

Klausurvorbereitung

- Es gibt keine Minuspunkte, aber halbe Punkte
- Antwortlänge an Punkte pro Frage anpassen

Transformation: (Local Space)

- World
- View
- Projection

Zugelesen für Rechnungen:

- Taschenrechner (Windows oder Physicks)

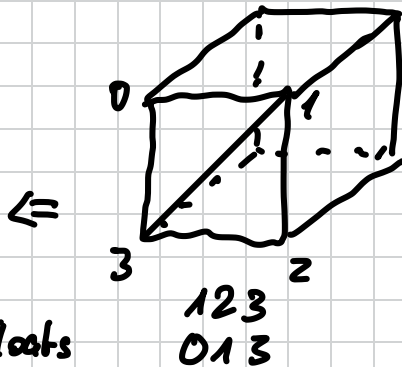
- Reihenfolgen und Sortierungsfragen auch
- Fragen sind auf Englisch, Antworten können auch auf Deutsch sein
- Datentypen oder Enumwerte aus D3D (z.B. WM_QUIT, WM_DESTROY)
- keine History und Unity kommt nicht ran

Themengebiete:

- **General:**
 - Graphics, C++ (z.B. warum C++ benutzen)
 - Unterschied zw. Pointer und Handle, Pointer-Pointer
 - COM (Component Object Model)
 - generelle Graphics Programmingsachen
- **Fenster:**
 - Fenster Nachrichten (Typen aus Vorlesung)
 - Message queue
 - Wozu braucht man Nachrichten
 - Message queue - Funktionen: Window Prop
- **Graphics - API:**
 - Welche gibt es?
 - Direct 3D
 - Fixed Function vs. Programmable Pipeline
 - Elemente der Pipelines: Vertex, Mesh, Transformation, Spaces, Projektion, Primitive Types, Unterschied List Strip, Texture Mapping (was ist eine Normale)
 - Homogenisierte Koordinaten und dessen Bedeutung
 - Berechnungen: Koordinaten, FPS, Delta Time, Größe von Strukturen (z.B. eines Index buffers kiBiByte bezeichnen => Indices zählen, z.B. Vertex buffer berechnen (umschreiben, nicht direkt gestellt), z.B. Texture buffers Mipmapping (zählen der Stufen)
- **Textures:**
 - UV-Mapping, UV-Tiling (Address Modes)
 - Bild von Modell und gefragt wie viele Vertices benötigt werden und welche

- **Lighting** :
 - Sources, Lichtbestandteile (Diffuse, ...)
 - Vertex - vs. Pixel lighting
- **Stages** :
 - Screen - Stage
 - Rasterizer - Stage: Frontfaces / Backfaces
- **Direct 3D / Direct X** :
 - Direct 3D9 und Direct 3D11
 - ID3D11 Device Context, IDirectDevice9
 - konkrete Datentypen aus Framework
 - ID3D11 Device vs. ID3D11 Device Context
- **Shader** :
 - Shader-typen (6) :
 - Vertex shader
 - Hull Shader
 - Domain Shader
 - Geometry Shader
 - Pixel shader
 - Compute shader
 - Zu jedem Shader Grundfunktionalitäten, wozu sie da sind
 - Shader resources (Vertexbuffer, Indexbuffer, Constantbuffer)
 - HSL :
 - High Level Shader Language
 - Genereller Sinn von Semantiken in HSL

Vertices berechnen:



Lichtberechnung als Ziel und Texturierung

2. Vertex gröÙe:

- Position: 3 Floats
- Normale: 3 Floats
- UV-Koordinaten: 2 Floats
- 1 Float = 4 Byte
- $\Rightarrow 8 \text{ Floats} \cdot 4 \text{ Bytes} = \underline{\underline{32 \text{ Byte}}}$ pro Vertex

1. Vertices = Ecken
 $\Rightarrow 24$ (4 pro Fläche)

Eine Normale pro Fläche wegen des Lichts, deswegen 24 Vertices statt 8

- Vertex color: 4 Floats
- Grafikkarte ist auf Floats optimiert

2 Byte = 1 Word

3. $24 \cdot 32 \text{ Byte} = \underline{\underline{768 \text{ Byte}}}$
 (Vertexbuffer gröÙe)

Indexbuffer berechnen:

- Primitive Type
- Triangle list (oder Linelist, Point list)
- 6 Flächen, 2 Dreiecke pro Fläche
- Mehrfachverwendung der Vertices
- Zusatzinfos nötig: wie viele Flächen und welche Form haben die Flächen
- wenn die Wahl freistellt: Triangle list

4. $12 \text{ Dreiecke} \cdot 3 \text{ Ecken} = \underline{\underline{36 \text{ Indices}}}$
 $\cdot 2 \text{ Byte} = \underline{\underline{72 \text{ Byte}}}$

BytegröÙe selbst wählen: Anzahl der Indices muss in die BytegröÙe passen