Echtzeitanforderungen an Virtual Reality Systeme – Interaktive Anwendungen mit 6 Freiheitsgraden

Echtzeit 2016

Sebastian Thomeczek
Fakultät für Informatik - Hochschule für angewandte Wissenschaften Landshut
sebastian.thomeczek@haw-landshut.de

Struktur

Einführung Virtual Reality

- Grundlagen
- Hardware: Headsets
- Hardware: Controller/Tracking
- Motion Sickness
- Echtzeitaspekte

Umsetzung einer VR-Anwendung

- Überblick
- Hardware
- Software: Tracking-Server
- Software: Controller Plugin
- Implementierung der Beispiel-Anwendung

Grundlagen Virtual Reality

- Präsentation einer virtuellen Realität
- Realistischer 3D „mittendrin“ Eindruck, computergeneriert in Echtzeit
- Ziele: Immersion & Präsenz durch Ersetzen von Wahrnehmungen
- sitzend, stehend, mit Bewegungsspielraum
- Herausforderungen:
  - Interaktion
  - Abbilden anderer Wahrnehmungen (Haptik, Bewegung)
  - Einschränkungen in VR (begrenzter Raum, Kabel)
- Unterscheidung:
  Virtual vs. Augmented Reality
- Anwendungszwecke:
  - Entertainment
  - Simulation / Visualisierung

VR-Hardware

VR-Hardware: Headsets

- Neue Hardware Generation
- HMDs (*Head-mounted displays*) mit Orientierungs- und Positions-Tracking (6 Freiheitsgrade)
- Darstellung mittels Linsen/Optiken in Kombination mit hochauflösenden Displays
- Tiefeneindruck mittels Parallaxe

- Beispiele VR Hardware:
  Oculus Rift, HTC Vive, PSVR, Razer OSVR
- Beispiele AR Hardware: Microsoft Hololens

VR-Hardware: Controller, Tracking

- *„Motion Controller“* für natürliche Interaktion
- Räumliche Interaktion (3D Position + 3D Orientierung)
- Alternativen:
  - Hand-Tracking
  - Gestenerkennung
  - Sprachsteuerung

- Tracking Techniken:
  - Optisch (IR): extern über Kamera oder „inside-out“ mit „Lighthouse“
  - Inertial: über interne Sensoren (Accelerometer, Magnetometer Gyroskop)

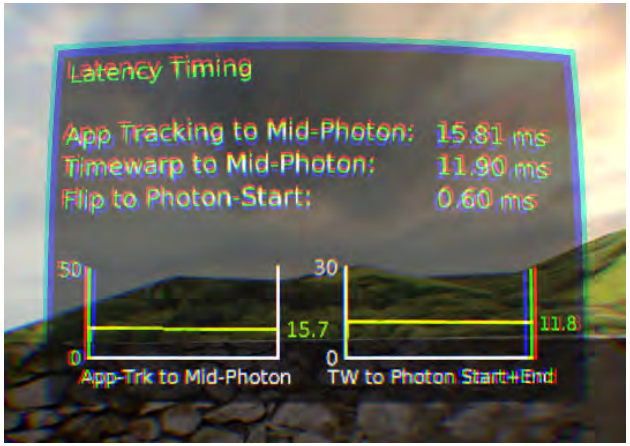
VR: Motion Sickness

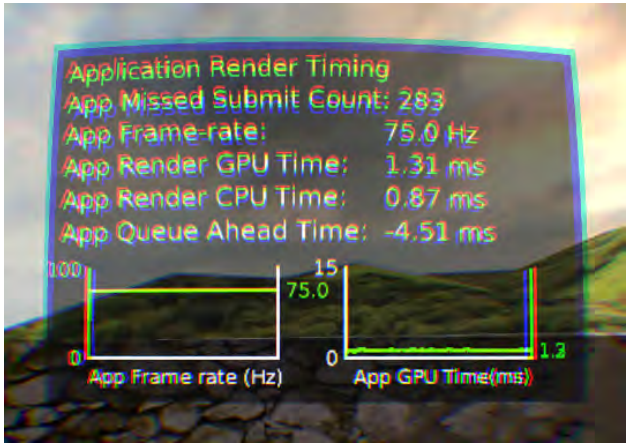
- Gefahr von Unwohlsein, Übelkeit und ähnlicher Symptome bei falscher Darstellung von VR
- z.B. durch Diskrepanz verschiedener Sinneswahrnehmungen
  (z.B. optischer Eindruck ↔ Gleichgewicht)

- Mögliche Abhilfen:
  - Referenzrahmen: Cockpit, „Virtual Nose“
  - Techniken zum 'verstecken' von Latenzen (asynchronous time-warp, positional time-warp)
  - Vermeidung von Kamerabewegungen

VR: Echtzeitaspekte

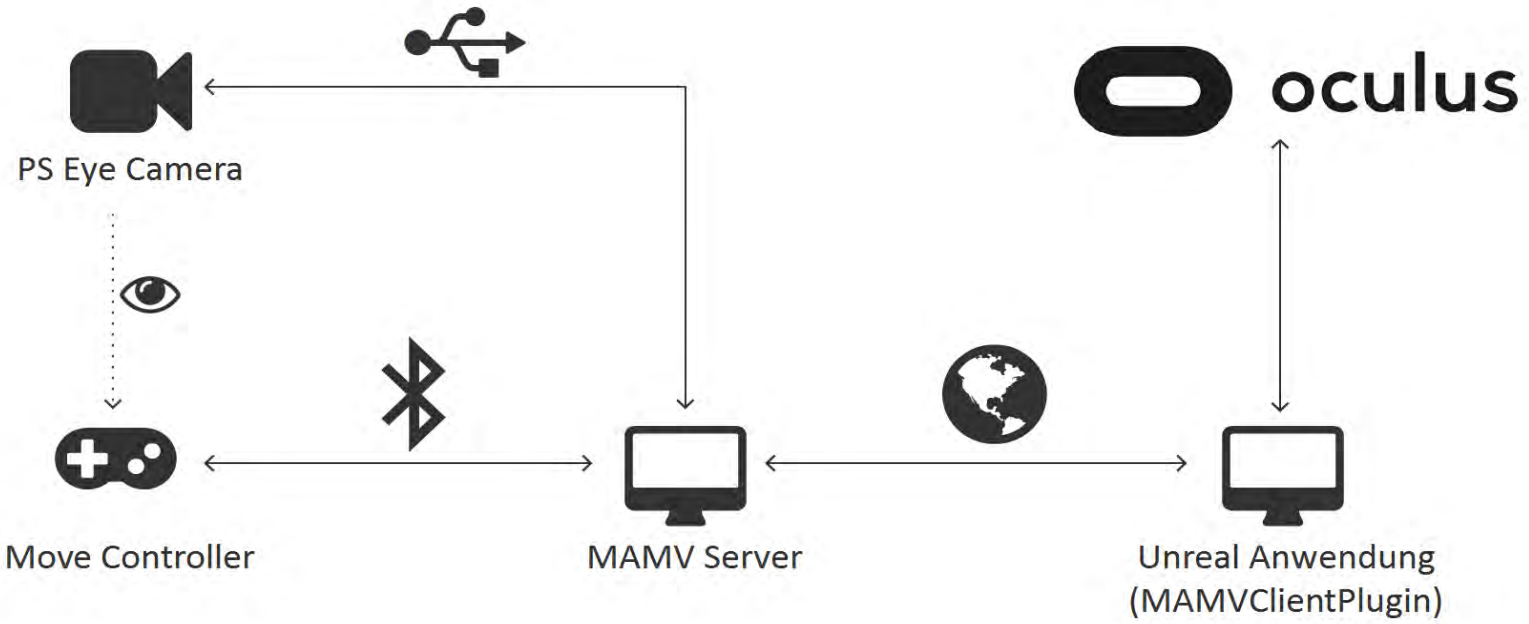
- Anforderungen ähnlich wie bei Computerspielen/3D Rendering
- Aber: wegen Motion Sickness deutlich strikter
- Aktuelle Referenzwerte:
  - Auflösung: 2160x1200 Pixel
  - Bildraten: 90 FPS / 11ms pro Bild
  - Latenz im Gesamtsystem: < 20ms
  - Regelmäßige Frames





Praktischer Teil der Arbeit

- „Interaktion in Virtual Reality – An Application with 6 DoF Input“
- Keine günstig verfügbaren Controller (03.2015)
- Integration von günstigen Motion Controllern in eine Game Engine (Unreal Engine 4) mittels einer Beispielanwendung
- Ziel: funktionierendes System, aus verfügbaren Komponenten (SW + HW)



Hardware

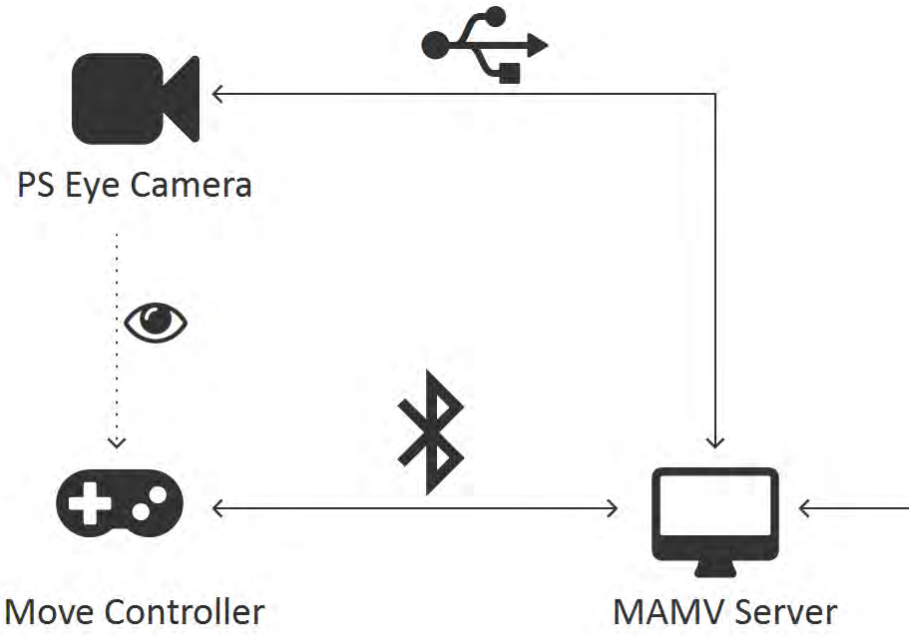
- Controller: Playstation MOVE Set bestehend aus:
  - PS Eye Kamera (60 FPS VGA via USB2)
  - „*Move*“ Motion Controller (Bluetooth)
- Oculus Rift DK2 (DevKit 2)

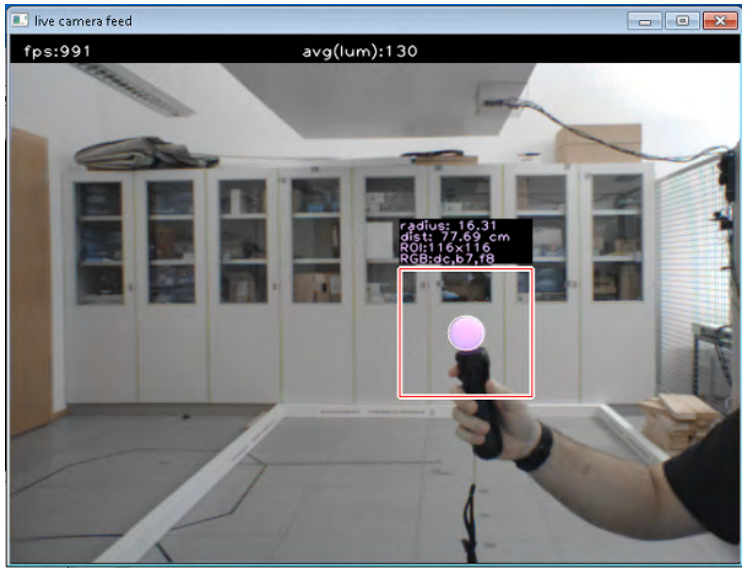
Herausforderungen

- Entwickeln einer VR-Beispielanwendung mit Motion Controller Support
  - VR-Rendering
  - Art der Interaktion
  - Integration der Controller in eine bestehende Game-Engine
- Verwenden der Move Controller als Motion Controller
  - Bluetooth-Pairing?
  - Abfrage von Informationen (Buttons, Sensoren)
  - Anbindung der PSEye Kamera (32bit-Treiber)
  - Tracking der Controller
- Integration der Move Controller in das Koordinatensystem der VR-Brille

Software: MAMV Server

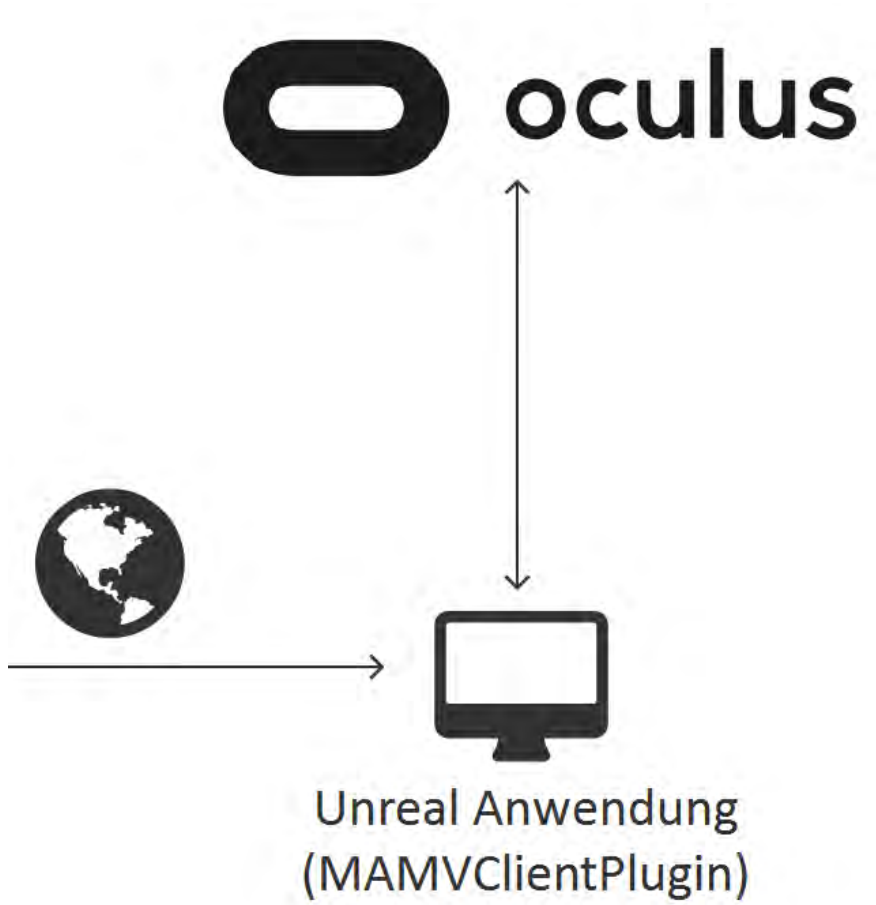
- Tracking-Server mit TCP/UDP API
- „MAMV Server“ entwickelt mit Simon Köllnberger [2]
- Nutzt das Open-Source Projekt „*PS Move API*“ [3] für:
  - Bluetooth Pairing
  - Abfragen von Controller-Status
  - Positionstracking: Optisch
  - Orientierungstracking: Inertial
  - Kalibrierung (Kamera/Controller)





Software: Unreal Engine Plugin

- Client für MAMV Server. Stellt MAMV API in der Unreal Engine 4 zur Verfügung.
- Integriert über UE Plugin System
- Netzwerkkommunikation über Background Thread
- Implementierung der InputDevice API
- Später: Implementierung der UE4 Motion Controller API



Software: Unreal Anwendung

- Präsentation einer virtuellen Maschine und einer Steuerung für diese
- Einfache mechanische Interaktion über Bedienelemente (Slider/Buttons)
- Kollisionsvermeidung durch Integration der Maße/Geometrie des Anwendungsraumes

Fazit

+ Günstiges System

+ Open Source Komponenten

- Genauigkeit des Trackings (insbesondere auf der Z-Achse)

- Komplexität des Stacks (Bluetooth Pairing / Move Controller)

- Verwendung einer einzelnen Kamera

- PS-Eye (Lichtempfindlichkeit, Auflösung, Treiber)

- Dokumentation und Einarbeitung PS Move API und Unreal Engine

(optional) Aktuelle Entwicklungen in der VR

- Playstation VR
- Controller: Oculus Touch, STEM
- Reduzierung von Leistungsanforderungen
- Entwicklung spezieller Hardware und Softwaretechniken um VR performanter zu machen
- „Magic Leap“

HOCHSCHULE LANDSHUT

Fragen?

Quellen

[0] Sebastian Thomeczek, Interaction in Virtual Reality - An Application with 6DoF Input, Fakultät Informatik, HAW Landshut, 2015

[1] Thomas Perl, Cross-Platform Tracking of a 6DoF Motion Controller, Institut für Softwaretechnik und Interaktive Systeme, TU Wien, 2013. http://publik.tuwien.ac.at/files/PubDat_214197.pdf (abgerufen am: 26.7.2016)

[2] Simon Köllnberger, "Bereitstellung von Tracking-Daten von Human Interface Devices und Aufwertung dieser Daten", Fakultät Informatik, HAW Landshut, 2016

[3] http://thp.io/2010/psmove/

Bildnachweise:

- PSMove Controller+Kamera, http://www.gamespot.com/articles/playstation-move-was-ahead-of-its-time-will-be-imp/1100-6421842/
- PSMove API Tracker, aus [0].
- OSVR Explosion: http://www.gamespot.com/articles/razer-reveals-open-source-vr-headset-the-osvr/1100-6424485/
- Vive controller: http://www.roadtovr.com/wp-content/uploads/2016/02/HTC-Vive-Consumer-Launch-SteamVR-controllers.jpg
- Lighthouse base station: http://www.roadtovr.com/wp-content/uploads/2015/03/DSC0060.jpg
- Oculus Constellation: https://de.ifixit.com/Teardown/Oculus+Rift+Constellation+Teardown/61128
- Oculus Rift, Gear VR: https://www.oculus.com/en-us/
- Oculus Rift DK2: Oculus Rift DK2 by Ats Kurvet - Own work. Licensed under CC BY-SA 4.0 via Commons – Source: https://commons.wikimedia.org/wiki/File:Oculus_Rift_development_kit_2.jpg
- HTC Vive: https://www.htcvive.com/de/
- Oculus Touch: https://vrworld.com/2016/09/27/oculus-touch-priced-usd-199/
- Project Cars: http://www.theriftarcade.com/project-cars-patch-3-0-improves-oculus-rift-support/
- Performance HUD: https://developer3.oculus.com/documentation/pcsdk/latest/concepts/dg-hud/
- Hololens Globus: https://www.microsoft.com/microsoft-hololens/en-us