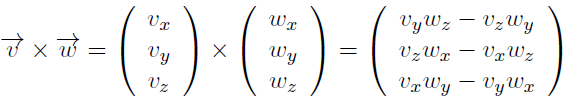
Zusammenfassung Computergrafik

# Einführung

**Vektoren**

* Länge berechnen: für 2D, für 3D
* **Normierung**: Verändern eines Vektors, sodass er die Richtung beibehält und die Länge 1 bekommt. Beispiel:
  1. Bsp.
  2. Länge berechnen:
  3. Normieren:
* **Ortsvektoren** haben ihren Startpunkt im Ursprung: Sie beschreiben eine Position im Raum
* **Richtungsvektoren** beschreiben eine Richtung und sind positionsabhängig
* **Skalierung:** Das Skalarprodukt zweier Vektoren liefert das **Skalar**
  1. Beispiel:
  2. Winkels zwischen Vektoren berechnen:  
      oder:  
      .
* **Orthogonalität:** Ein Vektor steht senkrecht auf einem anderen Vektor
  1. Beispiel 2D: , dann ist senkrecht zu .
  2. Im 3D verwendet man das **Kreuzprodukt**
* **Kreuzprodukt**
  1. Beispiel: 
  2. Winkel zwischen Vektoren berechnen:  
     (ähnlich wie beim Skalarprodukt – nur mit dem Sinus des Winkels)
  3. Die Länge des Ergebnisvektors entspricht der Fläche von den Vektoren aufgespannte Parallelogramm

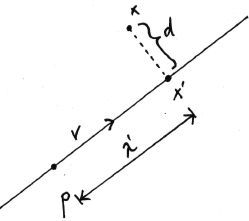
**Normale**

* Vektor der orthogonal auf einer Geraden, Kurve, Ebene oder Fläche steht
* Gerade:
  + Hat Gerade den Richtungsvektor , sind die beiden Vektoren und die Normalvektoren
  + Ist die Gerade durch gegeben, ist der Vektor ein Richtungsvektor der Geraden und und sind Normalvektoren, die Steigung jeder Normalen ist
  + Ist die Gerade durch gegeben, ist ein Normalvektor
* Ebene
  + Ist die Ebene durch gegeben, ist ein nach oben weisender und ein nach untern weisender Normalvektor
  + Kreuzprodukt zum Berechnen des Normalvektors
  + Ist die Ebene durch gegeben, ist ein Normalvektor
* Kurve
  + Orthogonal zur Tangente eines bestimmten Punktes auf der Kurve
  + Allgemeine Funktion der Tangente:

Dreiecks**normale**

* Kreuzprodukt eines Dreiecks mit den Punkten und berechnen
  + mit , …

**Abstand Punkt-Gerade**

* Es wird angenommen, dass der Richtungsvektor normiert ist:
* Berechnung des Abstands eines Punktes zu einer Geraden
  + Punkt bestimmen, der senkrecht auf der Geraden zu Punkt projiziert ist  
    
  + und und damit

**Implizite Funktion**

* Unter welcher Bedingung definiert eine Gleichung implizit eine Funktion   
  , für die gilt?
* beschreibt den Einheitskreis in der Ebene

# Polygonale Netze

**Matrizen**

* **Transponierte** Matrix:
* **Inverse:** Es gilt: (Identitätsmatrix: 1 auf der Hauptdiagonalen, sonst 0)
* **Skalierungsmatrix:** Die Skalierungsfaktoren für die Dimensionen sind auf der Hauptdiagonales, sonst 0
  + Beispiel: Keine Skalierung in x-Richtung und eine Skalierung um 100% in y-Richtung

# Transformationen

**Koordinatensysteme**

* **Homogene** Koordinaten: Zusammenfassung von Transformation und Translation in einer Matrix durch Verwendung homogener Koordinaten
  + Alle Vektoren um eine Zeile und alle Matrizen um eine Zeile und eine Spalte erweitert. Die zusätzliche Komponente heißt -Komponente
  + Berechnung eines Vektors in einen homogenen Vektor:
  + Die Zusätzlichen Werte einer Matrix werden mit 0 aufgefüllt, bis auf die Hauptdiagonale, die mit 1 aufgefüllt wird:
  + Translationsmatrix um den Vektor :

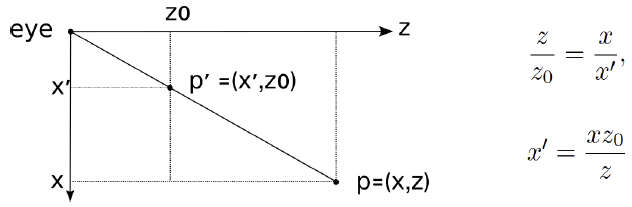
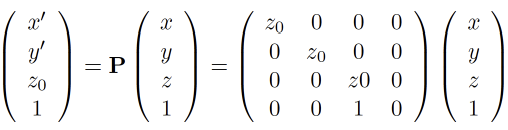
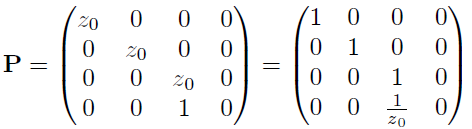
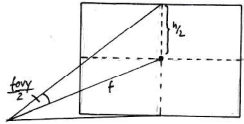
**Basisvektoren**

* Basisvektoren spannen den Vektorraum auf und repräsentieren die Achsen
* Kombiniert man Basisvektoren miteinander, kann man Punkte im Vektorraum erreichen (**Linearkombinationen**)
  + Im Vektorraum, der durch die Basisvektoren und aufgespannt wird, bedeutet der Punkt also eigentlich:

**Transformation zwischen beliebigen Koordinatensystemen**

* Um von einem Koordinatensystem in ein Koordinatensystem zu transformieren, geht man den Weg über das kartesische Koordinatensystem
  + Vektor in das kart. Koordinatensystem überführen:

**Kameratransformation**

* **Model-View-Projektion**
  + **Model**: Objekt-Koordinaten 🡪 Welt-Koordinaten
  + **View**: Welt-Koordinaten 🡪 Kamera-Raum
  + **Projektion**: Kamera-Raum 🡪 Bildschirm
  + Also: Objekt 🡪 Welt 🡪Kamera 🡪 Bildschirm
* Beispiele
  + **Model**:
    - Punkt eines Objekts soll in Welt-Koordinaten überführt werden
    - Dann ist
    - Die Modelmatrix setzt sich aus einer Translation und einer Rotation (-45° um die z-Achse) zusammen
    - Nach der Transformation ergibt sich:
  + **View**:
    - Bestimmen der View-Matrix zur View-Transformation
    - Aus den Parametern der Kamera wird ein Koordinatensystem mit folgenden Achsen bestimmt:
      * x (Kreuzprodukt)
      * x (Kreuzprodukt)
    - Die drei Koordinatenvektoren werden als Spalten in eingetragen
      * Für die Transformation einen Objektpunktes ergibt sich
      * Also berechnet sich ein Punkt in Kamerakoordinaten:
    - Nach der View-Transformation lassen sich die *near* und  *far* Ebenen durch die Entfernungswerte (*z*-Werte) beschreiben
  + **Projektion**:
    - **perspektivische** Transformation:  
        
        
      mit als Zentralprojektionsmatrix  
      ,   
      mit als perspektivische Projektionsmatrix
      * Punkt auf der Bildebene berechnen:
      * Um homogene Koordinaten zu bekommen muss dann noch durch (im Beispiel: ) geteilt werden
    - **Pixel**-Transformation
      * Der Punkt (0|0) befindet sich in der unteren linken Ecke des Bildschirms mit Breite x Höhe Pixeln
      * Das Verhältnis (Aspekt) berechnet sich durch
      * Aus dem Öffnungswinkel oder ergibt sich dann die Matrik
      * Die fokale Länge berechnet sich aus oder  
        
      * Normalerweise gilt:
      * Der Punkt berechnet sich dann:

# Kurven

**Basisfunktionen**

* Basis besteht nicht aus Vektoren (Basisvektoren) sondern aus Funktionen (Basisfunktionen)
* Es kann nicht nur mit Skalaren (Basisvektoren) sondern mit Punkten beliebiger Dimensionen gerechnet werden

# Beleuchtung und Texturen

# Animation

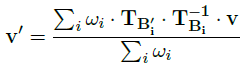
**Skelett** und **Knochen**:

* Mit jedem Knochen wird ein Koordinatensystem assoziiert
* Vertex zusammen mit einem Knochen bewegen
  + Transformation von in Ruhelage in das Koordinatensystem von :  
    - beinhaltet alle vorangegangenen Transformationen aller Knochen weiter oben im Skelettbaum und die Rotation von
  + Bei einem Knochen lassen sich nun alle Vertices des Dreiecksnetzes zu jedem Zeitschritt transformieren
  + Bei mehreren Knochen gibt es zwei Ansätze
    - **Nächster** Knochen
    - **Gewichtete** Knochen

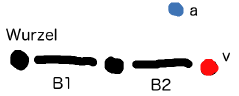
**Nächster** Knochen:

* Der Knochen wird verwendet, der dem Vertex am nächsten liegt
* Knoch mit Startpunkt und Endpunkt :
  + als Geradengleichung
  + Die Punkte des gesuchten Knochens liegen im -Interwall
  + Oftmals wird der Richtungsvektor der Geradengleichung normiert
    - Nun liegen die Punkte des Knochens im -Interwall
    - Für erreicht man , für erreicht man
    - Für wird mit weitergearbeitet, analog für die obere Grenze

**Gewichtet** über alle Knochen

* Jeder Vertex wird nicht nur auf einen Knochen festgelegt, sondern auf alle
* Je näher der Knochen, desto größer die Gewichtung für diesen Knochen  
  
  + Nacheinander skaliert man alle Knochen mit der Gewichtung und addiert dann auf; am Ende wird durch die Summe aller Gewichte geteilt

**Inverse Kinematik**

* Beispiel: Um eine „Gehen“-Animation zu erstellen, setzt man nur die Position der Hände und Füße, alle weiteren Parameter werden mittels inverser Kinematik bestimmt  
  
* Alle Transformationen der Gelenke und Knochen vom Rumpf bis zur Hand werden mit dem Vektor multipliziert

# Nützliche Transformationen

*Jeweils angegeben im 2D für x- und y-Achse, im 3D für x-, y- und z-Achse*

**Drehung:**

* im 2D:
* im 3D:

**Spiegelung:**

* Im 2D: ,
* Im 3D:

**Streckung:**

**Scherung:**

* im 2D: ,
* im 3D:

**Projektion**

# Weitere Begriffe

verschiedene **Puffer**,

**Sichtbarkeit**, **Transparenz**, **Baryzentrische Koordinaten**, **Triangulation**, **Phong**, **Nachbarschaft** (Datenstruktur), **Halbkanten** (Datenstruktur), **Laplace**-Glättung, **Gauß-**Glättung, **Interpolation**, **Bezier- & Hermite**-Kurven, **Interpolation**, **Approximation**, **Konvexe Hülle**, **Splines**, **Tensorproduktflächen**, **Culling**, **Hüllkörper**, **Schnitte**, **Octree** (Datenstruktur), **Partitionierung**

# Datenstrukturen

**Octree (B-Baum bei B Dimensionen)**

* Baum, dessen Knoten entweder acht (B beim B-Baum) direkte Nachfolger oder gar keinen Nachfolger haben
* Hierarchische Untergliederung dreidimensionaler Datensätze
* Nur der Wurzelknoten bezeichnet einen Octree (B-Baum) der Tiefe 0

**Gerade / Strahl**

* Eine Gerade wird durch die Punkte und bestimmt
* Der Richtungsvektor ergibt:
* Daraus ergibt sich die parametrische Geradendarstellung: oder  
   ( lässt sich schnell mit berechnen)
* Alle Punkte, die sich mit einem Wert konstruieren lassen, sind auf der Geraden
* Alle Punkte, die sich mit einem Wert mit konstruieren lassen, sind auf dem **Strahl**

**Ebene**

* Eine Ebene wird beschrieben durch die Punkte ,r
* Berechnen der aufspannenden Vektoren und
* Parametrische Ebenen-Darstellung:
* Alternative Repräsentation (Hesse-Normalform): mit als Normalvektor der Ebene in Einheitslänge ist und
* Den Normalvektor lässt isch aus dem Kreuzprodukt aus und berechnen
* liefert einen positiven Wert, falls sich auf der Seite der Ebene befindet, in die der Normalvektor zeigt, sonst einen negativen