Bestandteile einer Engine

Inhaltsverzeichnis

Low-Level-System

- Das Low-Level-System dient zur Speicherzuweisung, Speicherbereinigung

- Beinhaltet grundlegende Datenstrukturen und Funktionen wie beispielsweise eine Dateiverwaltung
  - Dateiverwaltung übernimmt allgemein die Steuerung und Überwachung von Daten
    - Angefangen bei der Eingabe über die interne Verarbeitung oder die Veränderung mithilfe von Programmen bis hin zur Ausgabe und Speicherung
    - Zur Datenverwaltung rechnet man auch Aufgaben wie Erfassung und Organisation von Daten, Beschriftung von Datenträgern, regelmäßige Datensicherung, Archivierung und das Löschen nicht mehr benötigter Dateien
    - (Savegames)

Mathematisches System

- System für Mathematik
  - Bildet die mathematische Funktionalität ab, die innerhalb der Engine verwendet werden kann
  - Wird beispielsweise für Vectoralgebra und Matrixalgebra benötigt
  - Die Einschränkungen und die Features dieses Systems müssen genau gegeneinander abgewogen werden
    - Ein größere Auswahl an mathematischen Funktionen und Features kann sich nachteilig auf die Performance der Engine auswirken

- Stellt Funktionen für die Berechnung von trigonometrischen Funktionen bereit
  - Beispielsweise Sinus, Cosinus und Tangens
  - Stellt ebenfalls Vektoren, Matrizen und Quaternionen bereit

Objektsystem

- Das Objektsystem stellt verschiedene automatische Dienste für die Entwickler bereit
- Dient zur Verwaltung der Objekte
  - Zum Beispiel das Abrufen des Typs eines Objektes oder einer Variable zur Laufzeit
- Dient dem Referenz-Management
- Bietet die Möglichkeit Objekte mit ihrem kompletten Zustand zu Speichern und zu Laden

Renderer

- Ist für Zeichnung der Objekte zuständig
  - Zum Beispiel Punkte, Linien und dreieckige Polygonnetze
  - Bringt die virtuelle Welt auf den Bildschirm

- Befasst sich mit geometrischen Transformationen und Typen

- Befasst sich ebenfalls mit Texturen, Materialien und Beleuchtung.

- Transferiert die 3D Modelle zur Graphikkarte

- Darstellung Effekte & Materialien (z.B. Schatten)

- Dient dem Mangement von Fenstern, Viewports und Kameras

Szenengraph

- Der gesamte Aufbau der Welt ist eine Szene
- Alle Objekte sind Teil dieser Szene
- Gegenseitige Beziehungen zwischen Objekten und ihren Attributen sind der Szenengraph
- Szenengraph dient zur Begrenzung der vom Renderer dargestellten Objekte
  - Zum Beispiel im Falle von keiner Sichtbarkeit des Objekts
  - Entlastung des Renderers, da weniger Verarbeitung

Kollisionserkennung und Physik

- Kollisionserkennung ist wichtig damit Objekte sich nicht gegenseitig durchlaufen können

- Kollisionserkennung ist unerlässlich für Interaktion mit virtueller Spielwelt
  - Daher wird ein Physiksystem nötig

- Physiksysteme muss physikalische Grenzen bestimmen und Kollisionen verarbeiten
  - Bewegung der kollidierenden Objekte verändern

- Physik eines Spiels ist eine Simulation der Dynamik von rigid bodies
  - Bestimmt wie rigid bodies sich über Zeit und unter Einfluss von Kräften bewegen und interagieren

- Heute entwickeln nur wenige Firmen noch ihr eigenes Kollisionserkennungs- bzw. Physiksystem
  - Stattdessen werden SDKs von externen Entwicklern verwendet
  - Beispielsweise die Havok Physik Engine oder PhysX von Nvidia
  - Viele weitere Open Source Systeme wie beispielsweise die Open Dynamics Engine (ODE)

Frontend

- Frontend eines Spiels ist häufig das Heads-Up Display (HUD)
  - Dazu gehören z.B. Energie- oder Munitionsanzeige
  - Ebenso Spielemenüs, eine Konsole oder bestimmte Entwicklungswerkzeuge

- Auch die grafische Benutzeroberfläche gehört zum Frontend(GUI)
  - Darüber können Spieler Aufgaben wie beispielsweise die Manipulation eines Inventars vornehmen

- Videos und Zwischensequenzen die während des Spiels gezeigt werden gehören auch zum Frontend

Ressourcen Manager

- Externe Daten (z.B. Bilder, Leveldateien und Skripte) sollten möglichst nur bei Bedarf geladen werden
  - Sind sehr groß und dadurch können sie nur langsam aus dem Festplattenspeicher geladen werden
  - Daher werden oft Ressource-Manager eingesetzt, welche die Verwaltung im Arbeitsspeicher übernehmen

- Stellt eine Schnittstelle für einen Zugang zu Bestandteilen des Spiels zur Verfügung
  - Beispiel sind Bitmaps, Modelle, Texturen oder Audiodateien

Animationssystem

- Animationen sind wichtiger Bestandteil in jedem Spiel

- Es existieren fünf verschiedene Grundtypen von Animationen, die in Spielen verwendet werden:
  - Sprite / Texture Animation
  - Rigid Body Hierarchy Animation
  - Skeletal Animation
  - Vertex Animation
  - Morph Targets

- Skeletal Animation ist die am weitesten verbreitete Methode der Animation
  - Dabei wird ein detailliertes dreidimensionales Charakter Polygonnetz von einem Animateur mit Hilfe eines relativ simplen Knochensystems positioniert
  - Wenn sich die Knochen bewegen, bewegen sich die Vertices des dreidimensionalen Objektes mit

Audiosystem

- Das Audiosystem ist für entsprechende Effekte wie etwa 3D Sound und um die Wiedergabe und Verwaltung von Audiodateien zuständig

- Nahezu alle aktuellen Spiele bzw. ihre zugrundeliegende Engine unterstützen 5.1 oder 7.1-Raumklang und Technologien wie z. B. EAX
  - Dadurch wird der räumliche Eindruck der Spielwelt verstärkt, indem ein differenzierter Raumklang entsteht
    - Zum Beispiel erlaubt es der Klang, die Position von Gegnern zu orten
    - Außerdem können klanglich verschiedene Räumlichkeiten wie Badezimmer, Hallen, Gänge, Höhlen oder Unterwasser-Klangdämpfung simuliert werden

Netzwerkfunktionalität

- Spiele haben heutzutage häufig einen Mehrspielermodus

- Daher enthalten Game Engines häufig Netzwerkfunktionen

- Wird ein Mehrspielermodus in ein Spiel integriert, so muss das gesamte Konzept bestimmter Komponenten dementsprechend angepasst werden

- Hier werden ebenso die Voraussetzungen für den Multiplayer-Teil festgelegt
  - z. B. wie viele Spieler gleichzeitig am Spiel teilnehmen können oder ob ein 56k-Modem ausreichend ist oder ein Breitbandinternetanschluss wie DSL oder TV-Kabel benötigt wird

- Man kann sich zwischen den zwei Netzwerkprotokollen UDP und TCP entscheiden
  - Der Vorteil von TCP besteht in der Sicherheit, dass Daten sicher und in der richtigen Reihenfolge beim Clienten ankommen
  - UDP hingegen ist schneller, es können aber Paketverdopplung, Paketverlust oder Durchmischung auftreten
  - Meistens werden beide Protokolle gleichzeitig verwendet, um flüchtige Daten über UDP zu senden und sichere Daten, wie zum Beispiel die Anmeldung über TCP

Scripting

- Skriptsprachen dienen der Programmierung der Spielabläufe
- Viele Game Engines verwenden Skriptsprachen, um die Entwicklung von spielspezifischen Regeln und Inhalten einfacher und schneller abwickeln zu können
- Ohne Skriptsprachen müsste das Spiel neu kompiliert und gelinkt werden, falls eine Änderung an der Logik oder den Datenstrukturen vorgenommen wird
- Skriptsprachen sind leichter zu erlernen als reine Entwicklungssprachen
- Durch Skriptsprachen wird die Spiele-Engine wird universeller und kann auch von den Nutzern erweitert werden

Künstliche Intelligenz (KI)

- Dieselben Muster für künstliche Intelligenz tauchen in so ziemlich jedem KI System auf

- Mit der KI wird versucht eine menschenähnliche Intelligenz nachzubilden, d. h., einen Computer zu bauen oder so zu programmieren, dass dieser eigenständig Probleme bearbeiten kann

- Bei Computerspielen wird die Intelligenz jedoch nur nachgeahmt indem durch meist einfache Algorithmen ein intelligentes Verhalten simuliert wird

- Die KI beschäftigt sich größtenteils mit Problemen, bei denen nach bestimmten Lösungen gesucht wird
  - Verschiedene Suchalgorithmen werden dabei eingesetzt
  - Ein Beispiel für die Suche ist die Wegfindung, die in vielen Computerspielen eine zentrale Rolle spielt

Vielen Dank für eure Aufmerksamkeit

Quellen

http://userpages.uni-koblenz.de/~cg/ss13/cg2/01_3D%20Engines.pdf

http://www.uni-protokolle.de/Lexikon/Game-Engine.html

http://cav.martin-schreiner.info/Ausarbeitung.pdf

http://www.uni-koblenz.de/~cg/Diplomarbeiten/DiplomarbeitFugger.pdf

http://de.wikipedia.org/wiki/Spiel-Engine

http://www.dennis-schneider.com/downloads/BachelorThesis_Evaluation3DGameEngines.pdf

http://theses.fh-hagenberg.at/system/files/pdf/Dammerer10.pdf