FAQ Vektoria-Software

Vektoria-Version:	V15
Stand:	06.02.2017

Autor:	Prof. Dr. Tobias Breiner
Ort:	Pfungstadt
Datum:	06.02.2017

Inhalt

Was ist Vektoria?

Vektoria ist eine umfangreiche C++ -Klassenbibliothek inklusive Szenegraph für interaktive dreidimensionale Anwendungen wie z.B. Games oder Fabrik-, Fahr- und Flugsimulatoren. Neben dem umfangreichen 3D-Szenegraf sind auch 3D Sound, 2D-Elemente, die Einbindung vieler gängiger Input- und Output-Game-Devices, Partikelsysteme und stereoskopische Sichtsysteme harmonisch integriert.

Was ist Vektoria nicht?

Vektoria ist kein Modellierungsprogramm und keine Point-and-Klick-Game-Engine!

Auf welcher Grafik-API basiert Vektoria?

Vektoria greift auf DirectX11und dem Shadermodell 5.0 zurück. Man kann bei veralteten Grafikkarten auch auf Shadermodell 4.1 downgraden. Schnittstellen zu einer potentiellen alternativen OpenGL-Anbindung und einem Echtzeitraytracer sind im internen Klassendiagramm schon angelegt.

Welche Hardware wird benötigt, damit Vektoria läuft?

Vektoria benötigt einen Computer, der DirectX11 mit Shadermodell 5.0 (bzw. 4.1 bei der abgespeckten Ausführung) aufwärts unterstützt. Dies ist für Computer und Laptops normalerweise der Fall, wenn sie nach dem Jahr 2008 gekauft wurden. Schnittstellen zu anderen Betriebssystemen (insb. Linux) sind in Planung und in der Software schon strukturell angelegt.

Welche Software wird benötigt, damit Vektoria läuft?

Vektoria benötigt das DirectX11 SDK (Achtung, das Software Development Kit und nicht das normale DirectX 11!), sowie mind. Microsoft Visual C++ 2015.

Welche grundsätzlichen Vorteile bietet Vektoria?

Mit Vektoria kann man im Gegensatz zu anderen Szenegrafen schnell und einfach gute Ergebnisse erzielen, sofern man in C++-Programmierung einigermaßen bewandert ist. Vektoria bietet zudem einige Features, die von anderen Szenegrafen bzw. GameEngines nicht angeboten werden. Durch die einheitliche klar objektorientierte Struktur von Vektoria wird ein Programmierer zudem schnell ein komplettes Verständnis der Engine erreichen.

Was kann Vektoria?

Hier sind einige der besonderen Fähigkeiten von Vektoria (für detailliertere Informationen siehe angehängtes Klassendiagramm):

- Komplexe Hierarchien, die auch wieder zusammenwachsen können (Eine Geometrie kann z.B. dadurch mehrfach an verschiedenen Positionen angezeigt werden, ohne dass sie mehrfach angelegt werden muss), (Softwareklassen: CRoot, CCamera, CHardware, CFrame, CViewport, CPlacement, CGeo, CEmitter)
- Smart Allocation (Vereinigt die Vorteile von Geschwindigkeit und wenig Speicherplatzverbrauch). Diese einzigartige Technologie nutzt den Umstand aus, dass die Anzahl von angehängten Knotenobjekten statistisch gemittelt exponentiell steigt.
  (Die Smart Allocation einer Klasse befindet sich ihrer sogenannten Pluralklasse, welche identisch mit einem hinten angestellten -s lautet)
- Netzwerkunterstützung für TCP/IP und UDP
  (Klassen CUDPClient, CUDPServer, CTCPClient, CTCPServer, CNetworkBase)
- Flexibles und schnelles Verteiltes Echtzeitrendering
  (Klassen: CRenderDistributed, CRenderClient)
- Licht (Parallellicht, Punktlicht, Scheinwerferlicht, Ambientes Licht, Glow-Light) & Schatten (bislang nur für Scheinwerferlichter)
  (Klassen: CLight, CLightParallel, CLightRadial, CLightSpot)
- Mehrere frei konfigurierbare Viewports (für Split-Screens, In-Screens, etc.)
  (Klasse: CViewport), Mirror-Vieports(z.B. für Rückspiegel)
- Handling mehrerer frei konfigurierbarer Fenster
  (Klasse: CFrame)
- Verschiedene Viewportstile (Sepia, Colorizing, Cartoon-Shading, u.a)
  (Klasse: CViewport)
- 3D Sound (inklusive Doppler-Berechnung)
  (Klasse: CAudio)
- Extrem leichte Materialerstellung und Textureinbindung (u.a. Imagemapping, Bumpmapping, Glowmapping, Reflexionmapping, Parallax-Occlution Mapping, Environmental Mapping, Sky-Mapping, Subsurface Scattering, Fresnel-Berechnung)
  (Klasse: CMaterial, CTexture, CImage)
- Level of Details (auch für Sound und Emitter)
  (Klasse: CPlacement, CAudio)

- Billboards, Achsenfixierte Billboards
  (Klasse: CPlacement)
- Appendages (dadurch kann man Multibillboards, Multiskydomes und Criss-Cross-Billboards erzeugen, um überzeugende Nebel- und Rauchschwaden, Bäume und Himmel zu generieren)
  (Klasse: CPlacement)
- Skydomes, Skyellipsoide, Multiskydomes, Skyboxes
  (Klassen : CPlacement, CGeoDome, CGeoSkybox)
- Transparenz, sowohl via Alphakanal oder per Chromakeying
  (Urheber: Tobias Breiner, Klassen: CMaterial, CDirectX)
- Unterstützung von Eingabegeräten (Tastatur, Maus, Joystick, Gamepad, Lenkrad, Motorrad-Controller, Flugcontroller, Tanzmatten, etc.)
  (Klassen: CDevice, CDeviceKeyboard, CDeviceGameController, CDeviceMouse, CDeviceCursor, CDIGameController, CDIMouse, CDIKeyboard)
- Umfangreiche Mathematikbibliothek für 3D-Grafik, unter anderem auch für homogene Vektoren und Matrizen, Quaternionen, Pyramidenstümpfe, Ebenen, Quader und Strahlen)
  (Klassen: CHVector, CHMat, CQuaternion, CBoundingBox, CPlane, CRay, CUtil)
- Unterstützung aller gängiger Bildformate (z.B. JPG, JPEG, GIF, BMP, TGA, TIF, PNG) in verschiedenen Auflösungen, Bittiefen, mit und ohne Alphakanal)
  (Klassen: CImage, CDirectX)
- Unterstützung verschiedener Farbsysteme (RGB, HSV, CMY, CMYK)
  (Klassen: CColor)
- Geometrieimporter für X3D, Blender, Maya, 3D Studio Max und Wavefront
  (Klassen: CImporter…)
- 2D-Overlay-Sprites und 2D-Backgrounds
  (Klassen: CSprite, COverlay, CBackground)
- 2D-Overlay-Text und Billboard-Text-Labels
  (Klassen: CWribel, CWritingChar, CWriting, CWritingFont)
- Hierarchie von Overlay-Sprites und Text für 2D-GUIs
  (Klassen: CSprite, CDirectX)
- Bibliothek vorgefertigter parametrisierbarer Geometrien (Kugel, Ellipsoid, Flächen, Ikosaeder, Zylinder, Kegel, Röhren, Quader, Tortenstückformen, Utah-Teapots, Quads, Grids, u.a.)
  (Klassen: Alle Klassen, die mit CGeo anfangen)
- Extruding und Sweep-Körper

(Klassen: CGeoSweep)

- Hierarchitekturmodellierung für die schnelle Erzeugung von 3D-Gebäuden und -Städten, diese neuartige Methode beinhaltet: 3D-Wände mit Giebeln, Fenstern (alle gängigen Fensterformen), Kuppeln, und Dächer, gebogene Wände, objektorientierte Gebäudeerstellung
  (Klassen: CGeoWindow, CGeoWall, CGeoWing)
- Bézier-Patches und - Splines
  (Klassen: CHVector, CGeoBezierPatches)
- Prozedurale Erzeugung eigener Geometrien
  (Klassen: GGeo, CGeoTriangleStrip, CGeoTriangleList, CGeoTriangleTable, …)
- Triangle Lists, Triangle Strips und Triangle Tables
  (Klassen: CGeoTriangleStrip, CGeoTriangleList, CGeoTriangleTable)
- Punktwolken und Linienlisten
  (Klassen: CGeoPolintList, CGeoLineList)
- Tapering, Twisting und Bending entlang beliebiger kartesischer Achsen
  (Klasse CGeo)
- Wavering und Rippling zur schnellen Modellierung spezieller geometrischer Körper
  (Klasse: CGeo)
- Triangle Subdivition (auch selektiv entlang beliebiger kartesischer Achsen)
  (Klasse CGeo)
- Mappingfunktionen (planar, zylindisch, kubisch, orthografisch, biorthografisch, etc.)
  (Klasse CGeo)
- Umfangreiches und flexibles Partikelsystem (noch nicht 100% fertig)
  (Klasse CEmitter)
- Switches zum einfachen An- und Ausschalten für alle wichtigen Knotenobjekte
  (CSprite, CPlacement, CCamera, CLight, CGeo, CViewport, CFrame …)
- Ansätze zum verteilten Rendering via LAN (noch in Arbeit)
  (Alle Klassen, die mit CDisFac anfangen)
- Postprocessing
  (Klassen: CDirectX11FP)
- Logger zum Debuggen
  (Klassen CLog)

- Genauer Timer (schnittstelle zu Windows- und Linux-Betriebssystemen) mit Frame Rate-Berechnung
  (Klassen CTimer)
- Kryptographisch codierter Splash-Screen mit So
- und (am Anfang jedes mit Vektoria erstellten Programmes wird automatisch das Vektoria-Logo und der Vektoria-Sound angezeigt). Jede Modifikation seitens Hacker oder Raubkopierer zerstört automatisch die Verwendbarkeit der Vektoria-Software.
  (Klassen CSplash)
- Dynamisches Shaderladen
  (Klassen: CDirectX, CShaderCache)
- Anbindung an die OculusRift
  (Klassen: CDeviceOculusRift)
- Vektorianische Datentypen- und Abkürzungsnotation (erlaubt es durch genormte Namenskonventionen im Source-Code, trotz großer Komplexität die Übersicht zu behalten)
  (Klassen: alle)
- Optionales State Sorting (erhöht die Frame Rate für statische Hierarchien), das State Sorting ist reversibel und wird einzigartig nur von Vektoria angeboten, unterhalb von Placements werden alle Geometrien, welche ein gemeinsames Material teilen, zu einer einzigen Geometrie vereinigt.
  (Klasse: CPlacement, CGeo)
- Spezielle Bounding Volume Hierarchy aus rekursiv hierarchisch ineinandergreifenden OBBs und AABBs für ein besonders schnelles Frustum Culling, Picking und Kollisionserkennung.
  (Klassen: CBoundingBox, COctree, CNode3D, CNode)
- Umfangreicher Forward-Plus-Shader auf DirectX11-Basis mit Postprocessing
  (Klassen: CDirectX11FP, CForwardPlusRenderer, CLightUtil, CMaterialUtil, CParticleRenderer, CPostBloom, CPostDoF, CPostGlow, CPostGodRays, CPostProcessor, CPostProcessorStep, RSMRenderer)
- Physik (Spring-Mass-Damper-Systeme, Gravitation, Luftwiderstand, spezifisches Gewicht, magnetische und elektrische Felder
  (Klassen:CScenePhysics, CPlacementPhysics)
- Umfangreiche Partikelsysteme
  (Klassen:CEmitter, CEmitterParticles)

- vordefinierte Emitter für Feuer, Nebel, Funken, Rauch, Hitzeflimmern, Regen, Schnee, etc., vordefinierte Emittermaterialien wie glühende Asche oder Duschkopfoberfläche
  (Klassen:CEmitter, CEmitterParticles, CMaterial)
- Umfangreiche Lade- und Speichermöglichkeiten für den gesamten Szenegrafen oder nur für Teile dessen oder einzelne Knotenobjekte oder Hardwarekonfigurationen
  (Klassen: Alle)
- Wetter-Engine für Wind-, Sturm-, Regen, Schnee-, Sonnenpositionsberechnung für jeden Längen- und Breitengrad
  (Klassen:SSceneWeather)

Wann wurde Vektoria erstellt?

Die ersten Gedanken und Programmierfragmente zu Vektoria stammen aus dem Jahre 2005 (Urheber: Tobias Breiner). Die Programmierarbeit startete ab Februar 2007 (Urheber: Tobias Breiner). Die erste Alphaversion kam im September 2011 heraus (Urheber: Tobias Breiner, Florian Schnell). Seitdem wurde Vektoria ständig weiterentwickelt.

Wie werden die Versionen von Vektoria gekennzeichnet?

Die Versionen werden einfach fortlaufend mit einem vorangestellten „V“ durchnummeriert.

Vektoria V1 (September 2011), erste öffentliche Version, Objekthierarchie aus 3D-Primitiven

Vektoria V2 (November 2011), u.a. mit Texturierung, Image, Glow- und Bump-Mapping, Sound)

Vektoria V3 (Dezember 2011, u.a. mit Schnelloperatoren, Billboards, Transparenz, Unterstützung von TriangleLists und Triangle Strips, LoDs, 3D Sound mit Dopplereffekt in Stereo

Vektoria V4 (Januar 2011), u.a. mit X3D- und Blender-Importer, Unterstützung von Input-Devices, Parallellichtern, Multibillboards, Skydomes und Skyboxes

Vektoria V5 (Anfang Mai 2012), u.a. mit frei konfigurierbaren multiplen Frames und Viewports, Stereoskopie, Punktlichtern und Scheinwerflichtern, Schatten, optionaler

Nullrenderer zum Debugging, vereinfachte Initialisierungsroutinen für Lichter und Geometrieen, verschiedenen Viewportstilen wie Sepia, Cartoon-Shading oder Outlining), Sweeping und Geometrische Primitive

Vektoria V6 (Ende Mai 2012), u.a. mit großer Frame-Rate-Beschleunigung der Release-Version, Fullscreen-Modus, echten Szenen-Handling, GeoTubes (gerade und gebogen), GeoCones, GeoCylinders, 2D-Overlays und Backgrounds, Input- und Output-Devices

Vektoria V7 (Juni 2012), u.a. mit Feder-Masse-Dämpfer-Systeme für Tissue und Soft-Object-Animation, Placement-Fixierung von statischen Objekten zur Beschleunigung des Szenegrafen, Cursor-Device für absolute Mauskoordinaten

Vektoria V8 (Dezember 2012), u.a. mit Downgrade-Möglichkeit auf Shadermodell 4.1, Architektur-Geometrien (Wände, Dächer, Kuppeln, Fensterhiearchien) Möglichkeit von Multiskydomes, Bounding-Boxen, Quaternionen, Overlay-Hierarchien für GUIs und vielen kleineren Verbesserungen

Vektoria V9 (Juli 2013), u.a. mit Modellierungsfunktionen (Tapering, Twisting, Bending, Rippeling, Triangle Subdivition, Magnets, etc.), Mappingfunktionen, Sub-Funktionen, um Objekte wieder von einer Hierarchie abzuhängen, Möglichkeit beliebig viele Objekte an ein Placement oder eine Hierarchie anzuhängen

Vektoria V10 (Dezember 2013), u.a. mit Architekturerzeugungsklassen (Wall, Windows und Wings), Parallax Occlution Mapping, Environmental und Sky-Mapping, Klassen für die Ebenen- und Strahlberechnung, Axenfixierte Billboards, optionales State Sorting zur Beschleunigung, Umstellung auf Visual Studio 2013

Vektoria V11 (April 2014), u.a. mit Beschleunigung beim Shader-Laden, Subsurface-Scattering, Switches für alle Knotenobjekte, 2D-Text und Text-Labels, Partikelsysteme.

Vektoria V12 (August 2014), u.a. mit Picking, Environmental Mapping, erweitertes POM, Sky-, Sprite- und BRDf-Texturen und einer kompletten Code-Modularisierung.

Vektoria V13 (Februar 2015), u.a. mit Forward-Plus-Renderer, BVH-Frustum-Culling, Postprocessing, Blurring, Godrays, Layer für Overlays, erweitertem Picking. Erweiterter Kollisionserkennung und vielen kleineren Verbesserungen.

Vektoria V14 (März 2016), u.a. mit Verteiltem Rendering, PointLists, LineLists, Bézier-Splines, TriangleTables, Bézier-Patches, kubischen Environmental Mapping,

Materialderivatbildung, Fresnel-Faktor, Feinjustierung von POS und Subsurfacescattering, pixelgenaue Steuerungsmöglichkeit von Refraktion, Reflexion, Highlighting und Color-Shift, verbessterter OBJ-Importer, Materialbibliothek, Neue genormte Renderschnittstelle, ambientem Licht in der Szene, automatisierter Ladebalken, Laden und Speichern von Konfigurationen, umfangreichen Debugging und Log-File-Möglichkeiten, einfachen Netzwerkfähigkeiten, allgemeine Renderbeschleunigung, uvm.

Vektoria V15 (Februar 2017), u.a. mit umfangreichen Partikelsystemen, vordefinierten Emittern für Feuer, Rauch, Wasser, etc., Physik, Wetterengine mit Sturm-, Wind-, Nebel, Schnee und Regenberechnung, Himmelsberechnung für jeden Breiten- und Längengrad, verbesserte Instanziierung, verbesserter OBJ-Importer, Loading und Saving für alle Knotenobjekte, Intellisense-Nachrichten, erweiterte Emitter-, Font- und Materialbibliothek, uvm.

Welche spezielle Mathematik bietet Vektoria?

- Homogene Vektoren
- Homogene Matrizen
- Quaternionen
- Mathematik der Ebenengleichung
- Mathematik für Strahlen und Geraden
- Frustum-Berechnungen
- Bounding-Box-Mathematik
- Zufallsgrafen
- Utility-Bibliothek für einfache Mathematik

Welche Knotenobjekte gibt es in Vektoria?

Mit den Knotenobjekten lässt sich eine komplexe Objekthierarchie aufbauen. Für jedes Knotenobjekt existiert eine eigene Klasse. Es existieren derzeit folgende Knotenobjekte:

- CRoot
- CScene
- CPlacement
- CGeo
    - CPointList

    - CLineList
    - CTriangleList
        - CGeoCube
        - CGeoIkosaeder
        - CGeoTetraeder
        - CGeoWall
        - CGeoWindow
        - CGeoWing
    - CTriangleStrip
        - CGeoQuad
        - CGeoDome
        - CGeoShere
        - CGeoEllipsoid
        - CGeoTube
        - CGeoSlice
        - CGeoSweep
    - CTriangleTable
        - CGeoUtahTeapot
- CMaterial
- CTexture
- CImage
- CEmitter
- CSound
- CLight
    - CLightParallel
    - CLightSpot
    - CLightRadial
- CCamera
- CHardware
- CFrame
- CViewport
- CDevice
    - CGameContoller
    - CKeyboard
    - CMouse
    - CDeviceCursor

- COverlay
- CBackground
- CWriting
- CWribel

Was ist eine Root?

Root (CRoot) ist die Wurzel der gesamten Knotenobjekthierarchie. Es gibt nur eine Wurzel. An eine Root können verschiedene Szenen (CScene) oder Computer (CHardware) angehangen werden. An eine Root müssen alle Materialien (CMaterial) angemeldet werden, die irgendwo im Szenegraf verwendet werden.

Was ist eine Scene?

Scene (CScene) stellt eine virtuelle Szene dar. Es können haptische oder optische Szenen erstellt werden. Eine Szene ist eine Konfiguration verschiedener Objekte z.B. für ein Game-Level, indem man verschiedene Szenen an eine Root anhängen kann, kann man auch z.B. zwischen verschiedenen Game-Level hin- und herschalten.

An eine Szene können beliebig viele Placements (CPlacement) angehangen werden. Auch können verschiedene Parallellichter (CParallelLight) an eine optische Szene angehangen werden, welche die gesamte Szene dann mit Licht durchfluten.

Was ist ein Placement?

Ein Placement (CPlacement) ist eine räumliche Platzierung von Objekten. Die platzierbaren Objekte können 3D-Klänge (CSound), Geometrien (CGeo), Punktlicher (CPointLight) und Scheinwerfer (CSpotLight) sein. An ein Placement können auch wiederum beliebig viele weitere Placements oder andere platzierbare Objekte angehangen werden. Es können sogar – anders als in den meisten anderen Szenegrafen – ein und dasselbe Placements gleichzeitig das Kind von mehreren anderen Vaterplacements sein, so dass die Äste der Knotenobjekthierarchie auch wieder zusammenwachsen können.

In einem Placement ist eine homogene Matrix versteckt, mit der die Translation, Skalierung und Rotation im Raum geschieht.

Ein Placement kann an- und ausgeschaltet werden, es kann auch mit Level of Details versehen werden, so dass es nur dann aktiv ist, wenn es in einem bestimmten Abstand zur Kamera liegt.

Es gibt spezielle Placements:

-**Directing Placements** zeigen immer in eine spezielle Richtung, z.B. zu einem Raumpunkt oder einem anderen Placement, Directing Placements kann man z.B. verwenden, damit eine darunter liegende Kameras immer ein anderes Objekt fokussiert.

-**Sky Placements** wandern immer mit der Kamera, die sie beobachten ohne ihre Rotation nachzuempfinden. Mit Sky Placements kann man Skydomes, bzw. Skyboxes für den Himmel erzeugen, indem man eine entsprechende nach innen gerichtete Geometrie an ein Sky Placement anhängt. Sky-Placements gibt es nur in Vektoria. Einige andere Szenegrafen bieten extra Klassen für Sky-Domes oder Sky-Boxes, die allerdings nicht so flexibel sind wie Sky-Placements, denn an Sky-Placements lassen sich beliebige Geometrien anhängen. Mit Appendages (kommt später) lassen sich sogar ganze Objekthierarchien an Sky-Placments anhängen, z.B. um mehrschichtige Skydomes zu erzeugen, die besonders realistisch aussehen.

-**Ground Placements** wandern immer mit der Kamera, die sie beobachten und vollführen zusätzlich ihre Rotation nach. Somit kann man Objekte erzeugen, die immer an einer Stelle des Bildschirms gerendert werden.

-**Billboard Placements** wenden sich stets zur Kamera hin. Billboard-Placements werden meist zusammen mit einem angehängten GeoQuad mit Alphatextur verwendet.

-**Appendage Placements** hängen stets am übergeordneten Sky-, Ground- oder Billboard-Placement fest, mit ihnen kann man z.B. Multibilboards (mehrschichtige Billboards, die z.B. für Volumennebel oder Niederschlagschichten), Criss-Cross-Billboards (z.B. für Bäume) oder Multiskydomes erzeugen. Appendage-Placements gibt es nur in Vektoria. Mit anderen Szenegrafen lassen sich daher diese Effekte nicht erzeugen.

Was ist ein Geo?

Geos beinhalten die dreidimensionale Oberflächengeometrie in Form von Dreiecksnetzen. Andere Polygone außer Dreiecke werden nicht angeboten. Geos können entweder als TriangleStrips (CTriangleStrips) oder TriangleLists (CTriangleLists) auftreten. Beide bieten ihre spezifischen Vor- und Nachteile:

TriangleStrips können von der Grafikkarte etwas performanter verarbeitet werden als TriangleLists, sind jedoch nur für kurvige nichtmanifolde Objekte geeignet und

erfordern bei ihrer Erzeugung eine größere Vorsicht beim Programmierer. Es gibt für häufig benötigte Geometrieformen verschiedene vorgefertigte TriangleStrips: GeoQuad (=Rechteck), GeoSphere (Kugel), GeoEllipsoid (=3D-Ellipsoid), GeoCone (Kegel), GeoCylinder (Zylinder), GepSweep (Sweeping und Extruding-Körper)

TriangleLists können – anders als TriangleStrips – Geometrieen mit scharfen Ecken und Kanten erzeugen. Sie sind bei kurvigen nichtmanifolden Objekten allerdings weniger performant (ca. Faktor 50%). Es gibt für häufig benötigte Geometrieformen verschiedene vorgefertigte TriangleLists: GeoCube (=Quader, bei gleichen X,Y;Z-Parametrisierung ein Würfel), GeoTetraeder (=Tetraeder, auch verzerrte Tetraeder sind erzeugbar), GeoIkosaeder (=Ikosaeder)

TriangleTables vereinigen weitestgehend die Vorteile von TriangleStrips und TriangleLists. Sie sind gleichbedeutend mit Indexed Face Sets.

Was ist ein Material?

Ein Material (CMaterial) ist eine Oberflächenbeschreibung für eine Geometrie (CGeo). Sie können verschiedene Texturen (CTextures) enthalten.

Was ist eine Textur?

Eine Textur (CTexture) enthält für gewöhnlich ein Image und kann man sich wie eine Tapete vorstellen, welches auf eine Oberfläche aufgebracht wird. Es gibt Image-Texturen (welche die reine Farbgebung beinhalten), Bump-Map-Texturen (für das Oberflächenfeinrelief), Glow-Texturen (für den selbst-leuchtenden, nicht beleuchteten Teil eines Objektes) und Reflexion-Texturen (für die Stärke des Highlights bzw. der Reflexion bei spiegelnden Objekten)

Was ist ein Image?

Eine Image (CImage) ist ein Pixelbild. Man kann ein Image durch die Pfadangabe (inklusive Suffix) einer Bilddatei initialisieren.

Was ist ein Emitter?

Ein Emitter (CEmitter) ist eine Quelle für Partikel, die flexibel parametrisiert werden kann. Mit Emitter können zahlreiche Partikeleffekte erzeugt werden, wie Feuer, Rauch, Regen, Schnee, Graupel, Blitze, Vulkane, Funken, etc.

Was ist ein Audio?

Ein Audio (CAudio) ist ein Klang, der entweder einmalig abgespielt werden kann oder als Endlos-Loop. Sounds initialisiert man durch Angabe eines Pfades zu einer

WAV oder MP3-Datei. Man kann Sounds als musikalische Hintergrundsuntermalung (Ambient Audio) oder als 3D-Audio verwenden, bei letzterem muss man den Sound an ein Placement anhängen. Bei 3D-Audio wird der Stereoeffekt und der Dopplereffekt (Ton erklingt heller, wenn man auf ihn zusteuert) automatisch mitberechnet.

Was ist ein Light?

Ein Light stellt eine Lichtquelle dar. Es gibt Parallellichter für das Sonnen- , Mond und Sternenlicht (CParallelLight), Punktlichter für punktförmige Leuchtkörper (CPointLight) und SpotLights für Scheinwerfer (CSpotLight). Lichter können in einer bestimmten Farbe leuchten, die Defaultfarbe ist weiß.

Was ist eine Camera?

Durch eine Kamera (CCamera) kann man eine Szene aus einem bestimmten Blickwinkel betrachten. Kameras positioniert man – genauso wie z.B. Geometrien oder 3D Sounds – durch Placements. Eine Kamera hat einen Öffnungswinkel, eine Near- und eine Far-Clipping-Plane.

Was ist ein Frame?

Ein Frame (CFrame) ist ein Fensterkanal, welches auf eine Basisgrafik-API zurückgreift. Zurzeit werden als Basisgrafik-API DirectX11 und der NullRenderer angeboten. Ein Frame braucht immer ein Handle zu einem Window, in dem es angezeigt wird. In einem Frame können verschiedene Viewports angezeigt werden. An ein Frame können verschiedene Devices angehangen werden.

Was ist ein Viewport?

Ein Viewport (CViewport) ist ein Sichtausschnitt eines RenderFrames, Der Sichtausschnitt muss immer mit einer Kamera verbunden sein, er zeigt dann das an, was die Kamera gerade sieht. Ein Viewport kann verschiedene Stile haben, z.B. Normal, Sepia, Schwarzweiß, Cartoon-Anmutung, Konturierung, etc.), die Stile sind miteinander kombinierbar (z.B. Schwarzweiße Cartoon-Anmutung mit Konturierung).Es können verschiedene Viewports nebeneinander (z.B, für Split-Screens oder übereinander (z.B. für In-Screens) angezeigt werden.

Was ist ein Device?

Ein Device (CDevice) ist ein Aus- bzw. Eingabegerät, es werden folgende Geräte angeboten:

- Maus (CDeviceMouse)
- Cursor (CDeviceCursor)
- Tastatur (CDeviceKeyboard)
- GamePad, JoyStick oder Lenkrad (CDeviceGameController)
- Oculus Rift (CDeviceOculusRift)

Was ist ein Overlay?

Ein Overlay (COverlay) ist ein 2D-Sprite welches immer im Vordergrund eines Viewports zu sehen ist. Es muss daher immer an ein Viewport angehängt werden und bekommt ein Image zugewiesen. Overlays können auch wiederum an andere Overlays angehängt werden, um Overlay-Hierarchien zu erstellen.

Was ist ein Background?

Ein Background (CBackground) ist ein 2D-Sprite welches immer im Hintergrund einer Szene zu sehen ist. Es muss daher immer an ein Viewport angehängt werden und bekommt ein Image zugewiesen. Backgrounds sind somit sozusagen das Gegenteil von Overlays

Was ist ein Writing?

Ein Writing (CWriting) ist ein spezielles 2D-Sprite welches immer im Vordergrund einer Szene zu sehen ist. Mit ihm lassen sich beliebige Zeichenketten auf den Bildschirm schreiben. Es muss daher immer an ein Viewport angehängt werden und bekommt ein TextFont (CWritingFont) zugewiesen.

Was ist ein Wribel?

Ein Wribel (CWribel, Zusammensetzung aus Write + Label) ist ein spezielles 2D-Billboard. Mit ihm lassen sich beliebige Zeichenketten in den 3D-Raum stellen. So können Objekte mit einem Textlabel versehen werden. Es muss daher immer an ein Placement (CPlacement) angehangen werden und bekommt ein TextFont (CWritingFont) zugewiesen.

Was bedeutet die vektorianische Abkürzungsnotation?

Die Klassen in Vektoria, die Knotenobjekte repräsentieren, fangen (fast) alle mit einem anderen Buchstaben an. Es gibt zurzeit folgende Buchstaben:

A: CAudio

B: CBackground

C: CCamera

D: CDevice

E: CEmitter

F: CFrame

G: CGeo

H: CHardware

I: CImage

J: *(frei)*

K: CKeyframe *(angedacht)*

L: CLight *(=> LL: LightParallel, LP: LightRadial, LS: LightSpot)*

M: CMaterial

N: CNode

O: COverlay

P: CPlacement

Q: *(frei)*

R: CRoot

S: CScene

T: CTimer

U: CUnion *(angedacht)*

V: CViewport

W: CWriting, CWritingChar, CWribel

X: *(frei)*

Y: *(frei)*

Z: *(frei)*

Die Buchstabenunterschiedlichkeit in Vektoria ist kein Zufall. Damit lässt sich gezielt Schreibarbeit durch leicht zu merkende mnemotechnische Abkürzungen vermindern. Zum Beispiel können die Präfixe für instanziierte Variablen in Vektoria immer mit einem „z“ und dem nachfolgenden Anfangsbuchstaben des Knotenobjektes abgekürzt werden:

za: CAudio; zas: CAudios

zb: CBackground; zbs: CBackgrounds

zc: CCamera; zcs: CCameras

zd: CDevice; zds: CDevices

zf: CFrame; zfs: CFrames

etc.

Für die mathematischen Klassen gibt es auch Präfixe:

v: CHVector; vs: CVHVectors

m: CHMat; ms: CVHMats

q: CQuaternion; qs: CQuaternions

r: CRay; rs: CRays