

Inhaltsverzeichnis

| | | |
|----------|--|----|
| | Einleitung | 15 |
| I | Erste Schritte | 19 |
| 1.1 | Voraussetzungen und Installation | 19 |
| 1.1.1 | Die Programmiersprache der Wahl: C++ | 19 |
| 1.1.2 | C++-Entwicklungsumgebung: Microsoft Visual C++ | 20 |
| 1.1.3 | Das DirectX-SDK | 20 |
| 1.1.4 | Der Sample Browser | 21 |
| 1.1.5 | SDK-Beispiele kompilieren und ausführen | 21 |
| 1.1.6 | Externe Module | 22 |
| 1.2 | Bestandteile eines DirectX-Programms | 22 |
| 1.2.1 | Modelle | 22 |
| 1.2.2 | Materialien, Texturen und Beleuchtung | 23 |
| 1.2.3 | Positionieren von Objekten und Kameraeinstellungen | 23 |
| 1.2.4 | Der Render-Loop | 24 |
| 1.3 | Das erste Beispiel: Ein Modell laden und anzeigen | 25 |
| 1.3.1 | SimpleSample11 aus dem Sample Browser installieren | 25 |
| 1.3.2 | Das Grundgerüst SimpleSample | 26 |
| 1.3.3 | DXUT | 32 |
| 1.3.4 | Modelle in einer Datei | 32 |
| 1.3.5 | Ergänzungen im Programmcode | 34 |
| 1.3.6 | Die Schritte im Überblick (SimpleSample) | 35 |
| 1.4 | Ein Modell in ein leeres Projekt einfügen | 35 |
| 1.4.1 | Shader und Konstantenbuffer | 36 |
| 1.4.2 | Kamera | 40 |
| 1.4.3 | Einfügen eines Modells | 43 |
| 1.4.4 | Die Schritte im Überblick (EmptyProject) | 44 |
| 1.4.5 | Beleuchtung | 45 |
| 1.4.6 | Materialeigenschaften | 46 |
| 1.4.7 | Texturen | 48 |
| 1.5 | RenderStates | 51 |
| 1.5.1 | Rasterizer State | 51 |

| | | |
|--------|--|-----|
| 1.5.2 | Depth-Stencil State | 53 |
| 1.5.3 | Blend State | 56 |
| 1.6 | Verschiedene Kameratypen | 59 |
| 1.7 | Texte anzeigen | 64 |
| 1.8 | Eingabegeräte | 65 |
| 1.8.1 | Tastatur | 66 |
| 1.8.2 | Maus | 67 |
| 1.8.3 | Gamepad | 68 |
| 1.9 | Sounds | 68 |
| 1.10 | DXUT gemeinsam nutzen | 69 |
| 1.10.1 | DXUT in einem gemeinsamen Verzeichnis verwenden | 69 |
| 1.10.2 | DXUT als Bibliothek kompilieren | 71 |
| 1.11 | Das Effekt-Framework | 71 |
| 2 | Basiswissen | 81 |
| 2.1 | Grundlagen der 3D-Mathematik | 81 |
| 2.1.1 | Rechnen mit Vektoren und Matrizen | 81 |
| 2.1.2 | Transformationsmatrizen | 96 |
| 2.1.3 | Beispiele | 98 |
| 2.2 | Die wichtigsten Anwendungen | 103 |
| 2.2.1 | Kamera | 103 |
| 2.2.2 | Bewegte Objekte | 105 |
| 2.2.3 | Geometrie | 109 |
| 2.2.4 | Picking | 111 |
| 3 | Spieleprogrammierung – natürlich objektorientiert | 119 |
| 3.1 | Verwendung der C++-Standardbibliothek | 119 |
| 3.1.1 | Namespaces | 121 |
| 3.1.2 | Warnungen deaktivieren | 122 |
| 3.1.3 | Collections | 122 |
| 3.2 | Kapselung wiederkehrenden Codes in Klassen | 122 |
| 3.3 | Effekt und Material | 123 |
| 3.3.1 | Die Klasse Material | 123 |
| 3.3.2 | Verwendung im Programm | 129 |
| 3.4 | Licht | 132 |
| 3.5 | Die abstrakte Basisklasse Renderable | 134 |
| 3.5.1 | Abstrakte Memberfunktionen | 135 |
| 3.6 | Entity – eine Klasse für Mesh-Objekte | 138 |
| 3.6.1 | Verwendung im Programm | 140 |

| | | |
|--------|---|-----|
| 3.7 | Szenenverwaltung | 142 |
| 3.7.1 | Die Klasse CSceneManager | 142 |
| 3.7.2 | Konstruktor und Destruktor | 143 |
| 3.7.3 | Hinzufügen und Entfernen von Objekten | 144 |
| 3.7.4 | Objekte erstellen und löschen | 145 |
| 3.7.5 | Aktualisieren | 146 |
| 3.7.6 | Rendern | 146 |
| 3.7.7 | Beleuchtung | 146 |
| 3.7.8 | Verwenden der SceneManager-Klasse | 147 |
| 3.7.9 | Programmcode | 148 |
| 3.8 | Die abstrakte Basisklasse Visual und eine neue Version der Entity-Klasse | 150 |
| 3.8.1 | Die Klasse Visual | 151 |
| 3.8.2 | Änderungen an der Klasse Material | 155 |
| 3.8.3 | Die Klasse Entity von Visual abgeleitet | 158 |
| 3.8.4 | Beispiel | 160 |
| 3.8.5 | Verwenden einer externen Textur | 160 |
| 3.9 | Effektverwaltung | 163 |
| 3.9.1 | Änderungen in der Klasse Material | 163 |
| 3.9.2 | Änderungen im SceneManager | 164 |
| 3.9.3 | Beispiel | 165 |
| 3.10 | Render States | 166 |
| 3.10.1 | Ergänzungen in der Klasse Visual | 166 |
| 3.10.2 | Alpha Blending | 169 |
| 3.10.3 | Depth-Buffer | 170 |
| 3.10.4 | Rasterizer State | 171 |
| 3.11 | Vervollständigen der Klasse Visual | 172 |
| 3.11.1 | Zugriff auf die Materialien | 172 |
| 3.11.2 | Sichtbarkeit und Auswählen | 175 |
| 3.11.3 | Draw Order | 176 |
| 3.12 | Hierarchisch organisierte Szenen | 178 |
| 3.12.1 | Die Klasse Node | 178 |
| 3.12.2 | Beispiel | 185 |
| 3.13 | Projektarchitektur | 187 |
| 3.13.1 | Engine-Dateien als Verweis zum Projekt hinzufügen | 187 |
| 3.13.2 | Die Engine als DLL kompilieren | 188 |
| 3.14 | Exkurs: Materialien mehrfach verwenden | 191 |
| 3.14.1 | Zuweisen eines Materials vor dem Create-Aufruf | 192 |
| 3.14.2 | Zuweisen eines Materials nach dem Create-Aufruf | 195 |

| | | |
|----------|--|-----|
| 4 | Physik | 199 |
| 4.1 | Translationsbewegungen | 199 |
| 4.1.1 | Weg, Zeit und Geschwindigkeit | 199 |
| 4.1.2 | Kraft, Masse und Beschleunigung | 200 |
| 4.1.3 | Energie und Impuls | 200 |
| 4.2 | Rotationsbewegungen | 202 |
| 4.2.1 | Winkelgeschwindigkeit | 202 |
| 4.2.2 | Drehmoment, Trägheitsmoment und Winkelbeschleunigung | 203 |
| 4.2.3 | Drehimpuls und Rotationsenergie | 203 |
| 4.3 | Programmcode | 204 |
| 4.3.1 | Verwendung im Spiel | 211 |
| 4.3.2 | Steuerung mit dem Gamepad | 217 |
| 4.4 | Harmonische Schwingung (Pendel) | 218 |
| 4.5 | Wellen | 219 |
| 5 | Punkte und Dreiecke | 223 |
| 5.1 | Vertexbuffer und Indexbuffer | 223 |
| 5.1.1 | Vertices | 223 |
| 5.1.2 | Indizes | 223 |
| 5.1.3 | Backface Culling | 224 |
| 5.1.4 | Vertexlayout | 224 |
| 5.2 | Eine Klasse für dynamisch erstellte Objekte | 224 |
| 5.2.1 | Vertextypen | 225 |
| 5.2.2 | Subsets | 230 |
| 5.3 | Templates | 231 |
| 5.4 | Der Programmcode | 231 |
| 5.5 | Einfache Beispiele | 238 |
| 5.5.1 | Dreieck | 238 |
| 5.5.2 | Rechteck | 240 |
| 5.5.3 | Würfel | 243 |
| 5.5.4 | Linien statt Dreiecke: Achsenkreuz | 247 |
| 5.6 | Die Schnittstelle IMesh | 249 |
| 5.7 | Kollisionserkennung | 250 |
| 5.7.1 | Bounding-Box | 250 |
| 5.7.2 | Bounding-Sphere | 252 |
| 5.7.3 | Dreiecksbasierte Kollisionserkennung | 252 |
| 5.7.4 | D3DXIntersectTri | 253 |
| 5.7.5 | Verwendung im Programm | 255 |

| | | |
|----------|--|------------|
| 5.7.6 | Kollisionskamera | 257 |
| 5.7.7 | Kollisionstest für hierarchisch organisierte Szenen | 262 |
| 5.7.8 | Octrees | 264 |
| 5.8 | Dateiformate | 272 |
| 5.8.1 | Verwenden des SDKMESH-Dateiformats mit eigenen Materialien für jedes Subset | 272 |
| 5.9 | Ein eigenes Dateiformat: Blender-Export | 277 |
| 5.10 | X-Import | 287 |
| 5.10.1 | DX9-Funktionen für das Laden von X-Dateien | 289 |
| 5.10.2 | X-Import »zu Fuß« | 289 |
| 5.11 | Fremdformate importieren | 301 |
| 6 | Techniken mit dynamisch erzeugten Objekten | 305 |
| 6.1 | Höhenfelder | 305 |
| 6.1.1 | Funktionsgraphen | 305 |
| 6.1.2 | Höhentexturen | 308 |
| 6.1.3 | Beispiel | 309 |
| 6.2 | Rotationskörper | 312 |
| 6.3 | Kugel | 315 |
| 6.4 | Billboards | 318 |
| 6.4.1 | Die Klasse TexturedQuad | 320 |
| 6.4.2 | Die Klasse Billboard | 321 |
| 6.4.3 | Verwendung im Programm | 323 |
| 6.5 | Zeichnen mit Linien | 323 |
| 6.6 | Skybox | 326 |
| 6.6.1 | Würfelförmige Skybox (Sky Cube) | 327 |
| 6.6.2 | Halbkugelförmiger Himmel (Skydome) | 335 |
| 6.7 | Die Sonne als Billboard | 340 |
| 6.7.1 | Die Klasse Himmelskoerper | 340 |
| 6.7.2 | Bahnbewegung | 342 |
| 6.7.3 | Beleuchtung | 343 |
| 7 | Grafische Benutzeroberflächen | 345 |
| 7.1 | GUI mit DXUT | 346 |
| 7.1.1 | Dialogfelder | 346 |
| 7.1.2 | Eigene Dialogfeldtypen ableiten | 349 |
| 7.1.3 | Steuerelemente | 353 |
| 7.1.4 | Textausgabe | 365 |
| 7.2 | GUI mit Windows Forms | 368 |
| 7.2.1 | Beispiel Quaternionen-Rechner | 370 |

| | | |
|----------|--|------------|
| 8 | Shader | 375 |
| 8.1 | Shader-Typen | 376 |
| | 8.1.1 Vertexshader | 376 |
| | 8.1.2 Pixelshader | 376 |
| | 8.1.3 Geometrieshader | 376 |
| 8.2 | Effektgruppen und Techniques | 377 |
| 8.3 | Globale Variablen | 377 |
| 8.4 | Strukturen für die Ein- und Ausgabe | 379 |
| 8.5 | Der Standardshader | 379 |
| 8.6 | Einfache Shader | 380 |
| | 8.6.1 Farbige Darstellung | 380 |
| | 8.6.2 Texturierte Darstellung | 382 |
| 8.7 | Beleuchtungsmodelle mit Vertexshadern | 385 |
| | 8.7.1 Einstellen der Lichtparameter | 385 |
| | 8.7.2 Richtungslicht | 386 |
| | 8.7.3 Glanzlichter | 389 |
| | 8.7.4 Punktlichtquellen | 393 |
| | 8.7.5 Scheinwerfer (Spotlight) | 396 |
| 8.8 | Beleuchtungsmodelle mit Pixelshadern | 399 |
| | 8.8.1 Scheinwerfer (Spotlight) | 400 |
| 8.9 | Effektvariablen gemeinsam nutzen | 402 |
| | 8.9.1 Shader-Includes | 402 |
| | 8.9.2 Verwendung im C++-Code | 405 |
| | 8.9.3 Verwendung im Beispielprogramm | 406 |
| | 8.9.4 Erweiterung des Szenenmanagers | 408 |
| | 8.9.5 Beleuchtung im Szenenmanager | 409 |
| 8.10 | Reflexion und Lichtbrechung | 413 |
| | 8.10.1 Reflexion | 414 |
| | 8.10.2 Lichtbrechung | 417 |
| | 8.10.3 Chromatische Dispersion (Prismeneffekt) | 421 |
| | 8.10.4 Verwendung im Programm | 423 |
| 8.11 | Partikeleffekte | 425 |
| | 8.11.1 Die Struktur Particle | 426 |
| | 8.11.2 Die Klasse ParticleSystem | 427 |
| | 8.11.3 Der Shader | 433 |
| | 8.11.4 Verwenden der Klasse | 436 |
| 8.12 | Rekursives Erzeugen von Geometrie | 437 |
| | 8.12.1 Rendern in Buffer (Stream Output) | 438 |

| | | |
|--------|--|------------|
| 8.12.2 | Der Shader | 439 |
| 8.12.3 | Die Klasse GSEntity | 443 |
| 8.13 | Tools | 449 |
| 8.13.1 | FX Composer | 449 |
| 8.13.2 | Das NVIDIA-SDK | 450 |
| 8.13.3 | Render Monkey | 451 |
| 9 | In Texturen hineinrendern | 453 |
| 9.1 | Render-To-Texture | 453 |
| 9.1.1 | Erzeugen der Rendertargets und des Viewports | 453 |
| 9.1.2 | Material | 459 |
| 9.1.3 | Rendern | 461 |
| 9.1.4 | Verwendung im Programm | 464 |
| 9.1.5 | Rendern als Overlay | 465 |
| 9.2 | Dynamische Cubemap für Reflexion und Brechung | 471 |
| 9.2.1 | Der Shader | 473 |
| 9.2.2 | Materialien | 477 |
| 9.2.3 | Rendern | 480 |
| 9.2.4 | Verwendung im Programm | 486 |
| 9.2.5 | Speichern von Cubemap-Texturen | 487 |
| 9.3 | Lightmaps | 493 |
| 9.3.1 | Der Shader | 494 |
| 9.3.2 | Ergänzungen in der Klasse Visual | 498 |
| 9.3.3 | Ergänzungen im Scenemanager | 499 |
| 9.3.4 | Ergänzungen in der Klasse Light | 502 |
| 9.3.5 | Verwendung im Programm | 503 |
| 10 | Postprocessing | 507 |
| 10.1 | Rendern in eine Textur mit einer Bildbearbeitungstechnik | 507 |
| 10.2 | Blur | 509 |
| 10.2.1 | Der Shader BlurHorizontal | 509 |
| 10.2.2 | Einstellen der Shaderparameter im Dialogfeld | 514 |
| 10.2.3 | Full-Screen-Rendern | 516 |
| 10.2.4 | Verwendung im Programm | 518 |
| 10.2.5 | Rendern in mehreren Durchgängen | 518 |
| 10.3 | Glow (Glühen) | 522 |
| 10.3.1 | Überlagern der Szene mit dem Glow-Effekt | 523 |
| 10.3.2 | Glow für Teile von Objekten | 525 |

| | | |
|--------|---|-----|
| 10.4 | Tiefenunschärfe | 525 |
| 10.4.1 | Ergänzungen im Scenemanager | 529 |
| 10.4.2 | Ergänzung in der Klasse TexturedMaterial | 531 |
| 10.4.3 | Verwendung im Programm. | 531 |
| 10.5 | Deferred Lighting | 533 |
| 10.5.1 | Rendern in ein Array von Texturen. | 533 |
| 10.5.2 | Der Shader | 534 |
| 10.5.3 | Verwendung im C++-Code | 542 |
| II | Animation | 545 |
| II.1 | Starre Körper | 545 |
| II.1.1 | Die Klasse Animation. | 545 |
| II.1.2 | Animierte Szenenobjekte. | 550 |
| II.1.3 | Animationen im Programmcode erstellen. | 554 |
| II.1.4 | Animationen in .x-Dateien. | 555 |
| II.2 | Skelettbasierte Animation | 563 |
| II.2.1 | Skelettbasierte Animationen in Dateien. | 563 |
| II.2.2 | Die Klasse AnimatedEntity | 565 |
| II.2.3 | Die Klasse CDXUTSDKMesh | 572 |
| II.2.4 | Skelettinformationen in .x-Dateien. | 579 |
| II.2.5 | Bone-Matrizen in einer Textur speichern. | 584 |
| 12 | Audio | 595 |
| 12.1 | XACT | 595 |
| 12.1.1 | Erstellen eines XACT-Projekts | 595 |
| 12.1.2 | Abspielen von Sounds | 599 |
| 12.1.3 | Runtime Parameter Controls. | 607 |
| 12.1.4 | 3D-Sound und Effekte | 609 |
| 12.1.5 | Benachrichtigungen | 615 |
| 12.2 | XAudio2 | 620 |
| 13 | Multiplayer-Spiele | 627 |
| 13.1 | Netzwerkspiele | 627 |
| 13.1.1 | Netzwerkcode selbst schreiben mit Sockets. | 628 |
| 14 | Landschaften und Wetter | 649 |
| 14.1 | Algorithmen | 649 |
| 14.1.1 | Noise | 649 |
| 14.1.2 | Perlin-Noise | 650 |

| | | |
|--------|--|-----|
| 14.1.3 | Die Klasse ProceduralMaterial | 655 |
| 14.1.4 | FBM (Fractional Brownian Motion) | 658 |
| 14.2 | Terrain | 664 |
| 14.2.1 | Die Klasse HöhenfeldGPU | 664 |
| 14.3 | Pflanzen | 673 |
| 14.3.1 | Fraktale Pflanzen aus Linien | 674 |
| 14.3.2 | Bäume | 677 |
| 14.4 | Wasser | 689 |
| 14.4.1 | Die Klasse Watermesh | 689 |
| 14.4.2 | Reflexion und Lichtbrechung | 700 |
| 14.4.3 | Dynamische Cubemap | 701 |
| 14.5 | Regen und Schnee | 702 |
| 15 | Instancing | 705 |
| 15.1 | Gras | 705 |
| 15.1.1 | Die Klasse Group | 706 |
| 15.1.2 | Der Shader | 712 |
| 15.2 | Rendern von Charaktermodellen mit Instancing | 713 |
| 15.3 | Ein komplexeres Beispiel | 723 |
| 16 | Scripting | 725 |
| 16.1 | Skriptsprachen | 726 |
| 16.1.1 | C# | 726 |
| 16.1.2 | Python | 740 |
| 16.1.3 | Lua | 749 |
| 17 | Was ist neu an DirectX 11? | 757 |
| 17.1 | Tessellation | 757 |
| 17.1.1 | Terrainerzeugung | 759 |
| 17.2 | Computeshader | 765 |
| 17.2.1 | Berechnungen | 766 |
| 17.2.2 | Rendern von Grafikdaten | 773 |
| 17.2.3 | Rendern in Buffer | 776 |
| 17.2.4 | Postprocessing mit dem Computeshader | 783 |
| A | Anhang | 787 |
| A.1 | Einstellungen für Projekte mit .Net. | 787 |
| | Stichwortverzeichnis | 789 |