Technische Hochschule Rosenheim

# Entwicklung von Computerspielen: Game Engine; Teil 2

Fakultät Informatik
FWPM

Technische Hochschule Rosenheim

# Game Engines 2
# Übersicht

1. Nachtrag zu Scripting; KI Engine
2. Game Loop; verschiedene Ansätze

Technische Hochschule Rosenheim

# Game Engines 2
## Was gehört zur Engine; Was zum Scripting?

### Grafik:

- **Engine:**
  - Schatten
  - Beleuchtung
  - Verdeckungsberechnung
- **Scripting:**
  - Tageszeit
  - Entfernen/ hinzufügen von Lichquellen
  - Laden/ Entfernen von Objekten

Technische Hochschule Rosenheim

# Game Engines 2
## Was gehört zur Engine; Was zum Scripting?

### Physik:

- **Engine:**
  - Dynamik
  - Raycasting
  - Reaktion bei Kollisionen
- **Scripting:**
  - Kollisionsereignisse
  - Raycasting-Ereignisse
  - Masse von Objekten
  - Reibung
  - (ggf. andere Eigenschaften z.B. Elastizitätseinstellungen bei Kollision)

Technische Hochschule Rosenheim

# Game Engines 2
# Was gehört zur Engine; Was zum Scripting?

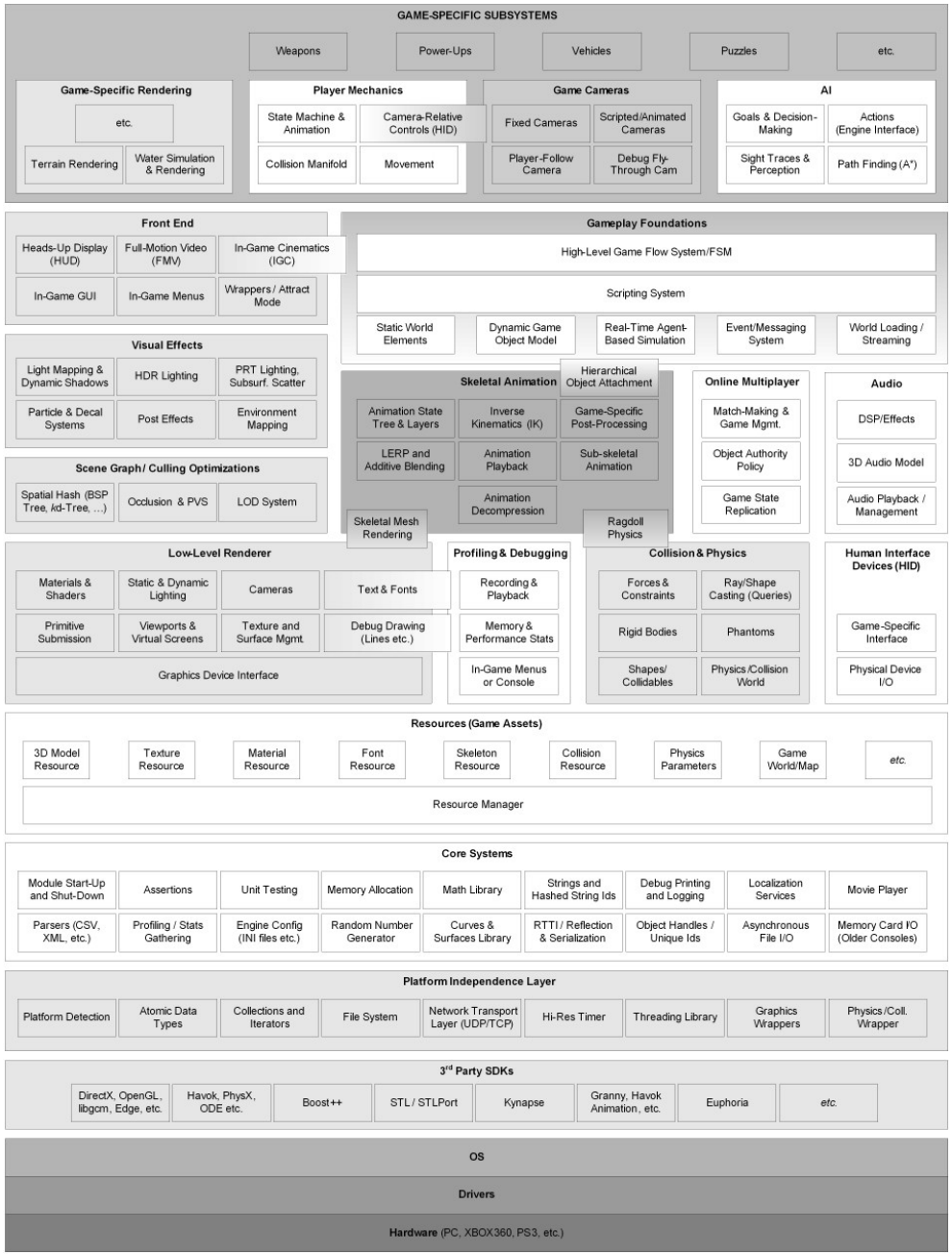
## KI:

- **Engine:**
  - Wegsuche
  - Planen
- **Scripting:**
  - Auswahl eines Weges
  - Treffen von Entscheidungen
  - Ziele

Technische Hochschule Rosenheim

# Game Engines 2
# KI Engine; Aufbau

- **Meist Agenbasierter Ansatz**
  - Wahrnehmung (Sensordaten, Dialog)
  - Entscheidung (Entscheidungsregeln, Lernprozess)
  - Aktion (Ausführen einer Liste von Operationen)
- **Traditionell eher Teil des Spiels als der Engine**
- **Verfügbare Engines:**
  - Kynapse (Autodesk)
  - AI-Implant (Presagis)
  - DirectIA (Masa)
  - SimBionic (Stottler Henke)




Technische Hochschule **Rosenheim**

Architektur aus:
J. Gregory: *Game Engine Architecture*, AK Peters, 2009.

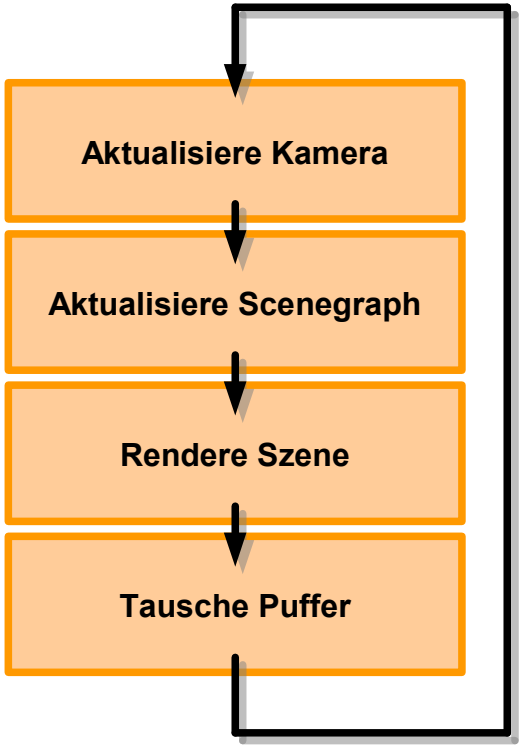
Technische Hochschule Rosenheim

# Game Engines 2
## Game Loop; Übersicht

- **Zeit ist in Spielen wichtig**
  Spiel muss in Echtzeit reagieren, um interaktiv zu sein
- **Zwei wichtige Schleifen:**
  Rendering Loop: Aktualisierung des Bildschirminhaltes
  Game Loop: Aktualisierung und Synchronisierung aller Subsysteme der Engine
- **Verschiedene Arten von Zeiten:**
  Reale Zeit
  Spielzeit (Simulierte Zeit)
  Lokale Zeit (Animationen, Audio etc.)
  Funktionale Zeit (CPU Zyklen)

# Game Engines 2
## Rendering Loop

- Typischer Aubau eines Schleifendurchlaufs:



Technische Hochschule Rosenheim

# Game Engines 2
## Rendering / Game Loop

- **Rendering muss mit relativ hoher Frequenz erfolgen:**
  25 – 30 Durchläufe pro Sekunde (Je nach Art des Spiels mehr)
- **Andere Subsystem haben unterschiedliche Aktualisierungsraten*:**

| | | |
|---|---|---|
| Ki : | 10 | *Durchläufe / s* |
| Eingabe : | 40 | *Durchläufe / s* |
| Physik: | 100 | *Durchläufe / s* |
| Audio: | 50 | *Durchläufe / s* |
| Haptische Rückkopplung: | 3000 | *Durchläufe / s* |

*Einige Davon müssen Synchronisiert werden: z.B. Grafik und Physik*

**Richtwerte; Je nach Engine Konfigurierbar*

- **Gameloop muss sicherstellen, dass alle Komponenten zur richtigen Zeit aufgerufen werden.**

Technische Hochschule Rosenheim

# Game Engines 2
# Game Loop

- **Die Meisten Subsysteme verwenden Lokale Zeit**
- **Typischerweise laufen drei Tasks Parallel:**
  - Eingabe (Spieler Input über HID Devices)
  - Spielelogik (Handlungsstrang, Spieler/Welt Zustand)
  - Feedback (Rendering/ HID Feedback/ Audio)

Technische Hochschule Rosenheim

# Game Engines 2
# Game Loop Ansätze

- **Gekoppelter Ansatz**
  Rendering und Spielelogik in einer einzigen Schleife nacheinander
- **Entkoppelte Ansätze**
  Lösung über getrennte Threads
  In gleicher Schleife, aber entkoppelt. (Spielelogik nicht jedes Frame)
  In gleicher Schleife, entkoppelt und mit eigener Frequenz pro Modul

Technische Hochschule Rosenheim

# Game Engines 2
# Game Loop; Beispiel Unity

https://docs.unity3d.com/Manual/ExecutionOrder.html