Ludwig Maximilians Universität München
Institut für Informatik
Lehr- und Forschungseinheit für Datenbanksysteme

Skript zur Vorlesung

Managing and Mining Multiplayer Online Games

im Sommersemester 2016

Kapitel 1: Computerspiele

Skript © 2012 Matthias Schubert

http://www.dbs.ifi.lmu.de/cms/VO_Managing_Massive_Multiplayer_Online_Games

Organisatorisches

- ***Aktuelles***
  - Vorlesung: Mi, 13.00 - 16.00 Uhr, Raum S 006 (Schellingstr. 3)
  - Übungen:
    - Do, 12.00 - 14.00 Uhr Raum M 209 (Hauptgebäude)
    - Do, 14.00 - 16.00 Uhr Raum M 010 (Hauptgebäude)
- ***Homepage der Vorlesung***:
  http://www.dbs.ifi.lmu.de/cms/VO_Managing_Massive_Multiplayer_Online_Games
- ***Übungen***: Gregor Jossé
- ***Klausur***:
  Der Stoff der Klausur wird in der Vorlesung und in den Übungen besprochen. (Das Skript ist lediglich eine Lernhilfe)

Was sind Computerspiele ?

Computerspiel:

„An interactive experience that provides the players with an increasingly challenging sequence of patterns which he or she learns and eventually masters.“ [Ralph Koster: A Theory of Fun for Game Design, Phoenix, AZ, Paraglyph 2004]

⇒ *Interactive*: Interaktion mit dem Computer oder Mitspieler

⇒ *Experience*: Wahrnehmen und Erleben
des Inhalts spielen eine wichtige Rolle

⇒ *Challenge, Learning and Mastering*:
Ein gutes Spiel ist herausfordernd, aber erlaubt eine Verbesserung bis hin zur Meisterschaft.

Verwandte Arten von Software

- *Real-Time Simulationen*: Modellierung und Darstellung einer realen physikalischen Umgebung und ihrer dynamischen Entwicklung. Verkehrssimulationen oder Simulation von Schwarmverhalten.

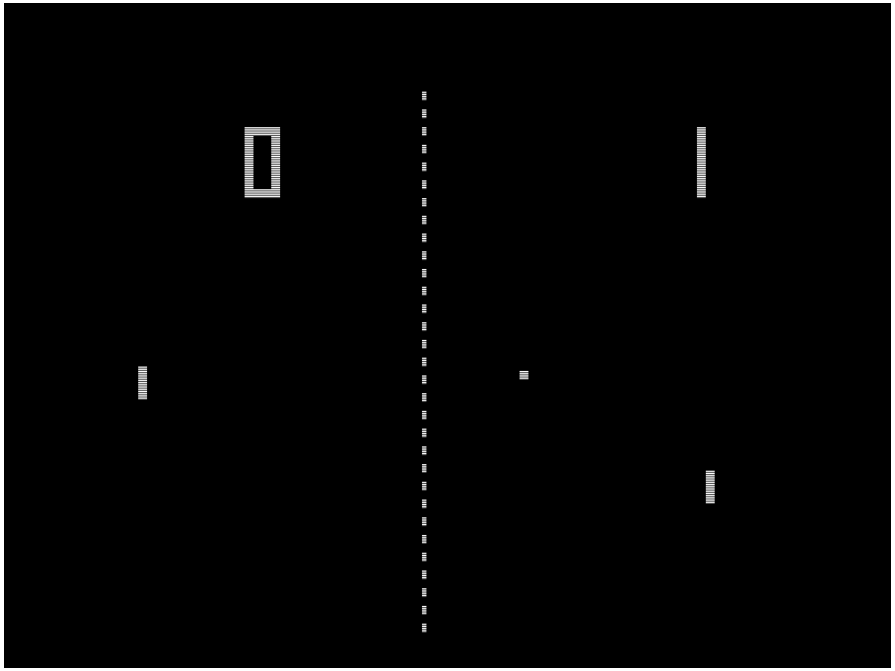
- *Geo-Informationssysteme*: Verwaltung von räumlichen Informationen (2D-/3D-Karten) und der sich darin bewegenden Objekte, z.B. Autos.

- *Agentensysteme*: Systeme, deren Zustand über eine Menge von unabhängig agierenden Einzelentitäten abhängen.

- *Virtuelle Realität*: Rendering von Sound und 3D Umgebungen.

Die obligatorische Pong Folie

erstes kommerziell erfolgreiches Computerspiel: PONG (Atari 1972)



- Pong ist ein einfaches Tennisspiel (Geschicklichkeitsspiel)
- es wurden eigene Kontroller dafür gebaut
- wurde in den folgenden Jahren oft kopiert und modifiziert

Soziale Aspekte von Computerspielen

- 58% der Amerikaner konsumieren Videospiele.
- Der Durchschnittsspieler ist 30 Jahre alt und spielt seit 13 Jahren.
- Der Durchschnittskäufer ist 35 Jahre alt.
- 45% der Spieler sind Frauen. Frauen, die älter als 18 J. sind haben mit 31% einen größeren Anteil als Jungen unter 18 J.. (19%)
- In 2013 spielten 36% der Amerikaner über 36.
- Eine Verbindung zwischen Videospielen und Gewaltbereitschaft ist bis heute nicht wissenschaftlich belegt.
- Es gibt mehrere Untersuchungen, die ein negative Korrelation zwischen der Verbreitung von Computerspielen und der Anzahl von Gewaltverbrechen belegen.

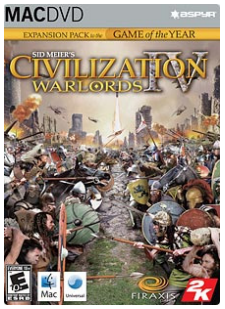
Quelle : http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf

Einnahmequellen und Geschäftsmodelle

Verkauf

- Einmaliger Preis, den der Kunde zahlen muss, um die Software zu erwerben.
- Verkauf über Geschäfte (Boxed Games) oder auch online
- für Offline-Spiele geeignet
- hohes Einstiegsrisiko für Spieler / bei nicht Gefallen wird Kaufpreis nicht rückerstattet
- Gefahr von Raubkopien
- Marketing und Werbung für Erfolg wichtig

Beispiele: Warcraft, Call of Duty, Resident Evil, Gran Tourismo, Grand Theft Auto, Civilization, …

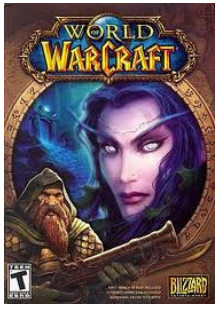


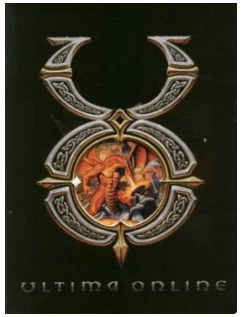
Einnahmequellen und Geschäftsmodelle

Abonnement

- Kunde muss regelmäßig Geld für ein Spieler-Account zahlen
- nur für Online-Spiele geeignet
- Account-Verwaltung ist notwendig
- neue Kriminalitätsformen entstehen (z.B. Account-Kidnapping)
- garantiert dem Betreiber regelmäßige Einkünfte
- Content-Updates und Kundenzufriedenheit sind sehr wichtig

Beispiele: Ultima Online, World of Warcraft, Everquest, …





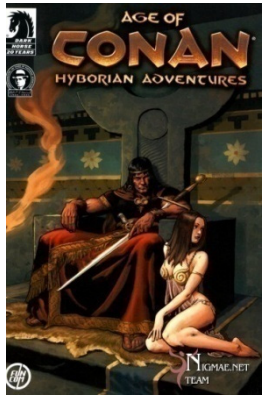
Einnahmequellen und Geschäftsmodelle

Micro-Transactions

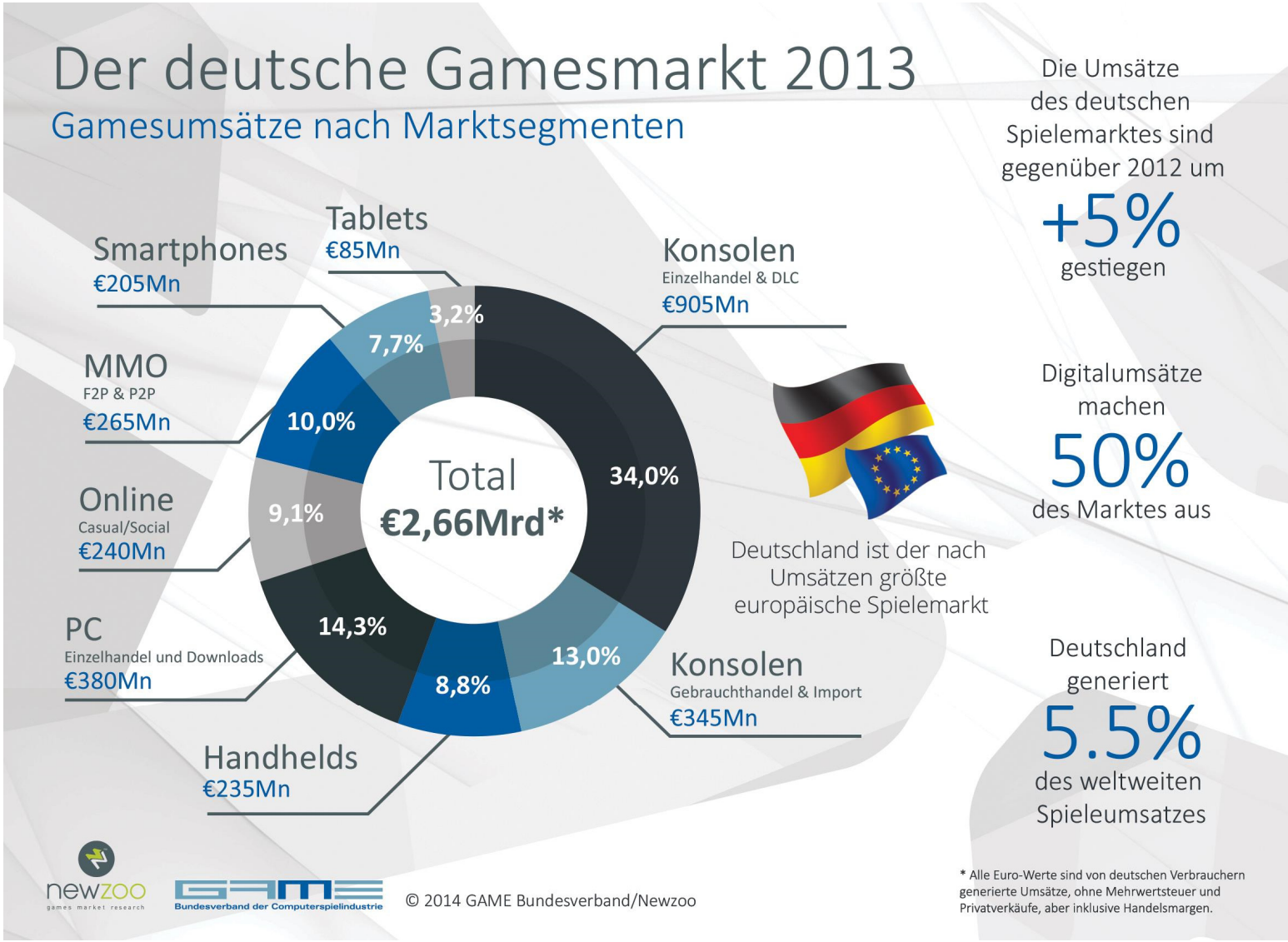
- Verkauf virtueller Zahlungsmittel, Gegenstände oder Fähigkeiten gegen reales Geld
- geeignet für free-to-play Spiele
- Entwicklungsrisiko am höchsten aber sehr hohe Gewinnspanne
- geringste Einstiegsschwelle für Kunden, geringes Einstiegsrisiko

  (Spieler investiert erst bei Gefallen Geld)
- birgt die meisten Gefahren durch Betrug, da Spielwirtschaft mit realer Wirtschaft verbunden sein muss

Beispiele: Age of Conan, Herr der Ringe online, Farmville, Travianer, ..



Markt in Deutschland 2013



http://game-bundesverband.de/index.php/de/neues/140-game-newzoo

Marktentwicklung der letzten Jahre

- Der Markt für Computerspiele ist einer der größten in der Unterhaltungsbranche (ca. 93 Mrd. US-Dollar weltweit)

  Zum Vergleich: Der Markt für Datenbanksysteme hat ungefähr ein Volumen von ca. 20 Mrd. US-Dollar

- Der Anteil rein digitaler Umsätze ist 2013 um 25% gestiegen.

- MMO, Casual-Game-Portale and Social Network Games hatten einen Zuwachs von 27-66% in 2010

E-Sports

E-Sports: Computerspiele als sportlicher Wettkampf:

- 71 Millionen Zuschauer 2014
- 36 Millionen Zuschauer bei der League of Legends (LoL) World Championship 2015
- (NBA finals: 23,3 mil)
- 27 mil. täglich aktiver Spieler in LOL
- 18 mil. USD Preisgelder beim “The International” 2015, weltgrößtes Dota6
- ca. Platz 6 bei professionellen Sportwetten

Erfolgsfaktoren

- ansprechende Erfahrungswelt, herausfordernd aber stetige Verbesserung möglich (gutes Spieldesign)
- gute Servicequalität: Verfügbarkeit, Kundendienst, etc. .
- ausreichende Spieleranzahl: je mehr Spieler ein Spiel regelmäßig spielen, desto interessanter wird es.
- Spieldauer: Je mehr Zeit Spieler in ein Spiel investieren, desto erfolgreicher ist ein Titel:
  - Soziales Prestige des Spiels steigt bis zu einem gewissen Grad
  - Mehr Verkäufer, längere Abonnement-Dauer, mehr Transaktionen

  Aber es gibt auch negative Folgen des Erfolgs:
  - Suchtgefahr: Prestige fällt gewaltig
  - große Community => Lösungen werden schnell publik
  - Systematischer Betrug konzentriert sich auf erfolgreiche Titel

=> *Langfristige Kontrolle des Spiel-Design und des Spielerverhaltens sind wichtige Aspekte für Spielbetreiber*.

Unterscheidungsmerkmale von Computerspielen

- *Anzahl gesteuerte Spiele-Entitäten*: Einheit, Gruppen, Fraktionen
- *Darstellungsperspektive*: First-Person, Third Person, Vogelperspektive
- *Zeitlicher Ablauf*: Real-Time, Game-Time, rundenbasiert
- *Komplexität der Steuerung*: Simulation vs. Fun-Control,
  Manual Aim vs. Auto-Target, …
- *Dimension/Granularität der Spielwelt*: 2D, 3D,
  fixe Spielfelder vs. freie Bewegung
- *Anzahl der Spieler*: Single-Player, Multi-Player (1-100),
  Massive-Multiplayer (100-100.000)
- *Entwicklung des Spieler-Accounts*: Virtuelle Fähigkeiten, Spieler-Fähigkeiten (RPG vs. Schach)
- *Einfluss des Zufalls*: Würfelspiel vs. Schach

Klassische Spiele-Genres

Real-Time Strategie Games (RTS)

- Spiele, die Aufbau einer Armee und Sieg über einen Gegner zum Ziel haben
- Steuerung sehr vieler Einheiten, Vogelperspektive, geringe Granularität der Steuerung, Echtzeit, 2D-/3D-Welt, Spieleranzahl: 1-10
- Beispiele: Dune II, Starcraft, Warcraft, Command & Conquer, Age of Empires, …



Klassische Spiele Genres

Massive Multi Player Role Playing Games (MMORPGs)

- Entwicklung und Geschichte eines Spielercharakters
- Eine Einheit, Third-Person View, mittlere Granularität der Steuerung, Echtzeit, 3D-Welt, Spieleranzahl sehr hoch (mehrere tausend), Entwicklung des Charakters ist zentral
- *Beispiele*: Ultima Online, Everquest, World of Warcraft, Star Wars: The Old Republic, Age of Conan, Herr der Ringe Online, …

Klassische Spiele Genres

Multiplayer Online Battle Arena Games(MOBAs)

- Team PVP Spiel (meist 5 vs. 5)
- Mischung aus RTS und MMORPG/ populär als E-Sport
- Eine Einheit pro Spieler, Third-Person View, mittlere Granularität der Steuerung, Echtzeit, Entwicklung der eigenen Einheit, Teamplay,..
- Ziel: Zerstören der gegnerischen Basis, manchmal andere Ziele
- *Beispiele*: League of Legends(LoL), Dota, Dota2, Heroes of the Storm,..

Klassische Spiele-Genres

First-Person-Shooter (FPS)

- Simulation von Feuergefechten
- Eine Einheit, Ego-Perspektive, Echtzeit, sehr hohe Steuerungsgranularität, 3D-Welt, meist 1-20 Spieler
- *Beispiele*: Wolfenstein 3D, Doom, Unreal, Half-Life, Team Fortress, Halo, Counter Strike, …

Klassische Spiele-Genres

Rennspiele

- Steuerung von Fahrzeugen
- Eine Einheit, Ego und Third-Person Perspektive, Echtzeit, sehr hohe bis mittlere Steuerungsgranularität, 2D-/3D-Welt, meist 1-2 Spieler aber auch Multi-Player möglich
- *Beispiele*: Atari's Space Race (1970), Test Drive, Need for Speed, Outrun, Gran Tourismo, …

Klassische Spiele-Genres

Fighting Games

- Spiele, die Nahkämpfe mit Gegnern zum Inhalt haben
- Eine Einheit, Third-Person View, mittlere Granularität der Steuerung, Echtzeit, 2D-/3D-Welt, Multi-Player Game
- Beispiele: Heavyweight Champ (1973), Karate Champ (1984), Street Fighter, Tekken, Mortal Combat, ...

Klassische Spiele-Genres

Wirtschaftssimulationen und rundenbasierte Strategiespiele

- Spiele, die Aufbau einer Stadt, einer Nation, einer Firma oder eines anderen Verbundes zum Ziel haben
- Übergang zum Strategiespiel bei Konflikten mit Gegnern
- Viele Einheiten, Vogelperspektive, Echtzeit oder rundenbasiert, niedrige Steuerungsgranularität, 2D-Welt, ein bis zu tausende Spieler
- *Beispiele*: Intopia (1963), M.U.L.E (1983), Civilization, Die Siedler, Sim City, Travianer, ...



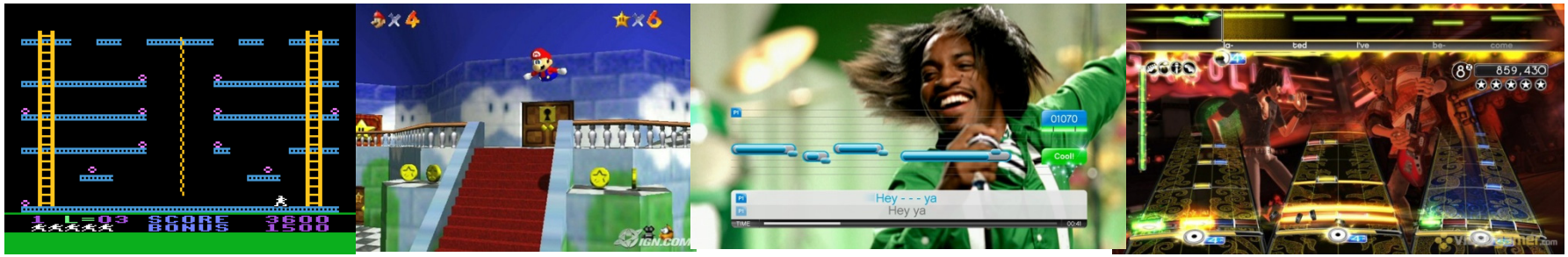
Klassische Spiele-Genres

Adventure Games

- Spiele, die eine interaktive Geschichte beinhalten
- Interaktion besteht meist aus dem Lösen von Rätseln und sammeln von Gegenständen
- Perspektive meist Third-Person, Granularität je nach Titel sehr variabel
- Übergang zu anderen Genres durch Action-Sequenzen ist möglich
- *Beispiele*: Zorc, Monkey Island, Leisure Suit Larry, Deponia, …

Weitere Genres

- ***Jump 'n' Run***: Steuerung eines Third-Person Charakters durch eine 2D-/3D-Landschaft (z.B. Super Mario, Jumpman, ...)
- ***Sing-, Musik-, Rhythmus-, Tanz-Spiele***: Zu einem Musikstück passende Eingaben werden mittels des Computers bewertet. (z.B. Sing Star, Rock-Band, ...)
- …



Es gibt noch zahllose weitere Spiele, die nicht unmittelbar in eine Kategorie fallen, sondern

- Kombinationen von bereits genannten Genres darstellen
- vollkommen andere Ideen in ein Spielerlebnis umwandeln

Aufbau von Computerspielen

- Moderne Computerspiele können vielschichtige komplexe Softwaresysteme sein, für deren Entwicklung große Teams notwendig sind.

- Je nach Aufbau und Eigenschaften unterscheiden sich Computerspiele in ihrer Komplexität und ihrem Umfang.

- Gerade durch die neuen Plattformen (Android, iPhone/iPad, Browser und Soziale Netzwerke) erfahren einfache Spielprinzipien gerade eine Renaissance.

- Selbst einfache Spielprinzipien erfordern im MMO-Umfeld einen erheblich höheren Aufwand durch Skalierungsproblematiken und der Notwendigkeit das Spielerverhalten zu analysieren.

Aufbau eines Entwicklerteams

- ***Software Engineers***: Entwicklung der Game Engine/Game Core und der Tools für Artists und Game Designer
- ***Artists***: Zuständig für den Content: Konzepte, 3D-Modelle, Sound, Animation, Texturen, Lichteffekte, Animation und Motion Capture, Schauspieler, ...
- ***Game Designers***: Zuständig für das Gameplay (Interaktion) und der Gaming-Experience: Story, Leveldesign, Spielziele, Verteilung der Gegner, Equipment, Spielerfähigkeiten, ...
- ***Produzenten***: Senior Game Designer, Projektmanager
- ***weitere Mitarbeiter***: Marketing, Team-Assistenten, technisches Personal, ...
- ***Publishers und Studios***: Marketing, Herstellung, Distribution des Spiels

Aufbau von Computerspielen

Faktoren, die gegen eine Standardarchitektur sprechen:

- Content steht im Vordergrund
- Unterschiedliche Anforderungen je nach Spiel
- hohe Anforderungen an Performanz
  (je weniger Ressourcen, je mehr potentielle Spieler)
- Hardware-Ressourcen verändern sich oft sehr schnell

Folge: Architektur ist häufig integriert und spezifisch als modular und wiederverwendbar.

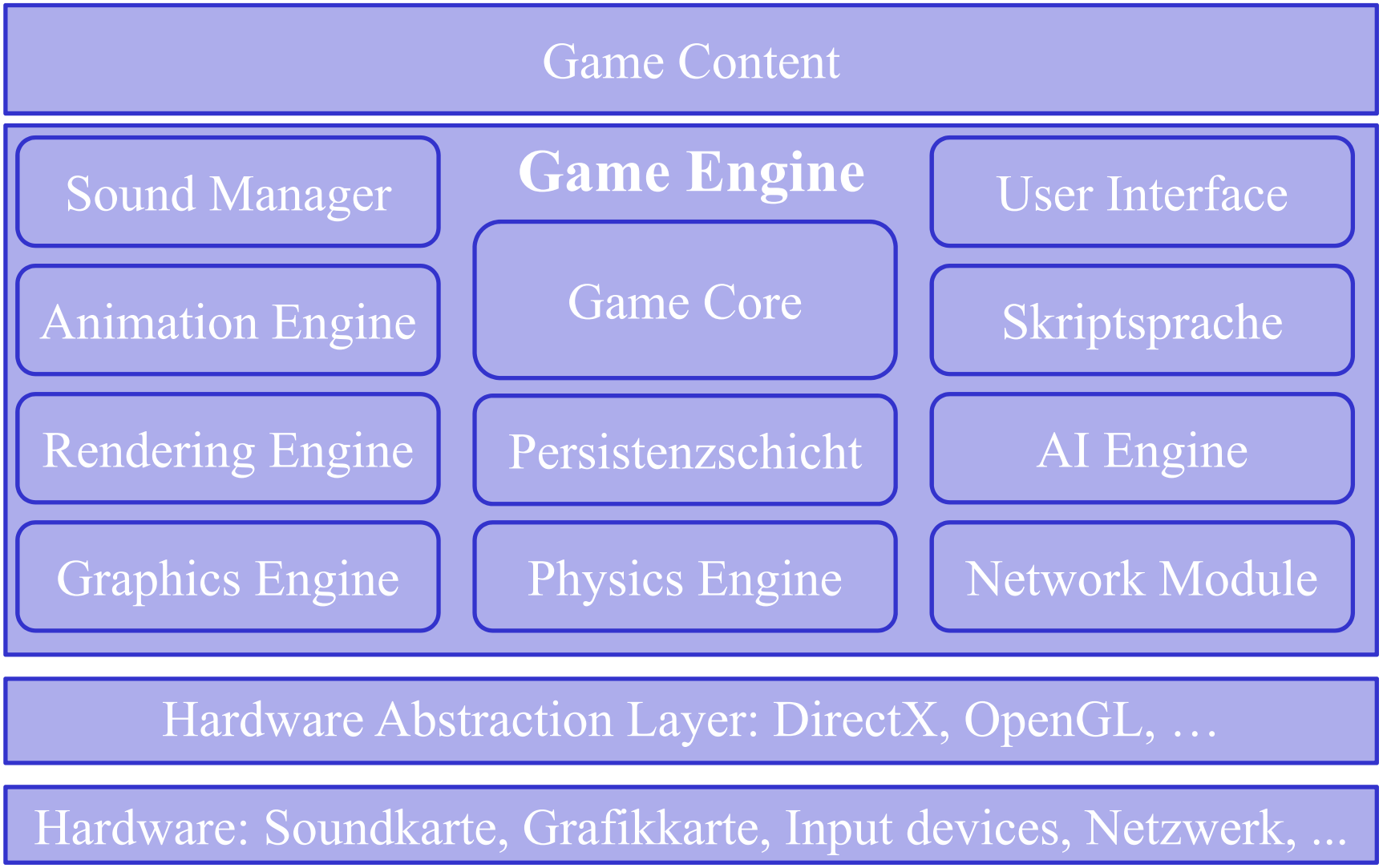
Aber: Spiele sind komplexe Softwaresysteme

$\Rightarrow$ Verwendung von Standardkomponenten (*Engines*)

$\Rightarrow$ Geschichtete Architekturen und modularer Aufbau

$\Rightarrow$ Wiederverwendung des Codes für weitere Spiele (*Mods*)

Bausteine in der Architektur eines Spiels



Hardware Layer und Hardware Abstraction Layer

- Komponenten der Hardwareplattform
  - Graphikkarte
  - Soundkarte
  - Eingabegeräte (Tastatur, Maus, Joysticks, Gamepads, Lenkräder, Mikrofone, Kamera, ...)
  - Sekundärspeicher
  - Hauptspeicher
  - Prozessor
- Device Driver:
  - Treibersoftware, die Hardware vom Betriebssystem aus ansprechbar macht.
  - Heutzutage setzen die meisten Spiele auf einem Betriebssystem auf.

Hardware Abstraction Layer

- Kapselung der Hardwareeigenschaften bei unterschiedlicher Hardwarekonfiguration (PC, Android)
- Bietet eine einheitliche Schnittstelle zur Hardware
- Angebotene Funktionalitäten erlauben Basisbefehle

- **Beispiele**:
  - Glide: 3D-Grafik SDK (überholt)
  - DirectX: Microsofts 3D-Grafik SDK
  - OpenGL: weit verbreitetes SDK, implementiert auf vielen Plattformen
  - libgcm+Edge: Graphic Interface der PlayStation 3

Graphics Engine

- Höhere Zugriffsebene auf Grafikfunktionalitäten
- Meist auf eine spezifische Art der Darstellung zugeschnitten
  - Sprites
  - Isometrisch
  - 3D
- Arbeit mit Modellen auf einer höheren Abstraktionsstufe
  - Sprites
  - Charaktere
  - Solids
- Realisiert kompliziertere Aspekte der Anzeige
  - Mini Maps
  - multiple Sichten
  - Overlays
  - Spezialeffekte

Rendering Engine

- Graphics Engine modelliert die Daten nur

- Rendering setzt die Modelle in eine Darstellung auf dem Bildschirm um:
  - Rendering ist abhängig von den Fähigkeiten der Graphikkarte
  - Setzte ebenfalls auf den Hardware Abstraction Layer auf

- Aufgaben der Rendering Engine:
  - low-level Aufbau und Bearbeitung des Szenegraphen
  - visuelle Effekte (Partikel, dynamische Schatten, High Dynamic Range Rendering (HDR-Effekte), Spiegeleffekt, Nebel, …)
  - Darstellung des Front-End (GUI, Video, Menüs, Head-Up-Display (HUD))

Physics Engine

- Simuliert die „Regeln“ der Welt
  - Physikalische Grundregeln (Schwerkraft, Bewegung, Trägheit, ...)
  - Kollisionen (häufig: Physics Engine = Collision Engine)
  - Krafteinwirkung auf Objekte (Zerstörung, Zerbrechen, Verformung, ...)
  - Explosionen
  - Ragdoll Charaktere (z.B. für dynamische Todesanimationen)

- Physik wird zu einem wichtigen Bestandteil mancher Spiele
  - Spiele und Echtzeitsimulationen rücken näher aneinander
  - Animationen basieren auf physikalischen Grundlagen
  - Objektinteraktionen nutzen Physik (Kollisionen, ...)

Physics Engine

- Verwendung je nach Genre und Perspektive
  - Simulationsspiele: Sehr hoher Detailgrad, umfangreiche Physics Engines
  - Rundenbasierte Strategiespiele: Keine direkte Physics Engine, mögliche Züge werden aus abstrakteren Regeln des Spiels abgeleitet.

- Einige verfügbare SDKs:
  - Havok
  - Open Dynamics Engines (ODE)
  - Tokamak
  - PhysX (Nvidia, Angeia) Software in Verbindung mit Spezialhardware (Physics Processing Unit (PPU))

Animation Engine

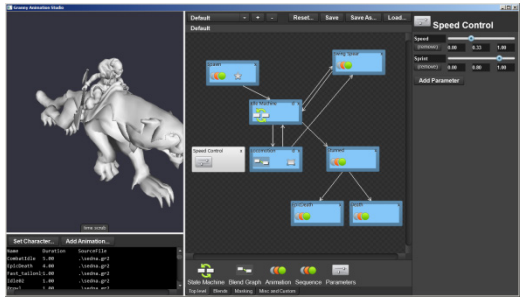
- Zuständig für eine natürliche Bewegung der Objekte/Charaktere

- Offline-Aufgaben:
  - *Motion Capture und Retargeting*: Festhalten von Bewegungsabläufen realer Vorbilder und Übertragung auf andere Computermodelle
  - *Motion Editing und Adaption*: Modifikation aufgenommener Bewegungsabläufe und Übertragung von einen Modell auf ein anderes

- Echtzeit-Aufgaben:
  - Animation von Sprites oder Texturen
  - Animation von Graph-Modellen
  - Festkörper- oder Skelettbewegungen

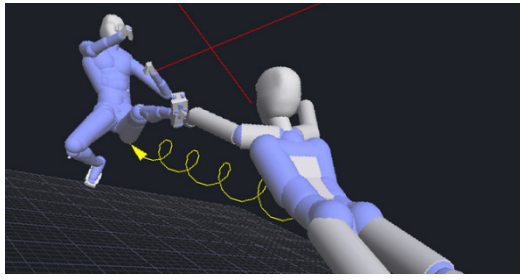
Animation Engine

- Pakete für die Implementierung von Animation Engines:
  - Granny (weit verbreitet, >2000 Spiele)



  - Havok Animation
    (Erweiterung der Havok SDK,
    siehe Physics Engine)

  - Endorphin (für Filme, aber auch Spiele)



  - Edge (Animation für die PlayStation 3)

Sound Engine und Audio Manager

- Abspielen und Erzeugen von Geräuschen und Musik
- Wichtig für das Spielerlebnis
- Sound Rendering:
  - Abspielen des Sounds in einer bestimmten Simulation
  - Synchronisation von Bild und Ton
  - Abmischen der Geräusche (Lautstärke der Kanäle zueinander)
  - 3D und Entfernungsunterschiede
- Gängige Formate: midi, wave, mp3, ogg, acc, ...
- Sound Manager:
  - xACT für PC und Xbox 360
  - EA: SoundRIOT
  - Sony: Scream für PlayStation 3
  - IrrKlang, OpenAL, FMOD, …

Game Core

- Kern der simulierten, virtuellen Welt
- Speichert den aktuellen Zustand der Spielwelt
- Kontrolliert den Übergang von einem gültigen Zustand zum nächsten (Game Loop)
- Zeit-Management: echtzeitbasiert (real time), rundenbasiert
- Granularität der Darstellung: Alle für die Regeln des Spiels relevanten Informationen
- Traditionell sehr eng mit den anderen Komponenten verwoben
- Neuere Modelle trennen die Komponenten stärker
- Bei großen Spielwelten:
  - Unterstützung räumlicher Anfragen
  - Aufteilung der Spielwelt
- Mehr im Kapitel 2

Network and Multiplayer Modules

- Regelt den Datenaustausch über mehrere Rechner hinweg
- Arten von Multiplayer-Spielen:
  - Single-Screen Multiplayer (z.B. Super Mario)
    - eine Sicht auf die Spielwelt
  - Split-Screen Multiplayer (z.B. Super Mario Kart)
    - jeder Spieler hat eigene Sicht auf dem gleichen Bildschirm
  - ***Networked Multiplayer*** (z.B. StarCraft, Counter Strike, ...)
    - ca. 2-20 Spieler, häufig kein dedizierter Server (dedizierter Client fungiert als Server)
    - geringer Overhead zum Verteilen der Daten (kleine Spielwelten)
  - ***Massive Multiplayer Spiele*** (z.B. Eve-Online, World of Warcraft, ...)
    - ca. 1.000-100.000 Spieler in einer Spielwelt
    - großer Overhead zum Verteilen der Daten (große Spielwelt)
    - aufwendiges Account-Management
    - sehr große Spielwelten

Network und Multiplayer Modules

- Aufwand je nach Genre: gering bis sehr aufwendig
- Verteilung der Spielfunktionalitäten auf verschiedene Rechner:
  - Synchronisation unterschiedlicher Spielzustände
  - Trennung der Funktionalitäten: Client-Server Modell
  - Alle machen alles und tauschen Daten aus: Peer-to-Peer (P2P)
- Design von High-Level Protokollen
- User-Authentifizierung
- Beeinflussung der Spielmechanik und des Designs
  - Spielgeschwindigkeit hängt von Latenz-Zeit ab
  - Spielfluss hängt von Serverkapazität ab
- Softwareplattformen für Netzwerkspiele: RakNet, GNE, ...
- Mehr im Kapitel 3

Persistenz System

- Schicht zur Speicherung von Charakteren und Spielständen
- Speichern von Replays/Spielverläufen
- Bei Online-Spielen:
  - Serveraufgabe
  - Kopplung mit der Account-Verwaltung
  - Absicherung gegen Datenverlust notwendig (Logging- und Recovery-Lösungen)
  - Effizienzprobleme bei sehr großen Spielständen und Echtzeitspielen
- Mehr im Kapitel 4

AI Engine

- Artificial Intelligence (AI) ist generell Teil des Spielinhalts und nicht unbedingt der Game Engine
- Grundfunktionalitäten werden aber mittlerweile über AI Engines gekapselt:
  - Wegewahl in beschränkten Umgebungen:
    Suche nach Ein- und Ausgängen, Hindernisse umgehen, ...
  - Verhalten und Interaktionen: meist regelbasiert
  - Entscheidungsfindung: ähnlich zu Expertensystemen
  - Gruppenverhalten: Schwarmbildung, Paniksimulation, ...
- Produkte und Packages:
  - AI-implant (Presagis)
  - Kynapse (AutoDesk)
  - DirectAI (Masa)
  - SimBionic
  - …

AI Engine

- Gruppenverhalten kann zentral gesteuert werden:
  - weniger Rechenoverhead
  - leichtere Kooperation der Entitäten
  - unflexibel bezüglich Gruppenzusammenstellung
  - in einigen Situationen unrealistisch
- Gruppenverhalten als Agentensystem:
  - Wahrnehmung: Input der Situation
  - Entscheidung: AI (feste oder gelernte Regeln, Zufall)
  - Action: Versuch der Ausführung der Entscheidung
  - Gruppenverhalten ergibt sich aus vielen Einzelhandlungen
  - Kooperatives Verhalten erfordert Kommunikation
  - Rechenaufwand i.d.R. höher, da jedes Gruppenmitglied einzeln berechnet werden muss
- Mehr in Kapitel 5

Scripting Engine

- Höhere Programmiersprache mit direktem Zugriff auf die Funktionalitäten der AI-Engine und des Game Cores
- Erleichtert den Designern den Inhalt zu gestalten
- Stellt eine wichtige Schnittstelle zwischen der Game Engine und dem eigentlichen Spielinhalten (Game Content) dar
- Beispiele:
  - LUA (http://www.lua.org)
  - GameMonkey (http://www.somedude.net/gamemonkey)
  - AngelScript (http://www.angelcode.com/angelscript)
- Alternativ kann das Spiel auch direkt in der Programmiersprache der Engine entwickelt werden. (häufig effizientere Lösung)

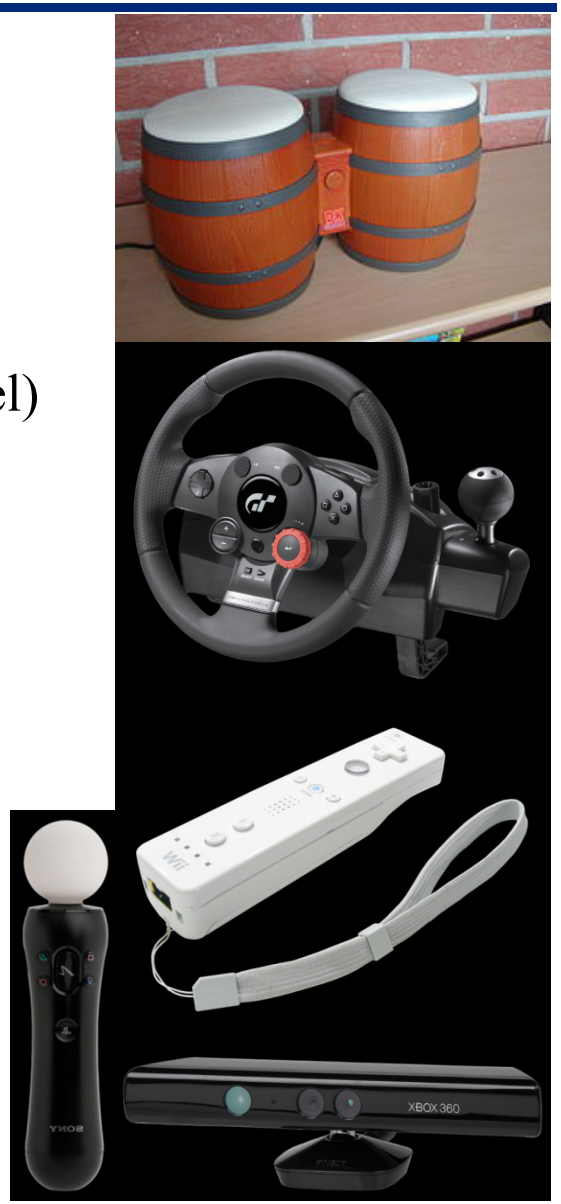
User Interface (UI)

- Effiziente und intuitive Schnittstellen sind notwendig für ein gelungenes Spielerlebnis:
  - direkte Reaktion des Spiels auf Eingaben
  - schneller Zugriff auf die notwendigen Fähigkeiten
  - geringe Lernkurve/Learning by doing (Tutorial Level)

- Enge Verknüpfung zur Graphics Engine

- Eingabemethode als Teil des Spielerlebnisses:
  - Eingabe über Spezialcontroller:
    Mikrofon, Wii-Gitarre, Lenkräder, DK Bongo, …

  - Eingabe über Bewegungserkennung:
    Wii-Controller, Xbox Kinect, PS Move, ...



Game Content

- Ziele, Aufgaben, ...
- *Entitäten*: Spielerentitäten (Charaktere), Non-Player Entitäten (NPCs, Mobs, ...), Umgebungsentitäten (Pflanzen, Felsen, ...)
- Levels, Karten, Aufgaben, ...
- Hintergrundgeschichte
- Videosequenzen
- *Balancing*: Ausgewogenheit der Einheiten, der Karten und des Schwierigkeitsgrades
- *Darstellungen*: Modelle, Texturen, Animationen, Effekte, Geräusche, Musik (künstlerischer Teil des Spiels)

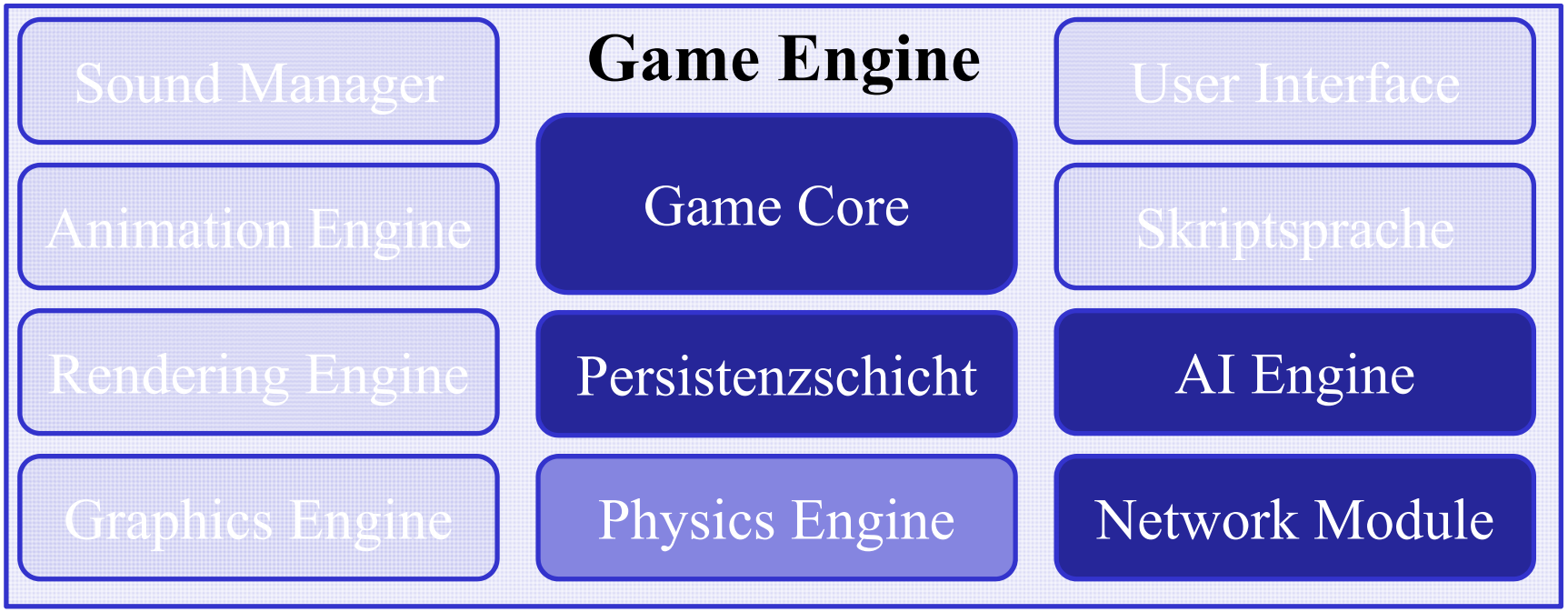
Überblick über die Vorlesung

Die Vorlesung gliedert sich in 2 Teile:

1. *Managing Massive Multiplayer Online Games*
   - Untersuchung der Komponenten, die für MMOs besonders interessant sind
   - Schwerpunkt liegt auf dem Game-Server
   - Client-Problematiken wie Grafik, Sound, Input/Output werden nicht besprochen

2. *Game Analytics*
   - Vorstellung zu Themenstellung und Aufgabenbereichen
   - Einführung in grundlegende Techniken des Data Mining
   - Modellierungsmöglichkeiten für Spieldaten
   - Techniken zur Lösung der oben eingeführten Problemstellungen

1. Teil der Vorlesung: Managing MMOs



- Untersuchung der dunkel gefärbten Komponenten
- Physics Engine wird in Verbindung mit dem Game Core angesprochen

2. Teil der Vorlesung: Game Analytics

- Erfolgreiches Design eines Computerspiels hängt stark vom Benutzerverhalten ab:
  - Ist das Spielerlebnis auch nach einiger Zeit noch interessant?
  - Gibt es nicht vorhergesehene Lösungswege, die das Spiel extrem vereinfachen und dadurch langweilig machen? (Exploits)
  - Passt der Schwierigkeitsgrad zu den Erwartungen meiner Zielgruppe?
  - Verschaffen sich manche Spieler unfaire Vorteile, oder umgehen sie die Regeln?
  - Wie suche ich passende Gegner in Spieler vs. Spieler Szenarien?
- Die neueren Geschäftsmodelle verstärken den Druck auf den Hersteller, ein dauerhaft interessantes Spielerlebnis zu erschaffen

$\Rightarrow$ ***Game Analytics***

Analyse des Spielerverhaltens zur Überprüfung des Einhaltens der Nutzerbedingungen und des Spiel-Designs.

Kapitelübersicht

1. Multiplayer Online Games

TEIL I: Managing MMOs

2. Der Game Core
3. Verteilte Spielearchitekturen
4. Persistenz-System
5. Künstliche Intelligenz

TEIL II: Game Analytics

6. Motivation und Aufgaben
7. Data Mining in a Nutshell
8. Sequenzdaten
9. Räumliche Daten
10. Spielerinteraktionen

Literatur

- Jason Gregory: Game Engine Architecture, AK Peters Ltd, MA,2009
- Nicolas Pronost: Game Engine Programming (http://www.cs.uu.nl/docs/vakken/mgep/)
- Critical Gaming Project: (https://depts.washington.edu/critgame/)
- Ralph Koster: A Theory of Fun for Game Design, Phoenix, AZ, Paraglyph 2004

Dinge, die man jetzt kennen sollte

- Definition Computerspiel
- Geschäftsmodelle
- Klassische Genres und Kategorisierungsmerkmale
- Aufbau von Computerspielen
- Komponenten der Spielarchitektur

insbesondere:

- Game Core und Physics Engine
- Netzwerk
- Persistenz System
- AI Engine
- Persistenz System