Standup Weckersystem

Computer Vision Projekt
HSHL – WS 2012/13
CVIC02
20.12.2012

Projektbeschreibung

- StandUp ist ein **Weckersystem** mit innovativer Mensch-System-Schnittstelle.
- Das System bietet die die Funktion ein audiovisuelles Wecksignal zu einer vom Nutzer festgelegt Zeit zu senden.
- Der Nutzer muss eine definierte **Bewegung** vor der zugehörigen Kamera durchführen, um das Wecksignal zu unterbrechen.
- Anhand der Videoinformation kann das System erkennen, ob der Nutzer aufgestanden ist.
- Die Uhrzeit wird grafisch dargestellt.

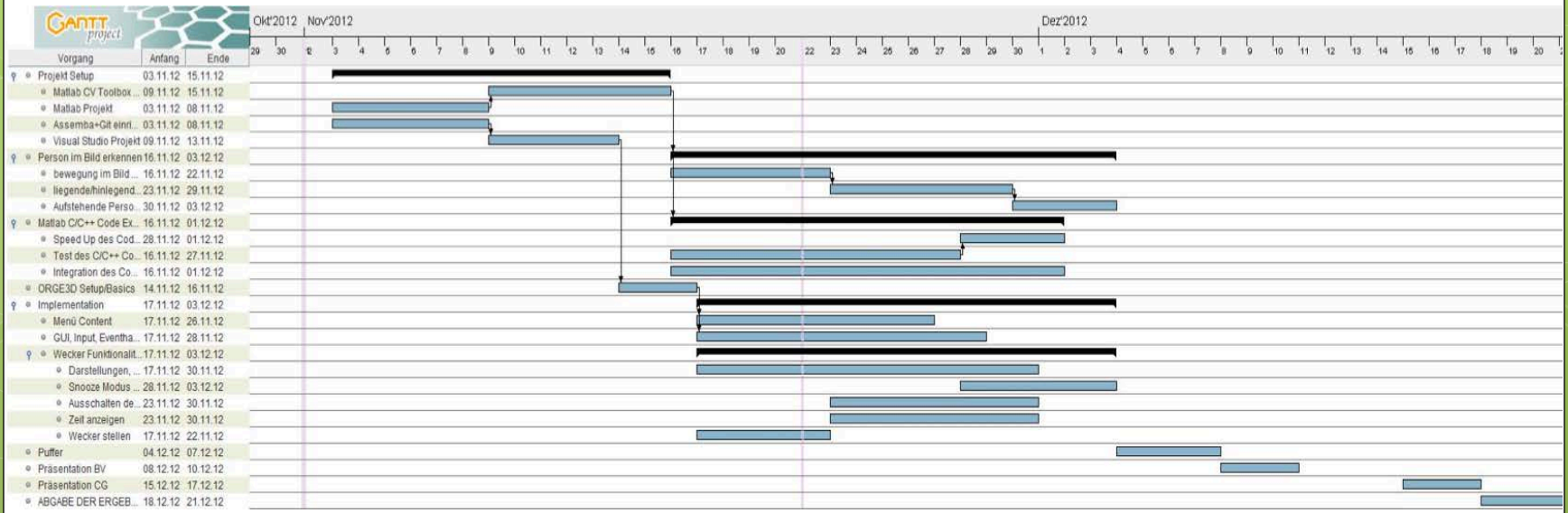
Projektmanagment
Das Team

- Hans Ferchland (Projektleiter)
- Roman Hillebrand
- Hady Khalifa

- Flache Gruppen-Hierarchie
- Kommunikation auf persönlicher Ebene

Projektmanagment
Projektstrukturplan

- Definition von 26 Teilaufgaben und deren Verantwortlichkeiten
- Abschätzung der Dauer + Puffer



Visual Computing I – VCIC02
Hans Ferchland - 20.12.2012

Grün

Ergebnisse seit letztem Statusbericht
Milestone: Abgabe
Wecker-Funktionalität optimiert durch GUI Steuerelement
Computergrafik: Optimierung GUI Layout + Animation, Darstellung Restschlafzeit
Bildverarbeitung: GUI implementiert
„menschliche Schnittstelle“ zwischen Matlab und C++
Komplett Kommentierter Code + Dokumentation

Nächste Aufgaben / nächster Meilenstein
Matlabskript in C++ Übersetzen

Arbeitspakete	Bemerkungen	Status
Bildverarbeitung	GUI implementiert	Grün
Computergrafik	Optimierung GUI Layout + Animation	Grün
	Wecker-Funktion an GUI	Grün

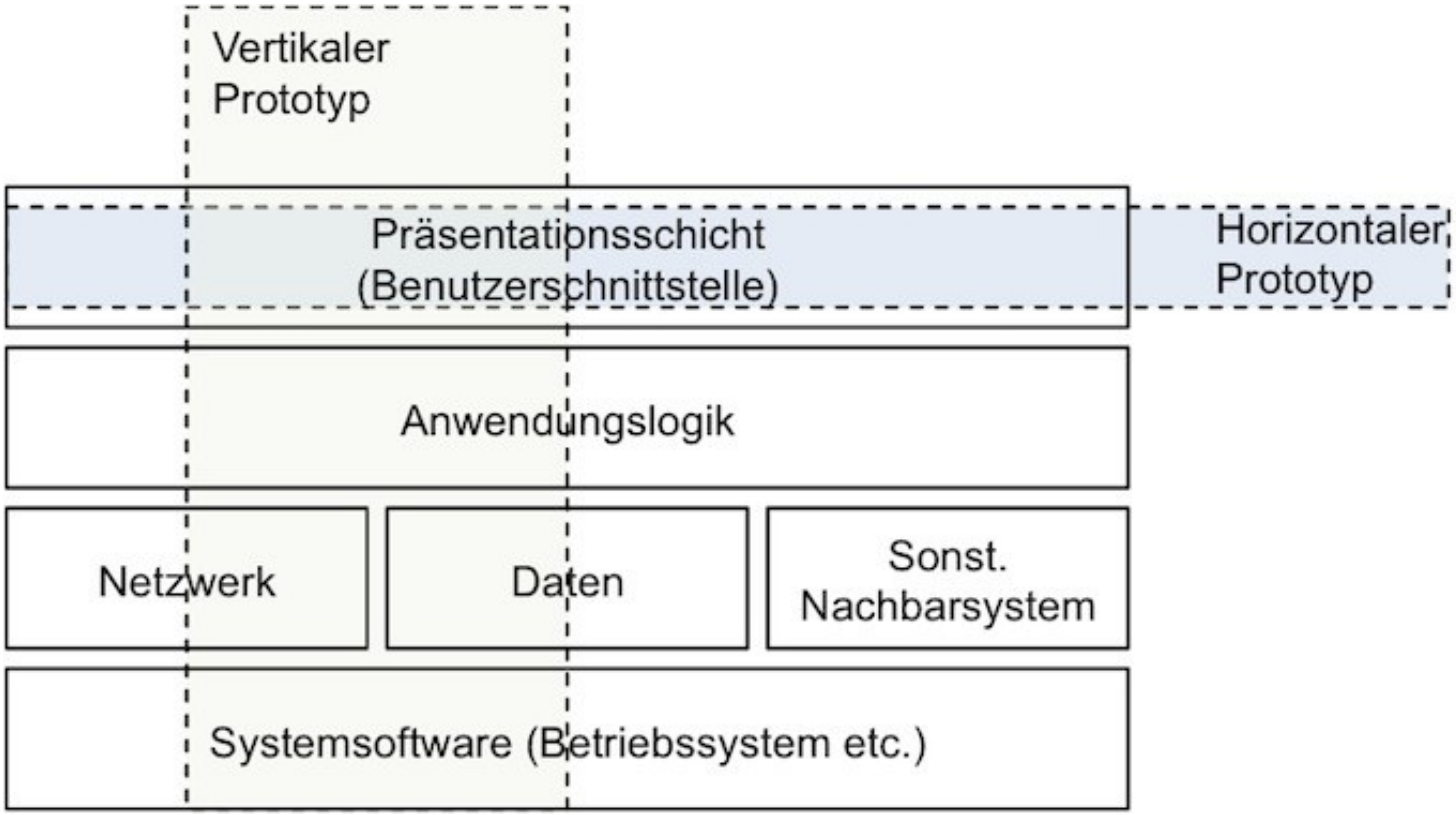
Herangehensweise

1. Erstellen eines GIT-Repository zum Versionsmanagement
2. Anlegen des Visual Studio Projekt und Einbinden der Bibliotheken
3. getrennte Bearbeitung der einzelnen Teilaufgaben und regelmäßige Integration
4. Zuvor definierte Abnahmetests durchführen

Vorgehensmodell

- im Bereich Computergrafik wurden nach dem Anlegen des Projekts horizontale Prototypen entwickelt
  - GUI + Animation (CEGUI)
  - Wecker -System
  - Kamera Simulation (Schnittstelle zu Matlab)
  - Uhr Visualisierung (OGRE3D)
- Regelmäßige Integration der Prototypen

Vorgehensmodell (Horizontal) Prototyping



Algorithmen und Strukturen

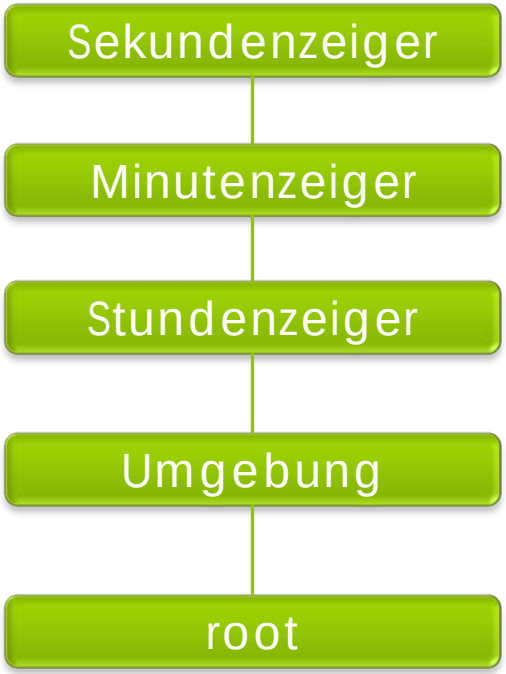
Computergrafik

Bibliotheken

- Ogre3D wird als Grafikbibliothek verwendet.
  - Ogre3D benötigt die boost-Bibliothek für einige Funktionen.
- Für die Darstellung der GUI wird CEGUI verwendet.
  - Die für den gleichzeitigen Einsatz von OGRE und CEGUI nötige Bibliothek wurde aus den Quelldateien erzeugt.
- CEGUI verwendet freetype für die Darstellung von Schriften und zlib für das Laden von Bilddateien.

Szenegraph

- Der Szenegraph der dreidimensionalen Uhr-Darstellung ist, vereinfacht dargestellt, wie folgt aufgebaut:



Render-To-Texture (RTT)

- Render-to-texture wird eingesetzt, um die Balkendarstellung der Uhr auf einer frei im Raum positionierbaren Fläche zu ermöglichen.
- Dazu wird der zur Balkendarstellung gehörige Szenegraph nicht, wie üblich, in den Backbuffer geschrieben, sondern in die Rendertextur.
- Diese kann anschließend auf ein Objekt im GUI-Szenegraphen abgebildet werden.

Szenegraph - Transformation

- Die Transformation ist eine Drehung um einen gemeinsamen Mittelpunkt.
- Da die Drehung der Objekte weiter unten im Szenegraph von der der weiter oben liegenden abhängt, ergibt sich ein schöner dreidimensionaler Effekt.

Cubemaps 1/2

- Für eine schönere Darstellung der dreidimensionalen Uhr werden für die Oberflächen der Uhr Cubemap-Reflexionen verwendet.

- Eine Cubemap kann man sich als ein Konstrukt aus 6 Texturen vorstellen. Diese Texturen sind an Seitenflächen eines Würfels angebracht.

- Die Cubemap wird nicht, wie bei einer einfachen cubemap üblich, mit Hilfe von u-v-Koordinaten abgetastet, sondern über einen Vektor mit 3 Komponenten.
- Dieser Vektor zeigt von der Mitte des Würfels nach außen. Die Stelle, an der dieser Vektor den Würfel schneidet, markiert den Bildpunkt, der abgetastet wird.

Cubemaps 2/2

- Für den Zweck der Darstellung einer Reflexion an einer Oberfläche wird in der Cubemap ein 360°-Panorama der Umgebung gespeichert.
- Der Vektor von der Kamera zu dem zu zeichnenden Punkt auf der Oberfläche wird an dem Normalenvektor der Oberfläche gespiegelt.
- Das Resultat wird als Parameter für die Abtastfunktion der Cubemap verwendet.

Skybox

- Die Skybox stellt weit entfernte Objekte als Texturen auf einem Würfel dar, dessen Mittelpunkt sich immer genau auf dem Betrachtungspunkt befindet.
- Für den Betrachter erzeugt dies den Eindruck einer dreidimensionalen Darstellung des Hintergrunds.
- Eine Skybox ist eine sehr performante und einfache Möglichkeit, einer Szene einen Hintergrund zu geben.
- Zusammen mit den Cubemap-Reflexionen kann ein sehr realistisches Bild erzeugt werden.

Stencil Shadows 1/3

- Um die Schatten der Uhr auf dem Boden sowie die Schatten von Komponten der Uhr auf darunterliegenden Komponenten darstellen zu können, wird der "stencil shadow"-Algorithmus verwendet.
- Der Algorithmus setzt voraus, das alle schattenwerfende Objekte eine geschlossene Hülle haben.
- Zunächst werden die Konturen des Schatten werfenden Objektes(der Uhr) vom Betrachtungsstandpunkt der Lichtquelle aus gefunden.

Stencil Shadows 2/3

- Wenn die Skalarprodukte der Normalenvektoren zweier benachbarter Polygone "n1" und "n2" mit dem Vektor von einem Punkt auf der Kante zwischen den Polygonen zur Position der Lichtquelle "l" zwei Resultate mit unterschiedlichem Vorzeichen liefern, wird diese Kante als zur Kontur gehörig gezählt.
- Die entstehende (geschlossene) Konturlinie wird nun von der Lichtquelle weg extrahiert und vorne und hinten verschlossen, um ein Schattenvolumen zu erzeugen.

Stencil Shadows 3/3

- Um dieses Volumen sichtbar zu machen, wird die Szene in folgenden Schritten gerendert ("depth fail"-Methode)

1. Die Szene wird gezeichnet, als befänden sich alle Oberflächen im Schatten.
2. Schreiben in den Farb-und Tiefenpuffer wird deaktiviert, Schreiben in die Stencil-Maske wird aktiviert.
3. Front-face-culling wird benutzt.
4. Die Stencil-Operation wird auf "increment on depth fail" gesetzt.
5. Das Schattenvolumen wird gezeichnet.
6. Back-face-culling wird benutzt.
7. Die Stencil-Operation wird auf "decrement on depth fail" gesetzt
8. Nochmaliges Zeichnen des Schattenvolumens.
9. Die Szene wird nocheinmal gezeichnet (diesmal mit Licht), die in den Stencilbuffer geschriebenen Werte werden dabei als Maske verwendet

Algorithmen und Strukturen

Bildverarbeitung

Gesichtserkennung mit CV System Toolbox

- bild, video, webcam
- vision.VideoFileReader()
- vision.VideoPlayer()
- vision.CascadeObjectDetector('UpperBody')
- vision.CascadeObjectDetector('Nose')
- step(noseDetector,videoFrame)

Optical Flow Analysis

- Von Matlab-Hilfe angepasstes Beispiel
- http://www.mathworks.de/de/help/vision/examples/tracking-cars-using-optical-flow.html

Bilddifferenz in Grauwertbild (Webcam)

- Bewegung (Änderungen/Differenzen) erkennen
- Bewegung in verschiedenen Bildbereichen

Bilddifferenz im Kantenbild (Webcam)

- Matlab Funktionen imabsdiff(), wiener2()
- gutes Ergebnis, nutzbar!
- prozentualer Bewegungsanteil links, oben und rechts
- Schwellwerte um Bewegung zu identifizieren

Kantenerkennung in der Zeit (Webcam)

- Sobel Filter über 3 Bilder in Folge
- timeFilter() auf Kantenbild
- gutes Ergebnis, aber nicht nutzbar => verworfen

Segmentierung über Boundary Detection

- zunächst auf ganzem Bild
- Abgewandelt auf Kantenbild
- matlab funktionen edge(), imfill(), strel(), bwperim(), bwtraceboundary(), boundaries()
- http://www.mathworks.de/de/help/images/examples/detecting-a-cell-using-image-segmentation.html

Finaler Algorithmus

- Bilddifferenz auf Kantenbild
- Segmentierung des Kantenbildes
- abwägen von segmentierter Bewegung, um eine genauere Entscheidung zu treffen
- GUI zeigt die verschiedene Zustände der Person und erkannte Bewegungsmuster an

Standup Weckersystem

Das Produkt