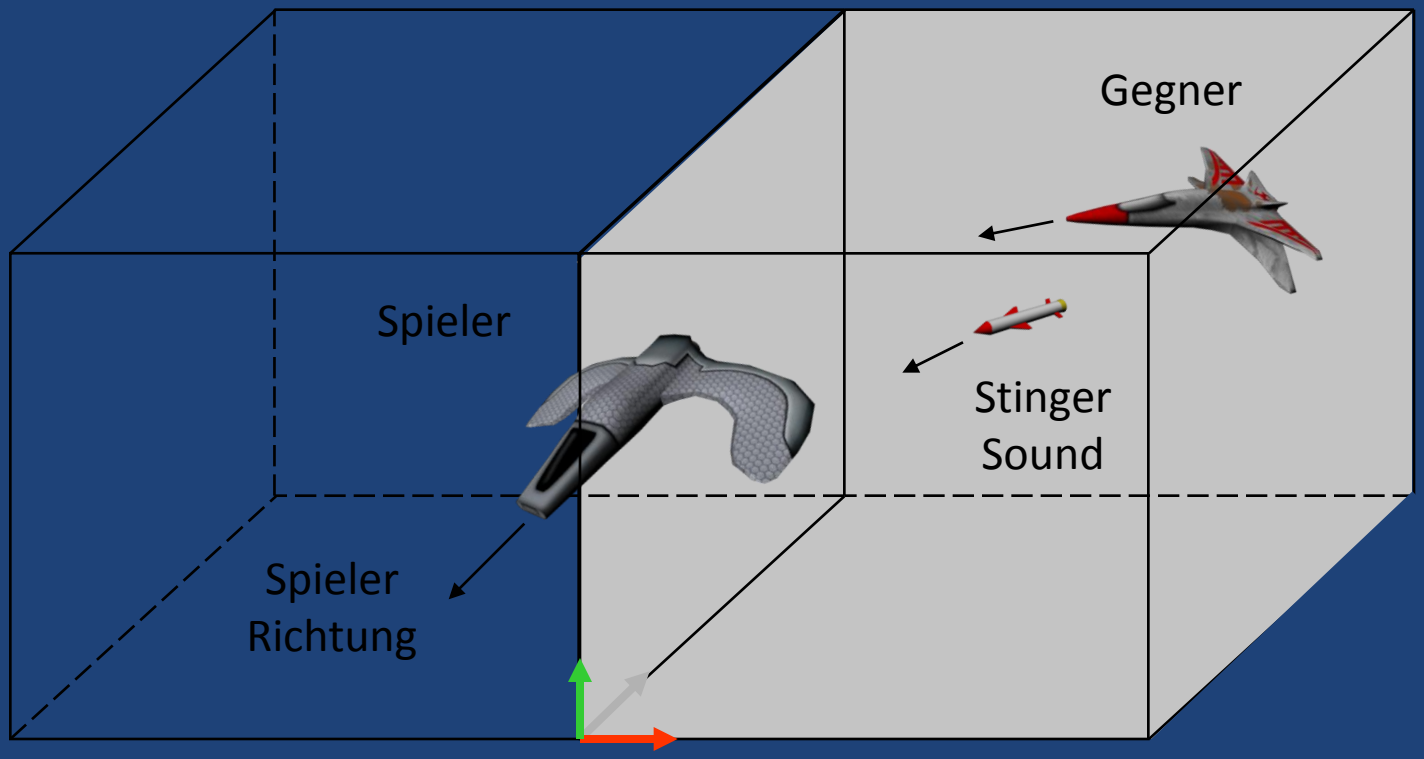
Manuel Rodriguez

Sound System

Leistungsmerkmale

- Musik und Effekt Wiedergabe.
- Objekt-orientiertes Sound System
  - Jeder Sound wird von einem Objekt repräsentiert.
- 3-dimensionale Effekt Erweiterung:
  - Sound Träger spielt den Sound (Rakete).
  - Zuhörer ist der Spieler (Raumschiff).
- Musikbox für die Wiedergabe der Hintergrundmusik.

XNA 3D Sound



- Aus welchem Lautsprecher kommt der Stinger Sound?

Sound Architektur

- Mehrschichtige Microsoft Architektur

Musik- und Effekterstellung

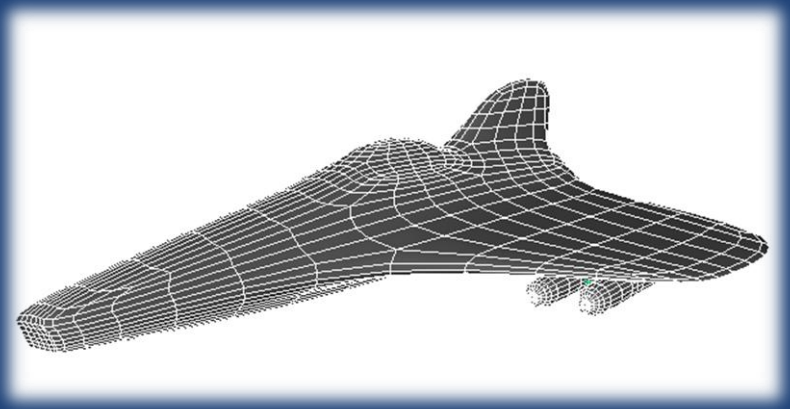
- Digitale Audio Produktionen mit:
  - Steinberg Cubase VST, Reason, ReBirth, …
  - Synthesizer: Korg, Yamaha, Roland, …
- Integration der Sounds in XNA mit XACT
- WAV/PCM RAW Schnittstellenformat
- ADPCM Kompression (75% durch XACT)
  - Größe der finalen Soundbank ca. 110 MB
- Kompatibel mit XBOX 360 und Windows

Manuel Rodriguez, Markus Lorenz

Sounds

Sounds

- Im Spiel stehen über 34 verschiedene Effekt-Sounds zur Verfügung
- Zum Beispiel für:
  - Explosionen
  - Waffen
  - Environment
  - Powerups
- Die Hintergrundmusik wurde von Markus Lorenz und Manuel Rodriguez komponiert.

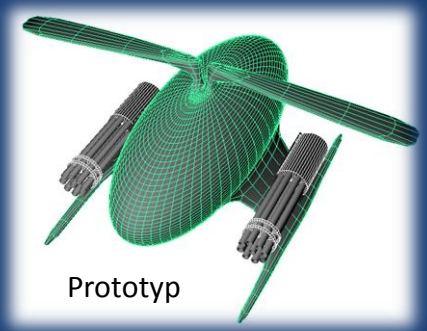


Manuel Rodriguez

3D Models

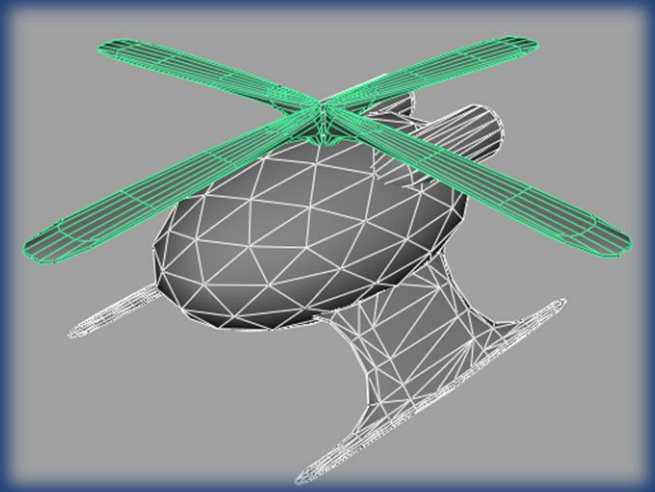
Spielemodelle

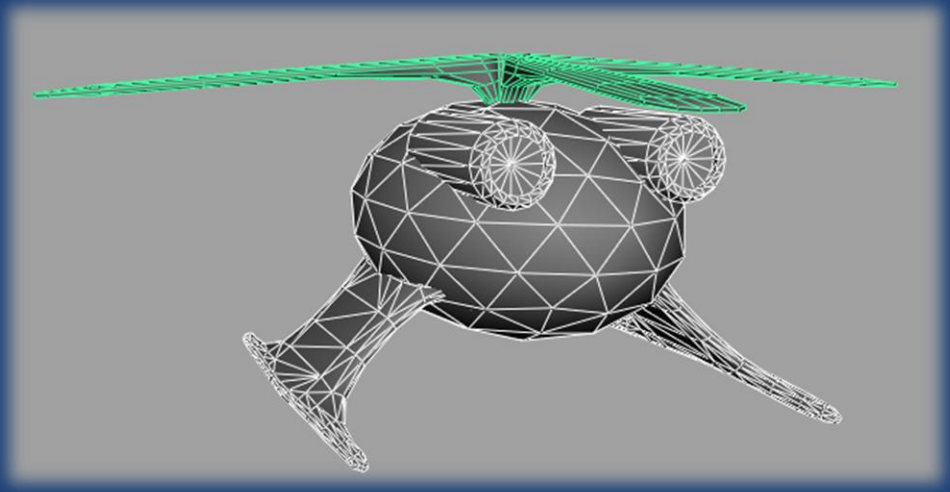
- Erstellung mittels Autodesk Maya
- Durchschnittlich. 600 Polygone pro Model
- Maximale Größe: 24x24x24 cm
- FBX Schnittstellenformat
- Model Dekomposition
  - Helikopter
  - Minigun



Prototyp

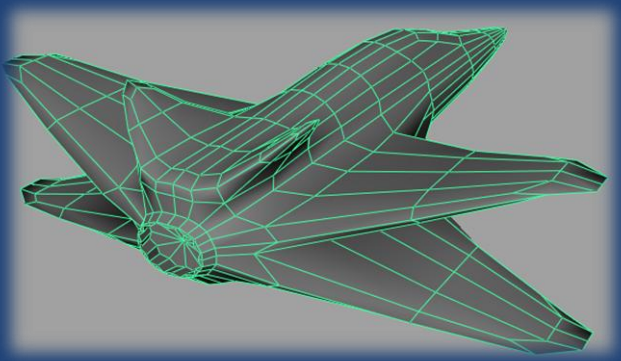
Model Impressionen

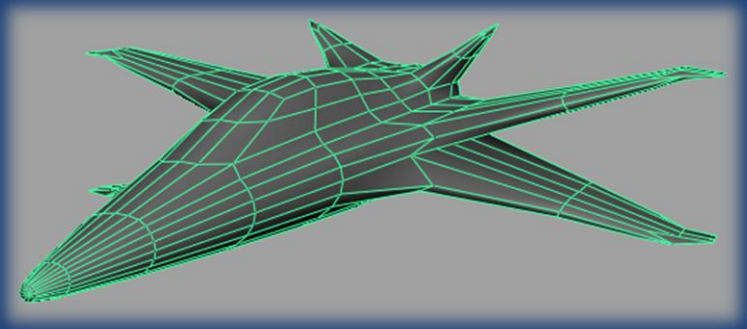


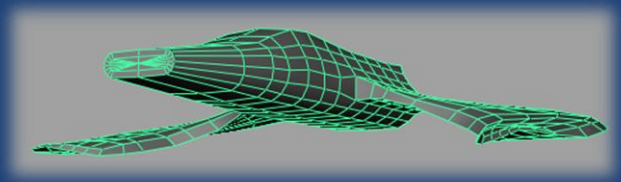


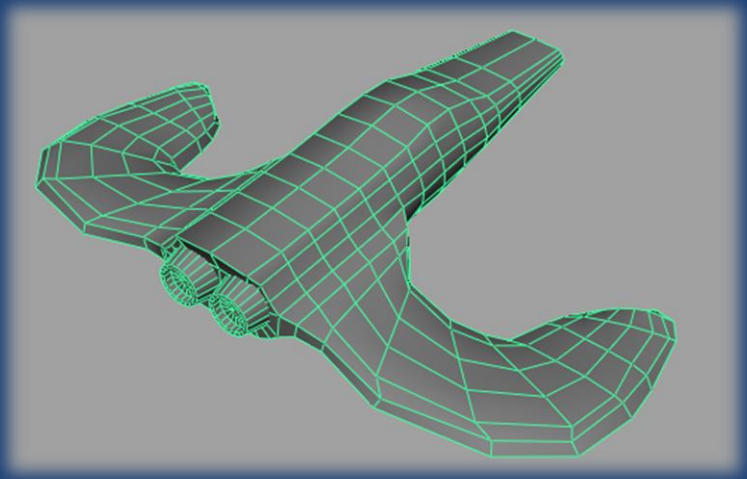
- Dekomposition des Helikopters in den Rumpf und den drehbaren Rotor.
- Einsatz von Polyhedra zur Reduktion von Polygonen.

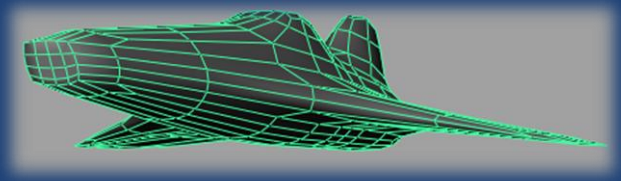
Model Impressionen



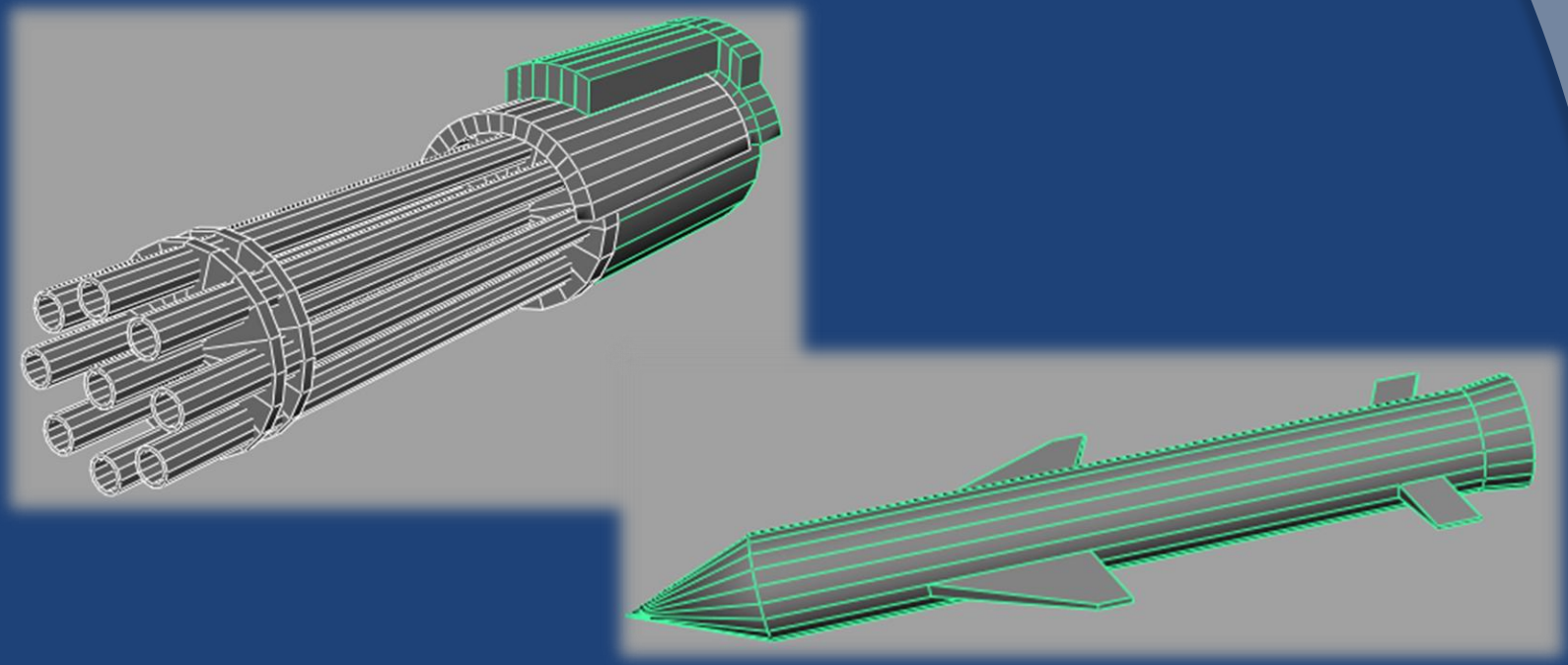




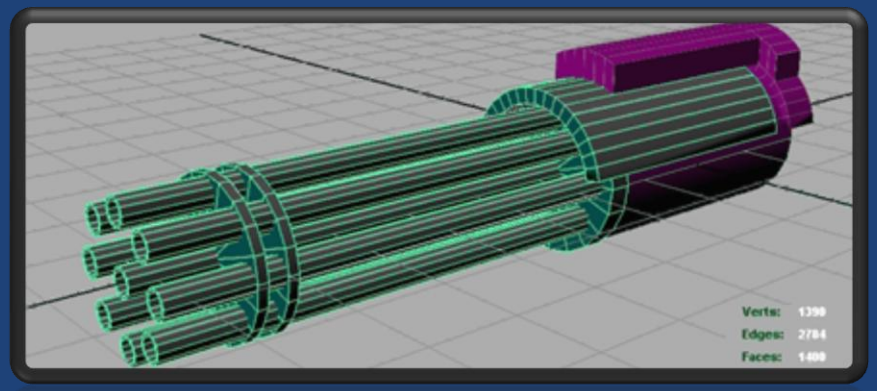




Model Impressionen



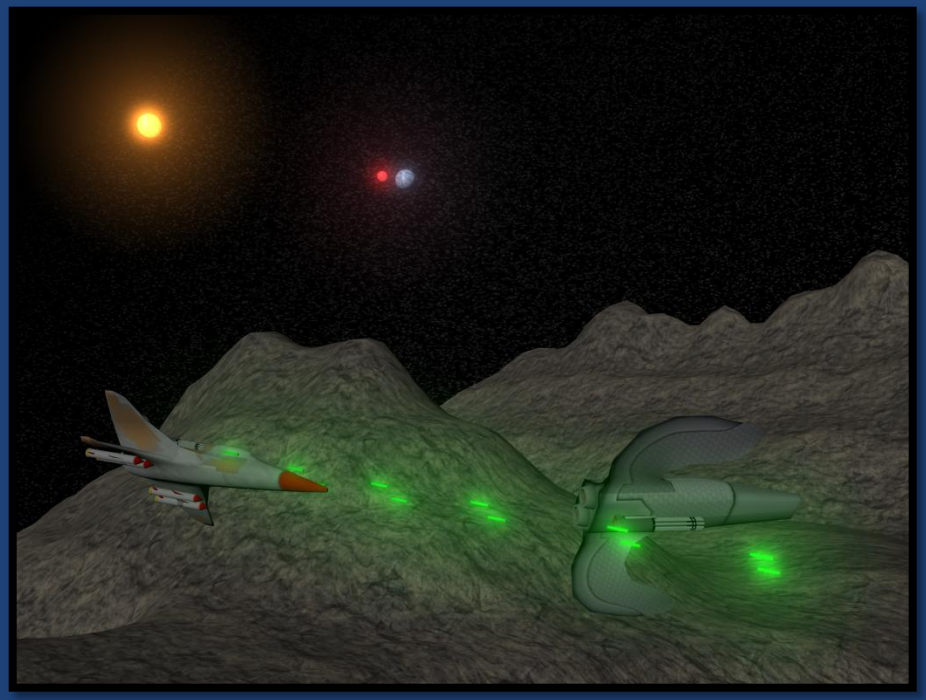
- Minigun mit rotierbaren Lauf.
- Stinger Raketen Model.

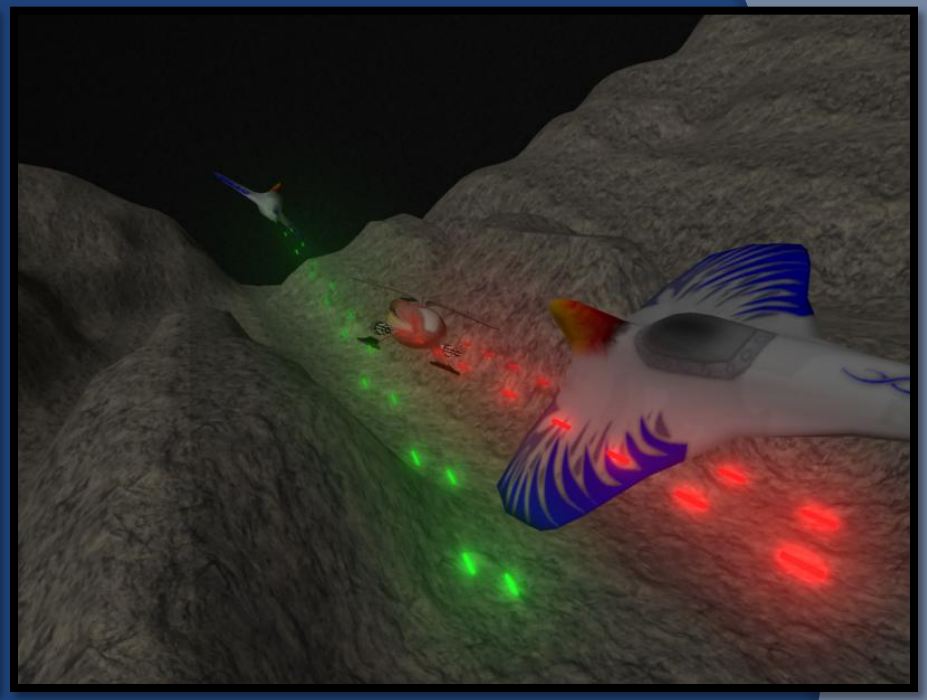


Live Demo

Animationsprototypen

Hintergrundbilder





- Die Szenen wurden in den Menüs als Hintergrundbilder eingesetzt.
- Modelliert mit Autodesk Maya.

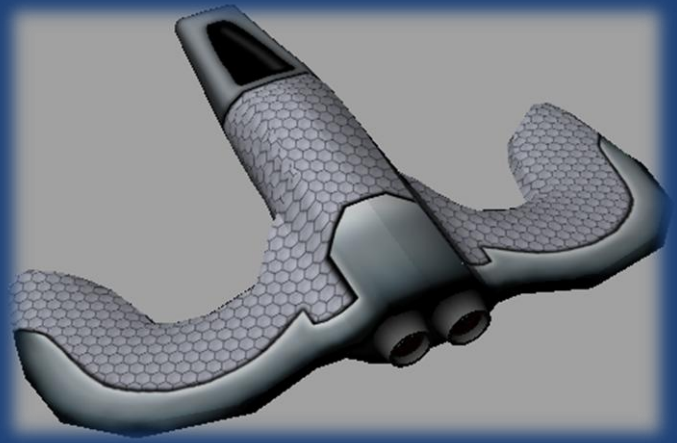
Danny Melching

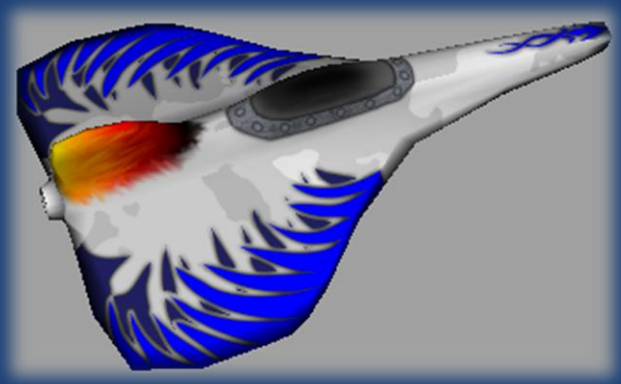
Texturen und Grafiken

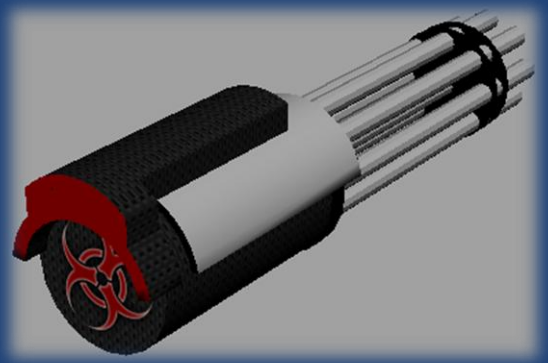
Grafiken

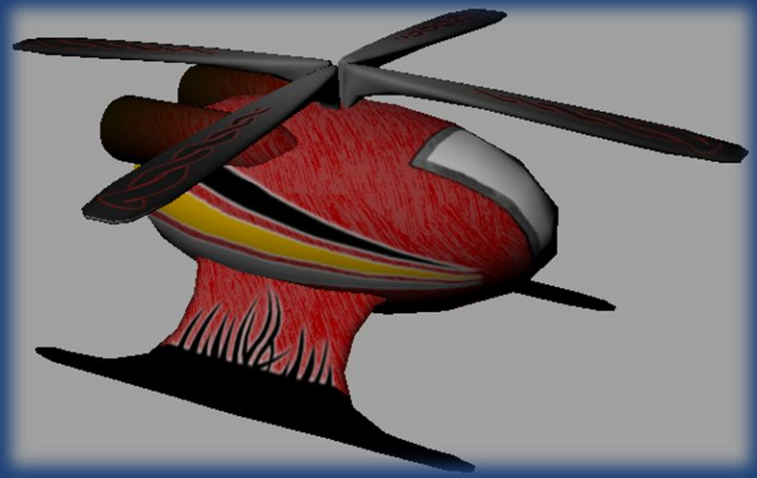
- Texturen für alle Menus vorhanden

Texturen

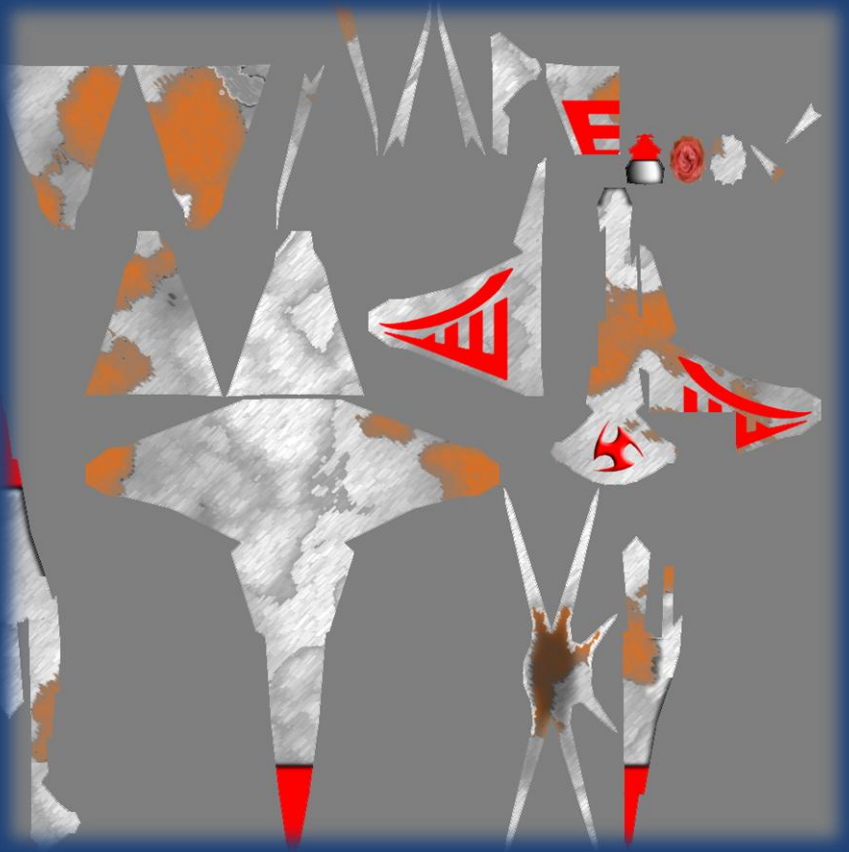








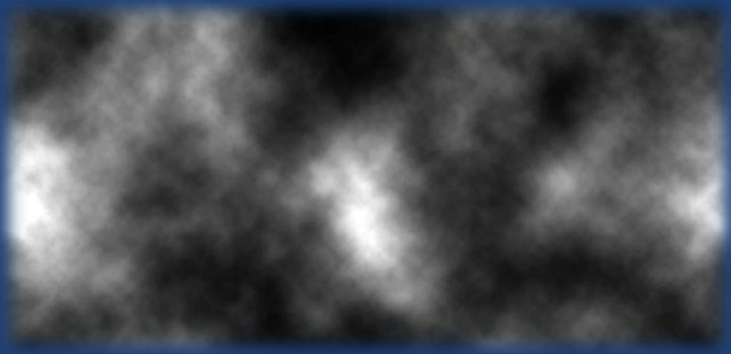
UV-Map zur Textur



Heightmapping

- Heightmapping realisiert um dem Canyon eine Struktur zu geben

Standard Mesh



Heightmap

Angewendete Heightmaps

Markus Lorenz, Martin Fiebig

Website

Website

- http://canyonshooter.tu-bs.de
- Blog-Software: WordPress
- WordPress Plugins
  - NextGEN Gallery
  - SyntaxHighlighter
- Canyon-Shooter Theme

Implementierung am Ende des Projekts

Der zweite Teil des Zwischenberichts bezieht sich auf die geplante Implementierung am Ende des Projekts.

Engine

Audio

Es soll zum Ende des Projektes der 3D Sound, Laser- und Explosionseffekte sowie das Abspielen von Musikstücken implementiert sein.

Graphics

Es soll eine globale Grafikklasse zum Einstellen von Grafikoptionen und Weitergabe dieser an XNA/Direct3D erstellt werden. Die Beleuchtung mit mehreren Lichtern und dem Phong-Modell soll eingebaut werden. Die Unterstützung für normal mapping soll implementiert werden. Das Anzeigen von Schatten soll ermöglicht werden (Wunschkriterium).
Die Effekte sollen noch erweitert werden fur Explosion, Düsen-Antrieb, Erdbeben, und weitere Special Effects für extra Waffen.

Den ganzen Beitrag lesen »

Kategorie Allgemein | 0 Kommentare
15. Januar 2008 | Markus

Suche

Archiv

» Januar 2008
» Dezember 2007
» November 2007
» Oktober 2007

Kategorien

» Allgemein (9)
» Probleme (4)
» WordPress (3)

Meta

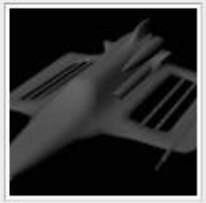
» Anmelden
» XFN
» WordPress
» WPD

CANYON-SHOOTER
Teamprojekt

Blog Galerie Impressum

Galerie

3D Modelle



Hier findet ihr ein paar gerenderte 3D Modelle für unseren CanyonShooter.

17 Fotos

Unsere Arbeitsumgebung

Ein paar Bilder vom Informatik-Zentrum und unserem Projektraum.

11 Fotos

Screenshots

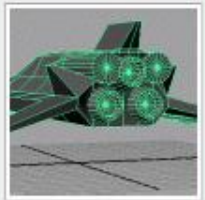
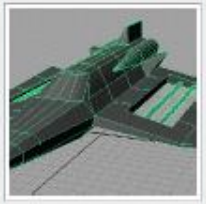
Diese Galerie enthält Screenshots aus dem Programm.

25 Fotos

CANYON-SHOOTER
Teamprojekt

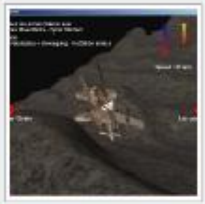
3D Modelle

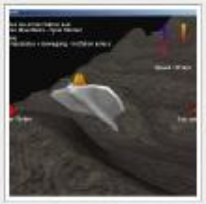
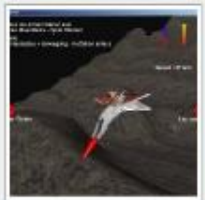
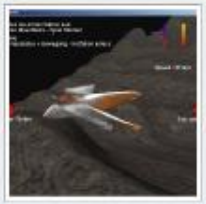
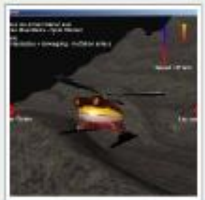
Hier findet ihr ein paar gerenderte 3D Modelle für unseren CanyonShooter.

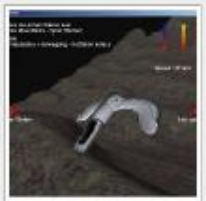
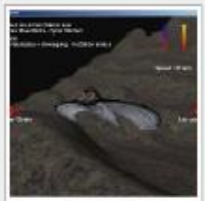










3

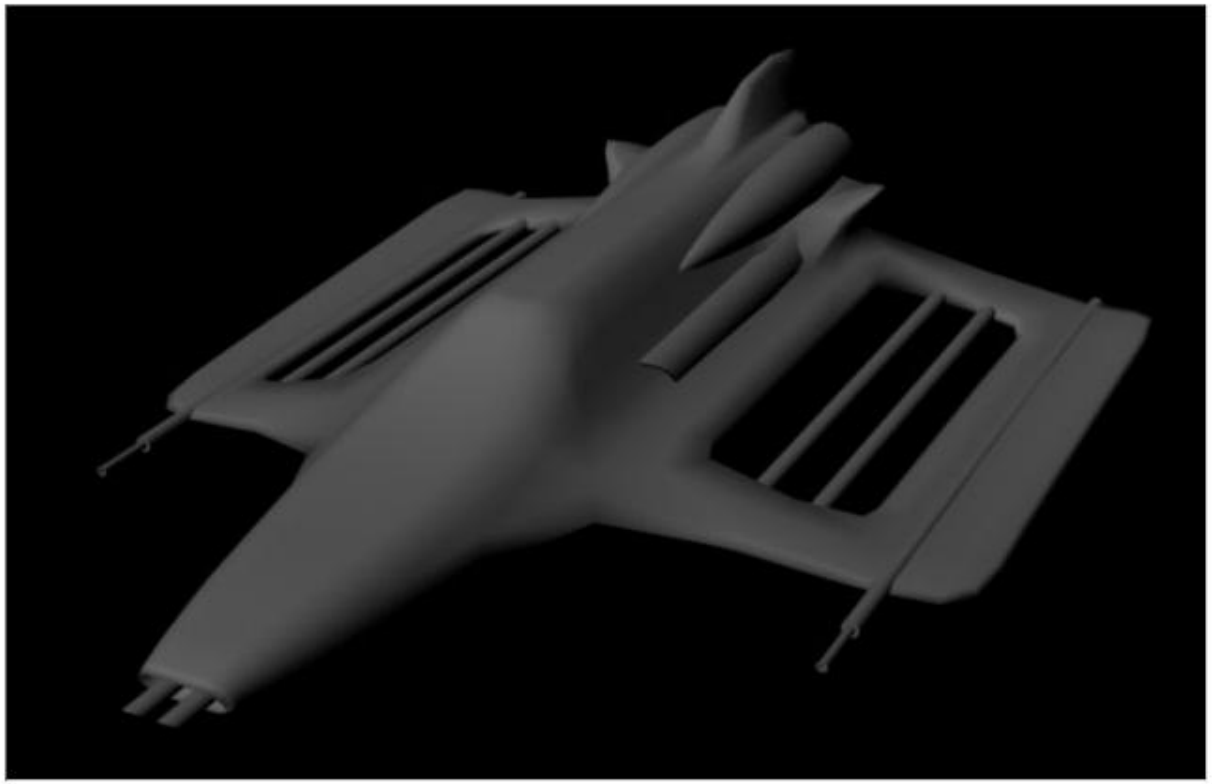


Image 1 of 17 Full Size Next >

close or Esc Key

Geplante Features, Verbesserungen, Bugfixes, Beta-Test

AUSBLICK

Bis zum 31.03.2008

Geplante Features

Geplante Features

Zum erfolgreichen Abschluß des Projektes am 31.03.2008 sind noch folgende Features geplant:

- Level Editor
  - Canyon-Generator
    - Damit es im Spiel endlich richtige Levels gibt.

Geplante Features

- Spiel
  - Ausgeklügeltes Balancing
  - Spiel-Start im Canyon:
    - Aufmachung wie bei einem Autorennen
  - Talking Enemys
  - Skalierung des Menüs für unabhängige Auflösungen
  - Destruction-Parts für Enemies und Spieler
  - Tod beim Verlassen des Canyons

Was man noch machen könnte

Geplante Features

Geplante Features

- Canyon live generiert per Musik
  - Winamp-Plugin
- Multiplayer Modus
  - Gleiter und Jeep am Boden
  - Netzwerkunterstützung
- Eine bessere Physik-Engine z.B. Havok
- Mehr Models, Waffen und Items.
- Enemy-AI Labor (z.B. Für ein SEP-Projekt)

Mögliche Verbesserungen

Verbesserung

Verbesserungen: Canyon

- Bounding-Boxes könnten sich noch etwas besser den Canyon-Segmenten anpassen
- Berechnung des gesamten Canyons beim Ladevorgang – bisher noch zur Laufzeit
- Wunsch
  - Interpolation zwischen den Profilen nicht linear

Verbesserung: Speed

- Das größte Manko am Canyon Shooter ist zur Zeit noch die Geschwindigkeit der Physik-Engine.

- Ein Lösung wäre Multi-Threading, um die Power von heute üblichen Multicore-Prozessoren besser auszunutzen.
  - Ein Ansatz dazu existiert bereits, bereitet allerdings noch Probleme

Verbesserungen: Menü

- Vollständige Menüs
  - 3 Highscore Menüs
  - Optionsmenü
    - Grafik/Sound
    - Tastenbelegung
- Finale Grafiken
- Interaktion in den Untermenüs
  - Optionen einstellbar
  - Highscore anzeigbar

Verbesserungen: Enemies

- Bessere AI mit mehr States.
- Nicht jeder Enemy mit der selben AI sollte gleich reagieren
  - Mehr Zufälligkeiten
  - Unvorhersehbarkeiten
  - Überraschungsangriffe

Verbesserungen: Waffen

- Eine Waffe sollte Upgrade-Möglichkeiten haben
  - Verbesserung der Durchschlagskraft
  - Mehr Projektile pro Schuss
  - Andere Sounds, etc..

Verbesserungen: Texturen

- Verfeinern des Heightmappings
- Erstellung einer Textur für ein „Special“ Raumschiff
- Evtl. unterschiedliche aufeinanderfolgende Landschaftstexturen für den Canyon einbinden

Verbesserungen: Waffen

- Eine Waffe sollte Upgrade-Möglichkeiten haben
  - Verbesserung der Durchschlagskraft
  - Mehr Projektile pro Schuss
  - Andere Sounds, etc..

Bis zum 31.03.2008

Bugfixes und Beta-Test

Bugfixes

- Die meiste Zeit bis zum Ende des Projektes für Bugfixing
- Ziel ist eine möglichst stabile Version für die Beta-Testphase
- Bis dahin muss noch eine Bug-Tracking Plattform aufgesetzt werden.

Beta-Test

- Nach Abschluss des Projektes am 31.03.2008
- Freiwillige Teilnahme
- Danach Bekanntgabe über
  - Mailinglisten, Foren, News
  - Und auf der deutschen Microsoft XNA Website

Weitere Ziele

- Referenzprojekt für XNA-Spieleentwickler
- Artikel in PC-Spielemagazinen
  - Spiel als Beilage
- Zeitungsartikel