

# 1 Koordinatensysteme

- Ursprung
- 3 Einheitsvektoren
- Weltsystem
- Objektsysteme
- Kamerasystem, Blickrichtung  $-z_c$

## 1.1 Kamera Positionierung

- Augpunkt  $e = \text{Ursprung}$
- Blickrichtungspunkt  $c \rightarrow e-c = z_c$  (nicht immer horizontal)
- $y_c$  wird am vertikalem  $y_w$  ausgerichtet  $\rightarrow x_c = y_w \times z_c; y_c = z_c \times x_c$
- Problem bei vertikaler Blickrichtung

## 2 Display Pipeline

- Abbildung Objektsystem ins Weltsystem
- Abbildung Weltsystem ins Kamerasystem
- Projektion
- Clipping/Division
- Abbildung ins Bildkoordinatensystem

## 3 Rotationen

### 3.1 Transformation

$p = M p' + t$  überführt Punkt  $p'$  bzgl. Koordinaten  $S'$  in  $p$  bzgl. Koordinaten  $S$ .

**inverse:**  $p' = M^{-1}(p-t)$ , bei orthogonalen:  $M^{-1} = M^T$   
es gilt:  $M = (x1, y1, z1); t = o1$  falls  $x1, y1, z1, o1$  die Koordinaten von  $S'$  bzgl  $S$

Bei lokalen Koordinaten  $\rightarrow$  Welt reicht also einfach  $M$  aufschreiben, Bei Welt  $\rightarrow$  Kamera muss Umkehrabbildung gebildet werden.

#### 3.1.1 Rotationsmatrizen

um x-Achse  $\rightarrow (0, 0) = 1, (1, i) = 0, (i, 1) = 0$

restliche:  $\begin{matrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{matrix}$  jeweils entsprechend verschoben

#### 3.1.2 Mehrere Transformationen

Die (4x4) oder (3x3) Matrizen miteinander multiplizieren!  
Dabei steht der ursprüngliche Punkt ganz recht, links davon die erste Matrix, links von den beiden die zweite Matrix usw.

**Rundungsfehler** - Orhogonalisierung:

- $m1' = m1/||m1||$
- $m3' = m1' \times m2/||m1' \times m2||$
- $m2' = m3' \times m1'/||m3' \times m1' ||$

### 3.2 Euler

- Pitch - Querachse ( $x$ )
- Yaw - Hochachse ( $y'$ )
- Roll - Längsachse ( $z''$ )

Drehung immer um die gedrehten Achsen.

Da  $R_{y'}(\beta) = R_x(\alpha)R_y(\beta)R_x(-\alpha)$  gilt:

Euler-Darstellung ist mit Festwinkel Darstellung in umgekehrter Reihenfolge identisch:

$$R_{z''}(\gamma)R_{y'}(\beta)R_x(\alpha) = R_x(\alpha)R_y(\beta)R_z(\gamma)$$

#### 3.2.1 Interpolation

Jede Rotation wird durch **eine** Achse (Eigenvektor der Rotationsmatrix!) und einen Winkel repräsentiert

- gemittelte Achse finden:
- Achse die orthogonal zu den beiden zu interpolierenden Achse ist finden  $b = a1 \times a2$
- $a1$  um den Winkel  $t * \cos(a1 \circ a2)$  um  $b$  drehen:  $a = R_b(t * \cos(a1 \circ a2))a1$
- Winkel auch linear interpolieren  $\theta = t\theta_1 + (1-t)\theta_2$

### 3.2.2 Rotation um bel. Achse

- Rotieren der normierten Drehachse  $b$  in  $x - z$ -Ebene (um  $z$ ) ( $\cos \theta = b_x, \sin \theta = b_y$ )
- Rotieren der Drehachse in auf die  $z$ -Achse (um  $y$ ) ( $\cos \theta = b_z$ )
- Rotieren um die  $z$ -Achse um Winkel  $\alpha$
- Rückgängigmachen von 1. und 2.

### 3.3 Festwinkel

- Die Achsen um die Rotiert wird ( $x, y, z$  oder  $x, y, x$ ) bleiben fest.
- Das Objekt dreht sich
- **Gimbal Lock:** Dreht man das Objekt nicht um die erste Achse, bei der zweiten Achse aber so um  $90^\circ$  dann ist es so dass es gerade mit der lokalen Achse die man ja schon beim erstem Mal nicht drehen wollte so zu liegen kommt, dass es bei der 3. festen Achse nur noch um diese lokale Achse zu drehen ist.

#### 3.3.1 Matrix in Festwinkel zerlegen

(abc) = Spaltenvektoren der Matrix.

- eine Drehung suchen um  $a$  in die  $x - z$  Ebene zu bringen. (um  $z$ )
- Dann  $a$  auf  $x$ -Achse drehen. (um  $y$ )
- Dann  $b$  auf  $y$ -Achse drehen (um  $x/a''$ )

Gesucht ist nun die Umkehrabbildung, vor alle Winkel ein Minus.

### 3.4 Linearkombination von Rotationen

Problem bei Interpolation von Orientierungen: **fehlende Kommutativität**

- **Lösung:**  $A @ B = \lim_{n \rightarrow \infty} (A^{1/n} B^{1/n})^n$
- so kann man 3 Rotationen um  $x, y,$  und  $z$ -Achse ohne Gimbal-Lock linear kombinieren.
- z.B.  $R_x(\alpha) @ R_y(\beta) = \lim (R_x(\alpha/n) R_y(\beta/n))^n = R_v(\gamma)$   
 $\gamma = \sqrt{\alpha^2 + \beta^2}; v = (\alpha/\gamma, \beta/\gamma, 0)$

## 4 Quaternionen

### 4.1 Quaternionen

4-Tupel  $[s, v]$   $v = (x, y, z)^T$

$$R_a(\theta) = [\cos(\theta/2), \sin(\theta/2)(a)]$$

In der Regel Einheitsquaternionen  $Q/||Q||$

Rotation von  $a$  dann:  $a' = Q a Q^{-1}$

#### 4.1.1 Rechengesetze

- **Addition:** Komponenten addieren
- **Multiplikation:** Wie komplexe Zahlen:  
Realteil: beide reall. multiplizieren - skalarprodukt der imaginärteile abziehen,  
Vektorteil =  $\text{real1} * \text{vektor2} + \text{real2} * \text{vektor1} + \text{kreuzprodukt der vektorteile}$
- **inverses:** Imaginärteil Vorzeichen umkehren und alles durch  $\text{betrag}^2$  dividieren
- **betrag<sup>2</sup>:**  $\text{Realteil}^2 + \text{vektorteil}^2 = s^2 + x^2 + y^2 + z^2$

#### 4.1.2 Quaternion Matrix

$$Q = [s, x, y, z], s^2 + x^2 + y^2 + z^2 = 1$$

(3x3): Diagonale summiert sich zu  $4s^2 - 1$

alle Informationen zum Quaternion ausser Vorzeichen kann man aus Diagonale errechnen

### 4.2 Quaternionen-Kurven

Naheliegend: Einfach Bezier oder deBoor-Algorithmus mit Quaternionen als Kontrollpunkten verwenden und **slerp** zum Interpolieren verwenden.

Problem: nicht gleiche Stetigkeitsgrade wie im Euklidischem Raum. Daher andres Verfahren!

#### 4.2.1 Interpolation

Die Interpolation von Einheitsquaternionen ist auf der 4-D Einheitssphäre durchzuführen,

- $\text{Skalarprod}(Q1, Q2) = \cos \alpha$  ( $\alpha$  wird linear interpoliert:)

- $\text{slerp}(Q1, Q2, t) = \frac{\sin((1-t)\alpha)}{\sin(\alpha)} Q1 + \frac{\sin(t\alpha)}{\sin(\alpha)} Q2$
- $\text{slerp}(Q1, Q2, t) = Q1 \exp(t \log(Q1^{-1} Q2)) = \mathbf{Q1} (\mathbf{Q1}^{-1} \mathbf{Q2})^t$

#### 4.2.2 Kurve durch Anf. Pkt und Ableitung

- $\mathbf{p}(t) = p_0 + \int_{t_0..t_1} v(t) dt = \mathbf{p}_0 + \sum_{i:1..n} \mathbf{v}_i \alpha_i(t)$
- $\alpha_i$  **Basisfkt.:** wachsen im Segment i von 0 auf 1, vor Seg. i = 0, nach Seg. i = 1
- $v_i = p_i - p_{i-1}$

Andre Darstellung für  $\alpha_i$ :

- $\alpha_i(t) \sum_{j:i..n} \mathbf{N}_j^2(t)$
- allgemein können Basisfunktionen für konventionelle Kurven mit Punkten in Basisfunktionen für Kurven mit Anfangspkt und Ableitung umgewandelt werden durch aufsummieren.

#### 4.2.3 Quaternionen-Kurvendarstellung

- statt **Summe: Produkt von exp()**;
- statt **Differenzvektor:**  $\omega = \log(Q_{i-1}^{-1} Q_i)$
- Basisfkt:  $\Phi_i(t) = \sum_{j:i..n} \phi_j(t)$

Kurvendarstellung

- $Q(t) = Q_0 \prod_{i:1..n} \exp(\omega_i \Phi_i(t))$

Basisfunktionen z.B. Lagrange-Polynome (interpolierend)

Ergebnis: deutlich kürze Wege bei der Interpolation von Winkeln als mit Euler-Winkeln.

## 5 Pfadverfolgung

### 5.1 Bogenlängenparametrisierung

- Key-Frame-Positionen  $\mathbf{p}_i$
- Interpolation nach Zeit:  $p(t_i) = p_i$
- **Trennung von Geometrie und Zeit!**
- **Bogenlänge:**  $s(t_1) = \int_{t_0..t_1} \|p'(t)\| dt$ , liefert Weg in Abh. von Zeit.  
Geschwindigkeit ist:  $\|p'(t)\|$
- $p(u) = p(s^{-1}(u))$  liefert Position  $(x, y, z)$  in Abhängigkeit von zurückgelegter Distanz  $u$
- Geometrie einer Bahnkurve kann nun unabhängig von der Zeit **zentripetal** parametrisiert werden! in der Regel Approximation der Bahnkurve durch lineare Segmente!

### 5.2 Frenet-Frame

Liefert **natürliche Orientierung** der Objekte die sich auf der Bahnkurve bewegen!

- x-Achse = **Tangente** =  $\frac{p'}{\|p'\|}$
- z-Achse = **Senkrecht zur Krümmung** =  $\frac{p' \times p''}{\|p' \times p''\|}$
- y-Achse in der Krümmungsebene =  $z \times x$

- Bei Orientierungswechsel in Einem Punkt evtl. Orientierung des Frenet-Frame umdrehen um stetig zu bleiben
- Auf linearem Segment (Frenet-Frames der Endpunkte des Segments) interpolieren

## 6 Kinematik

- Kinematik beschreibt **geometrische** Zusammenhänge ohne Berücksicht. d. Kräfte
- **Skelette**
- **Hierarchische Kinematik:** Becken als Wurzelgelenk, jeder Knoten ist ein Gelenkt, jede Kante eine Bindung, Gelenk-Rotationen auf Stack abgelegt, so dass alle Teile unterhalb des Gelenks direkt mitrotiert werden

### 6.1 Denavit-Hartenberg Notation

- z-Achse entspricht der Rotationsachse (bei Drehgelenken)
- Nachfolgende Bindung (hinter Gelenk) soll mit der x-Achse des Gelenks ausgerichtet sein.
- $z_i$  = Rotationsachse des Gelenk i
- $x_i = z_i \times z_{i+1}$

- $y_i$  → Ergänzung zum Rechtssystem
- $\alpha_i$  = Winkel zwischen  $z_i$  und  $z_{i+1} \Rightarrow \cos \alpha_i = z_i \circ z_{i+1}$
- $a_i$  = Distanz auf  $x_i$  bis Schnittpkt mit der Verlängerung von  $z_{i+1}$
- $d_{i+1}$  = Distanz auf  $z_{i+1}$  vom Schnittpkt bis Nullpunkt  $i+1$
- $\theta_{i+1}$  = Gelenkwinkel  $i+1$ , Rotation um  $z_{i+1}$  Achse

**Transformationsmatrix:**

- $M_i^{i+1} = (T_x(a_i) R_x(\alpha_i))(T_z(d_i + 1) R_z(\theta_i + 1))$

## 6.2 Inverse Kinematik

Iterativ: Gesucht: Zusammenhang zwischen Position/Orientierungsänderung eines Endeffektors ( $V = (v, \omega)$ ) und den Gelenkwinkel-Änderungen  $\Delta\theta_i$

Es gilt:  $(p, \alpha) = \text{Funktion}(\theta) \Rightarrow V = (\Delta p, \Delta \alpha) = \frac{\delta \text{Funktion}}{\delta \theta} * \Delta \theta$   
 $\frac{\delta \text{Funktion}}{\delta \theta} = J(\theta) = \text{Jacobi-Matrix}; V = J(\theta) * \Delta \theta$

### 6.2.1 Berechnung der Einträge der Jacobi-Matrix

in 2D:

$$\begin{pmatrix} \delta x / \delta \theta_1 & \delta x / \delta \theta_2 & \dots \\ \delta y / \delta \theta_1 & \delta y / \delta \theta_2 & \dots \\ \delta \alpha / \delta \theta_1 & \delta \alpha / \delta \theta_2 & \dots \end{pmatrix}$$

- q = Position des Gelenkes i
- p = (x,y) = Position des Endeffektors
- a = Gelenk-Achse i
- $\Rightarrow \delta p / \delta \theta_i = \mathbf{a} \times (\mathbf{p} - \mathbf{q})$
- $\Rightarrow \delta \alpha / \delta \theta_i = \mathbf{a}$  (in 2D = 1)

### 6.2.2 Least-Squares-Fitting

Falls es weniger Freiheitsgrade (Gelenke) als Bedingungen (Position+Orientierung die man erreichen soll) gibt!

$$\mathbf{r} = \|\mathbf{J} \Delta \theta - \mathbf{V}\| \rightarrow \min$$

Dazu: partielle Ableitungen von r müssen verschwinden. Also:  
 $2J^T J \Delta \theta = 2J^T V$

### 6.2.3 Lagrange-Verfahren

Falls mehr Gelenke als Orientierungs+Positions-parameter des Endeffektor  $\rightarrow$  'Möglichst Bequeme Stellung' finden  
 $\mathbf{h} := \sum \text{Abweichung}_i \text{ vom Optimalwert}$   
 $\mathbf{r} := \mathbf{h} + \lambda (\mathbf{J} \Delta \theta - \mathbf{V}) \rightarrow \min$   
 $J * \Delta \theta = V$  ist hier die **Nebenbedingung** des Optimierungsproblems die sicher erfüllt wird, h wird nur minimiert.

## 7 Dynamik

Animation mit inverser Kinematik  $\rightarrow$  unrealistische Bewegung, Physikalische Gesetze werden ignoriert.  
(Position, Geschwindigkeit, Beschleunigung)  $\rightarrow$  Kräfte  $\rightarrow$  Beschleunigung  $\rightarrow$  Änderung von Geschwindigkeit, Impuls  $\rightarrow$

### 7.1 Größen

- Position:  $s$  (m)
- Geschw.  $v = s'$  (m/s)
- Beschl.  $a = v''$  (m/s<sup>2</sup>)
- Kraft  $F = ma$  (kgm/s<sup>2</sup>)
- Impuls  $p = mv$  (kgm/s)
- Energie  $E = 1/2 mv^2$
- Winkelgeschw.  $\omega$  (grad/s) (Vektor: Richtung = Drehachse)
- Drehmoment  $\tau$  (kgm<sup>2</sup>/s<sup>2</sup>)
- Drehimpuls  $L$  (kgm<sup>2</sup>/s)

### 7.2 starre Körper

- Körper besteht aus Menge von Massepunkten  $q_i$
- Massezentrum  $c = \sum m_i q_i / m$

#### 7.2.1 Winkelgeschwindigkeit

Winkelgeschwindigkeit  $\omega$  um Massezentrum c, geradlinige Geschw.  $v \rightarrow$

- $q(t) = \omega \times (q - c) + v(t)$
- Rotationsmatrix  $R = (x(t), y(t), z(t))$  (x,y,z Einheitsvekt. des Objektsystems im Weltsystem)

- Ableitung der Rot.Matrix:  $R'(t) = \omega^*(t)R(t) = (\omega \times x(t), \omega \times y(t), \omega \times z(t))$

### 7.2.2 Drehmoment, Drehimpuls

**Drehmoment**  $\tau_i = (q_i - c) \times F_i$  (Abstand der Kraft vom Drehpunkt mal Kraft)

**Drehimpuls:** rotierende Masse gewichtet mit Abstand  
 $L_i = m_i(q_i - c) \times (q'_i - c') = m_i(q_i - c) \times (\omega \times (q_i - c)) = (q_i - c) \times P_i$   
**(Dreh)impulserhaltung!**

**Trägheitstensor:** Impuls = Masse \* Geschw  
 Drehimpuls = Trägheitstens. \* Winkelgesch.  $L(t) = I(t)\omega(t)$

### 7.2.3 STATUS

Position, Orientierung, Impuls und Drehimpuls:

$$S(t) = (c(t), R(t), P(t), L(t)).$$

$$S'(t) = (v(t), \omega^*(t)R(t), F(t), \tau(t))$$

**numerische Integration des Status** (Euler, Runge Kutta) um Bewegung eines Körpers zu Berechnen.

### 7.2.4 Kollision

Erkennung: Raumteilungsverfahren, verschneiden der Bounding-Boxes der Objekte

- Die zur Kollisionsfläche senkrechte Geschwindigkeitskomponente ändert ihr Vorzeichen und wird mit dem Elastizitätsfaktor  $e$  multipliziert.
- $v^-$  = vor der Kollision,  $v^+$  = nach der Kollision,  $v^+ = v^-_{\text{parallel}} - v^-_{\text{senkrecht}} * e$

**Zwei Körper, Normale der Kollisionsfläche  $n$ :**

- $v^-_{\text{senkr.}} = \mathbf{n} \circ (v^-_1 + \omega^-_1 \times r_1 - v^-_2 - \omega^-_2 \times r_2)$
- $v^+_i = v^-_i + /- (Pn)/m_i$
- $L_i = I(\omega^+_i - \omega^-_i) = r_i \times (Pn)$
- $P = -2v^-_{\text{senkr.}} / (1/m_1 + 1/m_2 + n(\dots))$

### 7.3 Federnetze

Feder besitzt

- Steifigkeit  $k$  ( $kg/s^2$ )  $\rightarrow$  Kraft in Abh. von Auslenkung
- Dämpfung  $d$  ( $kg/s$ )  $\rightarrow$  Kraft in Abh. von Geschw. der Auslenkungsänderung
- Ruhelänge  $l$

$q_i$  Masse in Federnetz  $\rightarrow$

$$\mathbf{F}_i = m_i q''_i + \sum_j d_{ij} (q'_i - q'_j) + k_{ij} ((q_i - q_j) - l_{ij} \frac{q_i - q_j}{||q_i - q_j||})$$

#### 7.3.1 Multigridverfahren

Hierarchie von Gittern in verschiedenen Auflösungsstufen, um Federnetz hierarchisch auszuwerten  $\rightarrow$  Federn die ganz weit weg sind werden zusammengefasst um ihren Einfluss auf die lokale Feder auszuwerten

#### 7.3.2 Variationsprinzip

- **geometrisch:** Position des einen Körpers in abh. der Pos. des anderen Körper
- **Kräftegleichgew.:** Summe alle Kräfte = 0, (dynamisch: Summe aller Kräfte = Beschleunigung)
- **Konstituierende Relationen:** Abhängigkeit der Kraft von der Auslenkung
- dynamisch: **Geschwindigkeit-Impuls-Relation**

statischer Variationsindikator:

- geometr.:  $VI = \sum_i F_i \circ \delta x_i - \delta E_p (= 0)$
- der Kräfte:  $VI = \sum_i x_i \circ \delta F_i - \delta E_p^*$
- $E_p^*$  = Legendre-transformierte Energie in Abhängigkeit von  $F$  statt von  $x$

**Hamiltonsches Prinzip:**

- $VI = \int_{t_0..t_1} (\delta E_k^* - \delta E_p + \sum F_i \circ \delta x_i) dt$
- $E_k^*$  kinetische Kooenergie,  $E_p$  potentielle Energie

### 7.4 Holonomisch

- holonomisch = nicht Auto  $\rightarrow$  Auto kann nie nach rechts oder links fahren, aber trotzdem sich nach rechts oder links bewegen, indem es sich dreht, geradeaus fährt, sich wieder dreht.
- System aus dreh und prismat. - Gelenken sind holonomisch

### 7.5 Animationsparameter statt Auslenkungen

Die  $n$   $x_i$  sind dann Funktionen der  $m$  Animationsparameter  $\xi_j$  Das **Hamiltonsche Prinzip** ändert sich zu:

- $VI = \int \delta(E_k^* - E_p) + \sum_{j:1..m} (\sum_{i:1..n} F_i \circ \frac{\delta x_i}{\delta \xi_j}) \delta \xi_j dt$
- $E_k^* - E_p =$ : Lagrange-Fkt( $\xi', \xi, t$ ) mit
- $\delta L = \sum_{j:1..m} \frac{\delta L}{\delta \xi'_j} \frac{d}{dt} \delta \xi_j + \frac{\delta L}{\delta \xi_j} \delta \xi_j$
- $\delta L$  für  $\delta(E_k^* - E_p)$  einsetzen  $\rightarrow$  **Lagrange-Gleichungen:**
- $\frac{d}{dt} \frac{\delta L}{\delta \xi'_j} - \frac{\delta L}{\delta \xi_j} = \sum_{i:1..n} F_i \circ \frac{\delta x_i}{\delta \xi_j}$

Animation also durch

- Festlegen der Parameter  $\xi_j$
- Identifikation der Kräfte  $\sum_{i:1..n} F_i \circ \frac{\delta x_i}{\delta \xi_j}$
- Lagrange-Gleichungen aufstellen und lösen  $\rightarrow$  neue Parameter  $\xi_j$
- Parameter numerisch aufintegrieren

### 8 L-Systeme

Ersetzungsregeln:

- S  $\rightarrow$  ABA
- A  $\rightarrow$  FF (F=Zeichen Linie)
- B  $\rightarrow$  TT
- T  $\rightarrow$  -F + +F- (+/-=ändere Winkel)

Stochastisch: Zu jedem Symbol mehrere Regeln mit jew. Wahrscheinlichkeit

Parametrisch: Jedes Symbol hat Parameter (Länge, Verzweigungsgrad, etc)

Einschränkung des Wachstumsraums  $\rightarrow$  geformte Pflanzen

### 9 Strömungen

- Nabla
- Gradient
- Divergenz
- Rotation (curl)
- Helmholtz-Hodge

- **Advektion:** Transport einer **Größe  $F$**  (Temperatur, Dichte, ...) durch das Geschwindigkeitsfeld:  $dF/dt = \frac{\delta F}{\delta t} + v_i \frac{\delta F}{\delta x_i}$
- Falls keine Quellen oder Senken  $\rightarrow \delta F/\delta t = -(\mathbf{v} \circ \nabla) F$
- **Selbst-Advektion der Geschwindigkeit:** (bei konstanter Dichte, wegen Advektion des Impulses durch die Trägheit):  $\delta \mathbf{v}/\delta t = -(\mathbf{v} \circ \nabla) \mathbf{v}$
- **Diffusion:** Unterschiedliche Konzentrationen gleichen sich an: Proportional zur **Divergenz des Gradienten:**  $\delta F/\delta t = \mu * \text{div}(\nabla F) = \mu(\nabla \circ \nabla) F$
- Diffusion der Geschwindigkeit = **Viskosität**  $\nu \nabla^2 \mathbf{v}$

#### 9.1 Advektions Diffusions Gleichung

$$\delta F/\delta t = \text{Diffusion} - \text{Advektion} = \mu \nabla^2 F - (\mathbf{v} \circ \nabla) F$$

#### 9.2 Navier-Stokes Gleichungen

- $\nabla \circ \mathbf{v} = 0$  (inkompressibel  $\Rightarrow$  Divergenz verschwindet)
- $\delta \mathbf{v}/\delta t = \text{Diffusion} - \text{Advektion} - 1/\rho \nabla p + a$
- $= \nu \nabla^2 \mathbf{v} - (\mathbf{v} \circ \nabla) \mathbf{v} - 1/\rho \nabla p + a$  (Impulserhaltung)
- Beschleunigung  $a$  durch äußere Kräfte, Kraft in Richtung des fallenden Drucks.

**Helmholtz-Hodge**  $\rightarrow$  liefert divergenzfreien Teil  $\rightarrow$  also ohne Druckabhängigkeit

Diese Gleichung wird dann mehrstufig integriert:

- externe Kräfte anwenden
- advektion anwenden
- diffusion anwenden
- divergenz-freier anteil (da die bekanntlich 0 wegen inkompressibel)  $\rightarrow$  neuer Wert

### 9.3 Diskretisierung

- Skalarfeld (Druck) als Gitter gegeben
- Ableitung (berechnen aus Differenzen zwischen Gitterpunkten) ergibt Vektorfeld
- 2. Ableitung = Summe aus 2. Abl. nach  $x + 2.$  Abl. nach  $y$  kann daraus wieder bestimmt werden.

**Advektion**  $F(x, t + \Delta t) = F(x - \Delta t(v(x, t), t))$

Problem: nicht divergenzfrei! Man kann stattdessen die Geschw. auf Zellwände beziehen, statt auf Zentren, und den Ausfluss der einen Zelle als Einfluss der nächsten Zelle einbeziehen.

**Diffusion** Integration von  $\mu \nabla^2 F$  mit Eulerverfahren wäre sehr instabil.

Daher implizite Integration:  $((\mathbf{I}) - \mu \Delta t \nabla^2) F(x, t + \Delta t) = F(x, t)$

**Auftrieb** Berücksichtigung der Temperaturdifferenz von Materialien

**Vorticity Confinement** Wirbelstärke nimmt durch Diskretisierung ab!

Wieder verstärken mit  $F = \epsilon \Delta \times \left( \frac{\nabla \|\text{rot}\|}{\|\nabla \text{rot}\|} \times \text{rot} \right)$

## 10 Photon Maps

Raytracing = Strahlen vom Augpunkt verfolgen.

Photon-Map = Strahlen von den Lichtquellen verfolgen bis sie Strahlen vom Augpunkt aus treffen

Geeignet zB. zur Simulation von Lichtkegeln die bei Brechung entstehen

Photon = Energie  $E +$  Ausbreitungsrichtung  $\omega$

### 10.1 Caustic Photon Map

rückwärts-Raytracing von Lichtquelle aus bis die Photonen an einer diffusen Fläche haften bleiben

BRFD beschreibt diffuse Fläche.

Bei trifft man danach normalem Raytracing auf eine diffuse Fläche, wird die Energie dort bestimmt, indem die  $n$  nächsten Photonen in die BRFD eingesetzt werden und dann über den Radius im diese Photonen gemittelt wird. Die Dichte der Photonen beeinflusst die Helligkeit.

### 10.2 Global Photon Map

Die Photonen werden mit gewisser Wahrscheinlichkeit (abhängig von BRFD) auch auf diffusen reflektiert und verbleiben nicht auf der Fläche. Später ist die Farbe von der Anzahl verbliebener Photonen abhängig.

### 10.3 Volumen

z.B. Nebel oder Wolken: die Absorption und Streuung beeinflussen die Wahrscheinlichkeit ob das Photon haften bleibt, oder reflektiert wird. Später werden wieder die  $n$  zu einem Punkt nächsten Photonen und die Kugel in die sie reinpassen (Abstand) benutzt, um den Photon-Map auszuwerten

## 11 Mensch modellierung

- kinematik und Dynamik des Skelettes
- Daran geknüpft Modelle der Schichten Haut, Knochen, Muskeln, Fett, Haare, Kleidung

### 11.1 Implizite Flächen

Eine Skalarwertige Funktion (Dichte) der Gewebe wird berechnet, und Später wird eine Isofläche gebildet

### 11.2 Skelett

- Abfilmung vom lebendem Darsteller
- Übertragung auf eine kinematisches Skelett
- Geometrisches Sekelett wird am kinematischem Skelett befestigt

### 11.3 Muskeln

- Muskel = Ellipsoid, Längsachse, 2 Querachsen
- Bewegung = Veränderung der Länge und der Radien um Kontraktion zu simulieren

### 11.4 äußere Schichten

Die einzelnen Dichtefunktionen der Muskeln werden addiert und man erhält eine Isofläche die alle Muskeln einhüllt.

Durch das Addieren  $\rightarrow$  Gesamtfläche rutscht nach außen  $\rightarrow$  evtl. kein Modellierung von Fett.

Haut kann auch als triangulierte Fläche modelliert werden, die iterativ an das Dichtefeld angepasst wird

## 12 Feuer

- Partikel die Lebensdauer haben und ihre Farbe ständig ändern
- Kombination mit konventionellen Strömungsmodellen