Implementierung von FEM-Methoden auf programmierbaren Grafikkarten

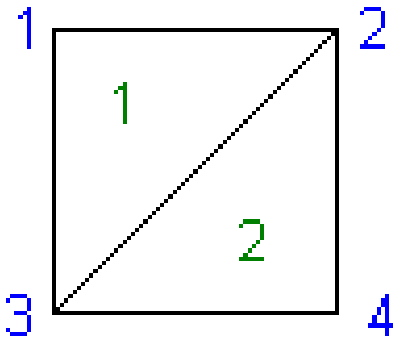
Stephan Thiess

Themenübersicht

- FEM-Methoden
- Rendering-Pipeline
- Shadersprachen
- Umsetzung der FEM-Methoden für eine GPU
- Cg Sprachbeschreibung
- Ergebnisse

Kompilation

Grundgebiet



Knotentabelle

1	1	3	2
2	2	3	4

Elementmatrix für Element 1

	u_1	u_3	u_2
u_1	a	b	c
u_3	d	e	f
u_2	g	h	i

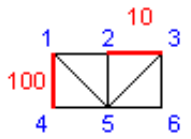
Elementmatrix für Element 2

	u_2	u_3	u_4
u_2	α	β	χ
u_3	δ	ε	ϕ
u_4	γ	η	ι

Gesamtmatrix

	u_1	u_2	u_3	u_4
u_1	a	c	b	
u_2	g	i+α	h+β	χ
u_3	d	f+δ	e+ε	ϕ
u_4		γ	η	ι

Einarbeitung von Randbedingungen (1)



Grundgebiet mit Knoten-nummern (blau) und Rändern mit Wert (rot)

Verarbeitung in Datenstrukturen

Randknoten

Rand 1	4	1
Rand 2	3	2

Randwert

Rand 1	100
Rand 2	10

	u_1	u_2	u_3	u_4	u_5	u_6	
u_1	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}	b_1
u_2	K_{21}	K_{22}	K_{23}	K_{24}	K_{25}	K_{26}	b_2
u_3	K_{31}	K_{32}	K_{33}	K_{34}	K_{35}	K_{36}	b_3
u_4	K_{41}	K_{42}	K_{43}	K_{44}	K_{45}	K_{46}	b_4
u_5	K_{51}	K_{52}	K_{53}	K_{54}	K_{55}	K_{56}	b_5
u_6	K_{61}	K_{62}	K_{63}	K_{64}	K_{65}	K_{66}	b_6

K (nach Kompilation) b

Einarbeitung des Randes 2 mit $u_2 = 10$, $u_3 = 10$

$u_2 = 10$

	u_1	u_2	u_3	u_4	u_5	u_6	
u_1	K_{11}	0	K_{13}	K_{14}	K_{15}	K_{16}	$b_1 - K_{12} * 10$
u_2	0	1	0	0	0	0	10
u_3	K_{31}	0	K_{33}	K_{34}	K_{35}	K_{36}	$b_3 - K_{32} * 10$
u_4	K_{41}	0	K_{43}	K_{44}	K_{45}	K_{46}	$b_4 - K_{42} * 10$
u_5	K_{51}	0	K_{53}	K_{54}	K_{55}	K_{56}	$b_5 - K_{52} * 10$
u_6	K_{61}	0	K_{63}	K_{64}	K_{65}	K_{66}	$b_6 - K_{62} * 10$

$u_3 = 10$

	u_1	u_2	u_3	u_4	u_5	u_6	
u_1	K_{11}	0	0	K_{14}	K_{15}	K_{16}	$b_1 - K_{13} * 10$
u_2	0	1	0	0	0	0	$10 - 0 * 10$
u_3	0	0	1	0	0	0	10
u_4	K_{41}	0	0	K_{44}	K_{45}	K_{46}	$b_4 - K_{43} * 10$
u_5	K_{51}	0	0	K_{54}	K_{55}	K_{56}	$b_5 - K_{53} * 10$
u_6	K_{61}	0	0	K_{64}	K_{65}	K_{66}	$b_6 - K_{63} * 10$

	u_1	u_2	u_3	u_4	u_5	u_6	
1	1	0	0	0	0	0	100
2	0	1	0	0	0	0	10
3	0	0	1	0	0	0	10
4	0	0	0	1	0	0	100
5	0	0	0	0	K_{55}	K_{56}	b_5
6	0	0	0	0	K_{65}	K_{66}	b_6

nach Einarbeitung des Randes 1

Einarbeitung von Randbedingungen (2)

$$b(k) = b(k) - K(k, u_i) * c$$

$$b(u_i) = c$$

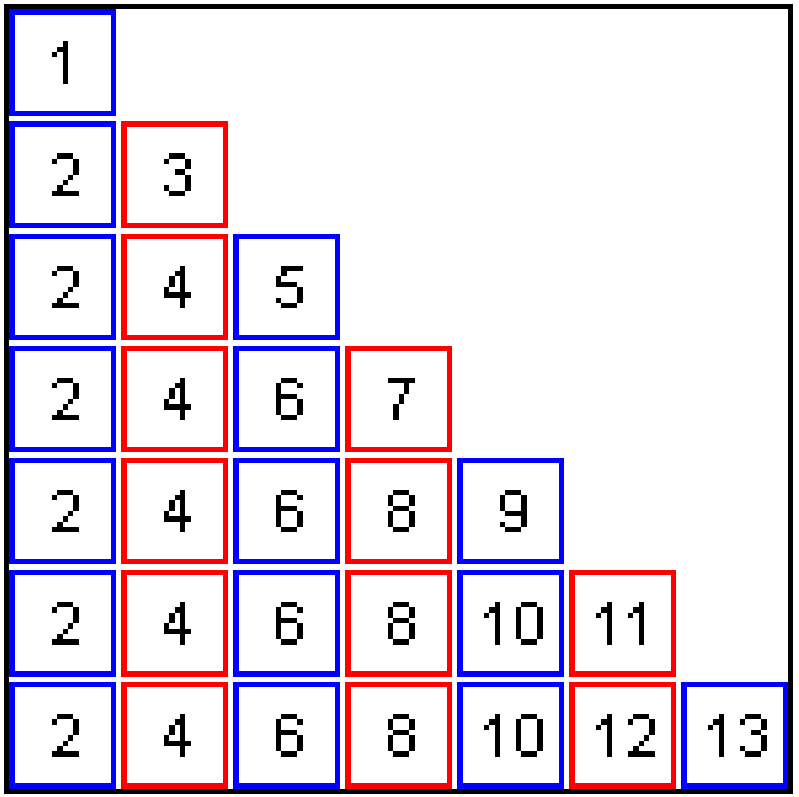
$$K(u_i, k) = 0$$

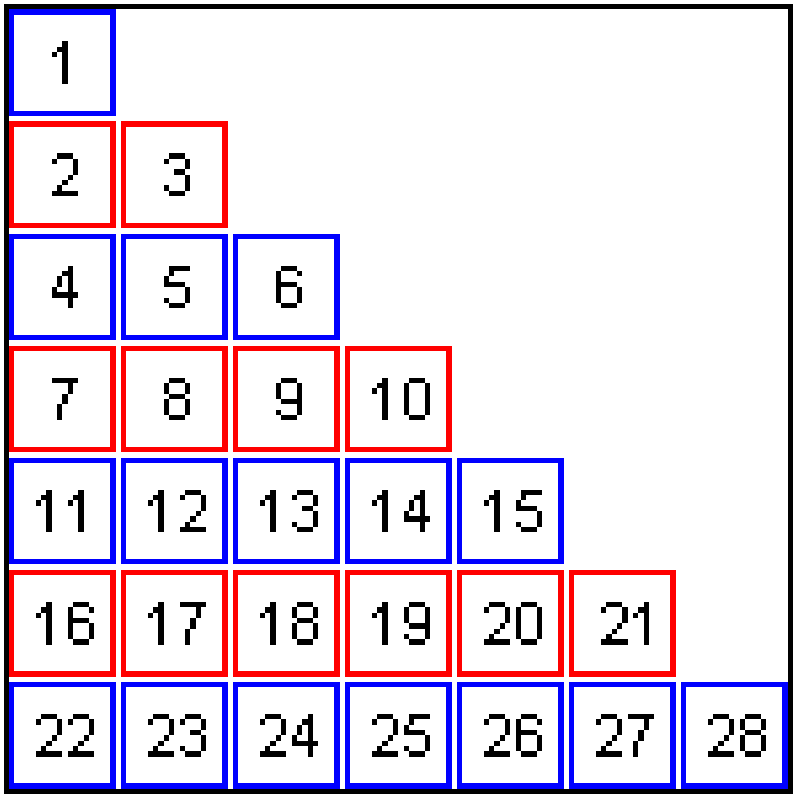
$$K(k, u_i) = 0$$

$$K(u_i, u_i) = 1$$

K = Gesamtmatrix, b = Ergebnisvektor, c = Randwert, u_i = Knotenindex

Cholesky-Verfahren (1)





Cholesky-Verfahren (2)

Diagonalelemente

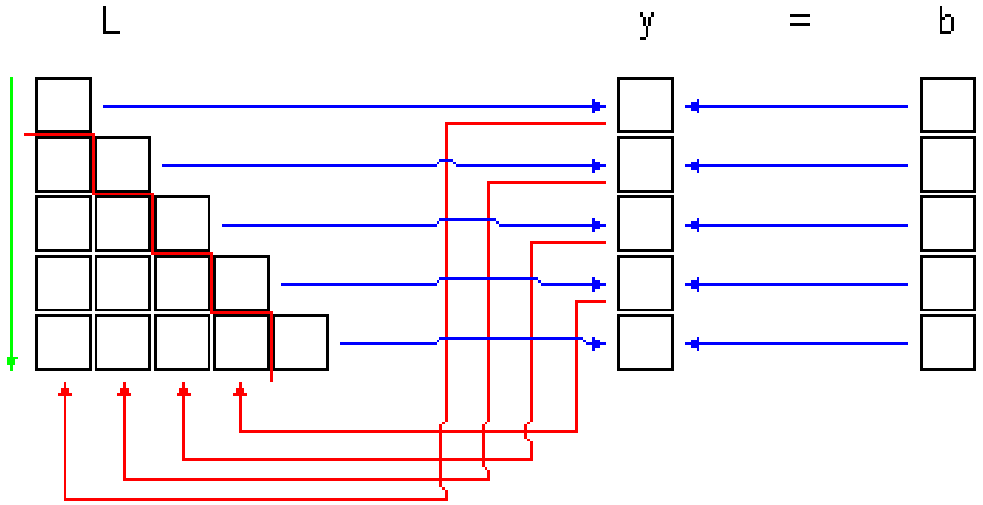
$$l_{1,1} = \sqrt{a_{1,1}}$$

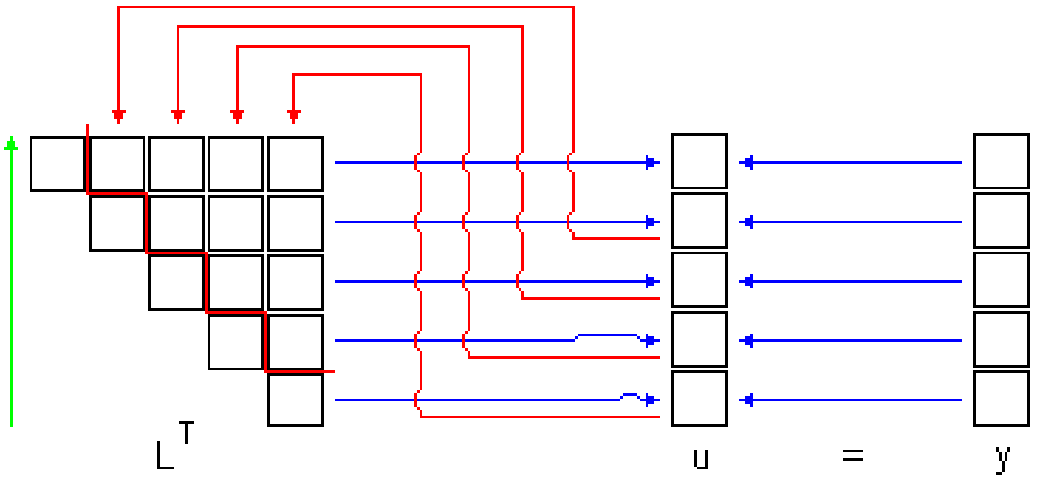
$$l_{k,k} = \sqrt{a_{k,k} - \sum_{\mu=1}^{k-1} l_{k,\mu}^2}$$

restlichen Elemente

$$l_{i,k} = \frac{a_{i,k} - \sum_{\mu=1}^{k-1} l_{i,\mu} \cdot l_{k,\mu}}{l_{k,k}}$$

Vorwärts-Rückwärts-Einsetzen ($K*u = L*L^T*u = b$)





Jacobi-Verfahren
Gauss-Seidel-Verfahren

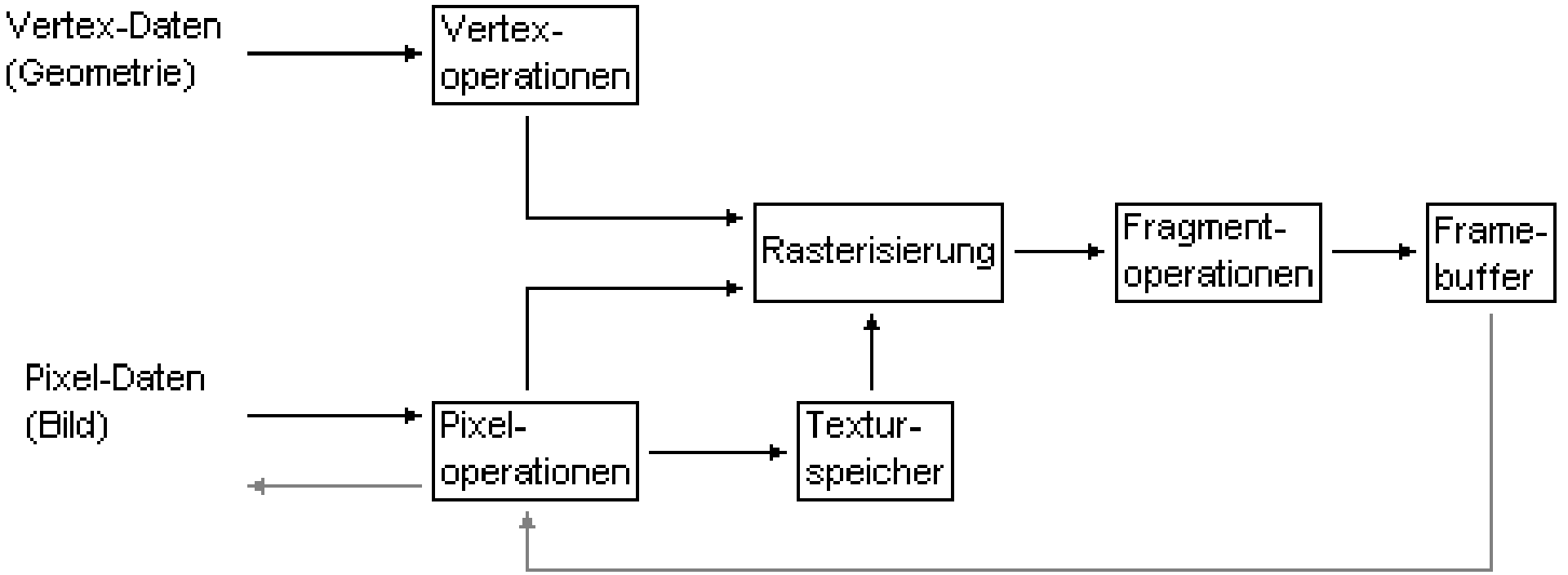
Jacobi-Verfahren

$$x_j^{(q)} = \frac{1}{a_{j,j}} \cdot \left(b_j - \sum_{k=0}^{j-1} a_{j,k} \cdot x_k^{(q-1)} - \sum_{k=j+1}^{n-1} a_{j,k} \cdot x_k^{(q-1)} \right)$$

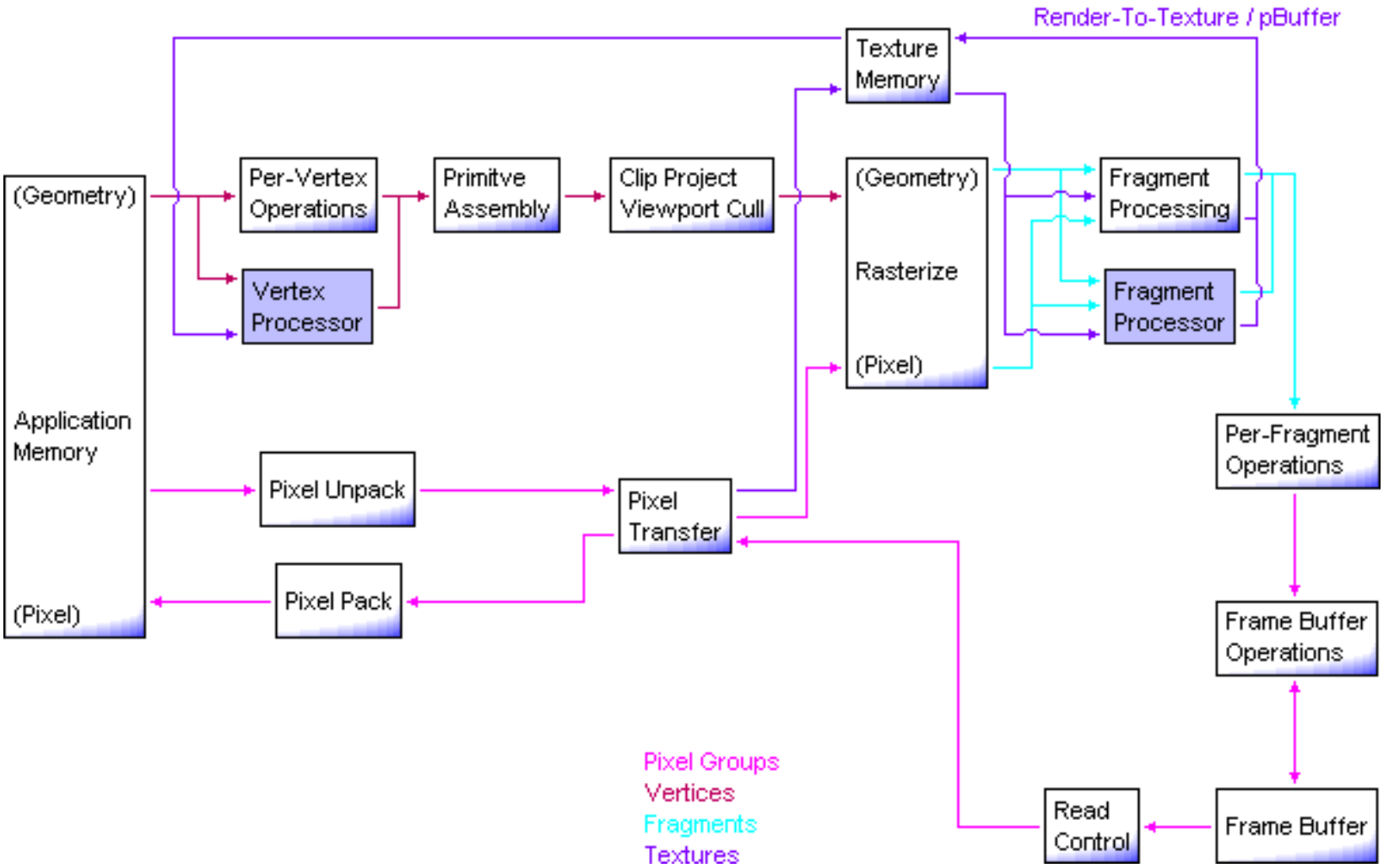
Gauss-Seidel-Verfahren

$$w_i^{\nu+1} = \frac{1}{a_{i,i}} \cdot \left(r_i - \sum_{j=1}^{i-1} a_{i,j} \cdot w_j^{\nu+1} - \sum_{j=i+1}^{M} a_{j,j} \cdot w_j^{\nu} \right)$$

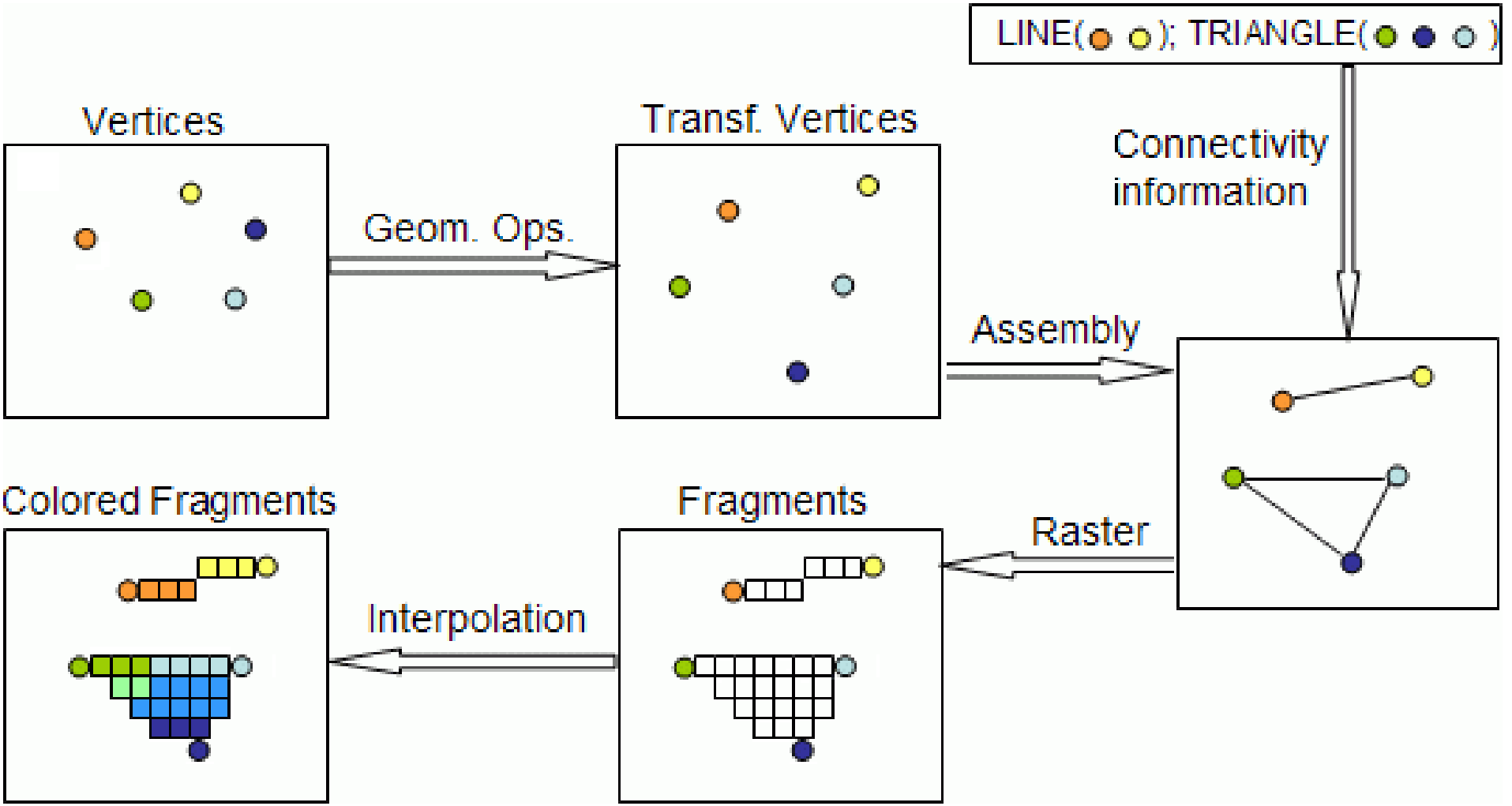
OpenGL Rendering-Pipeline (1)



OpenGL Rendering-Pipeline (2)



Rendering-Pipeline



Shadersprachen

- Cg (C for Graphics)
- GLSL (OpenGL Shading Language)
- HLSL (High Level Shading Language)
- Sh

Analogie zwischen GPU und CPU

- Textur = Array
- Fragmentprogramm = Code innerhalb von Schleifen
- Texturupdate = Feedback
- Vertexkoordinaten = Bereich für die Berechnung

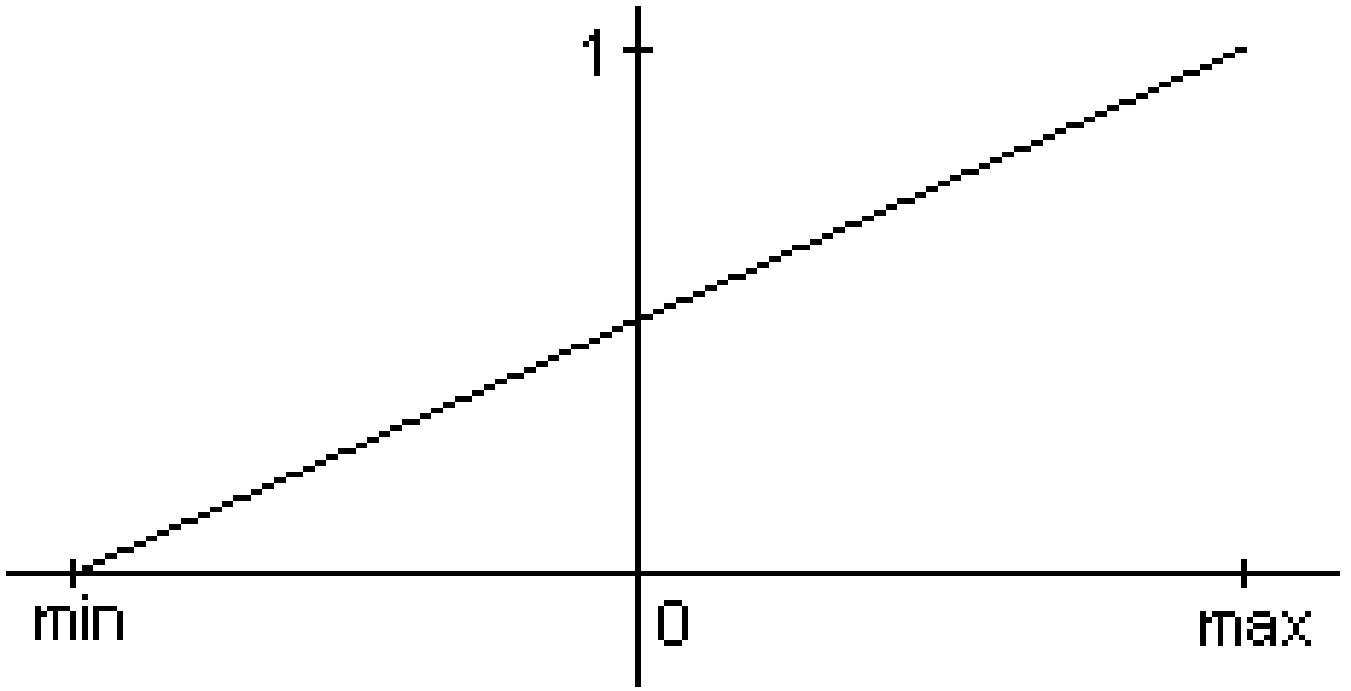
Kodierung auf den Bereich [0; 1] (1)

Darstellung der Floating Point Zahlen

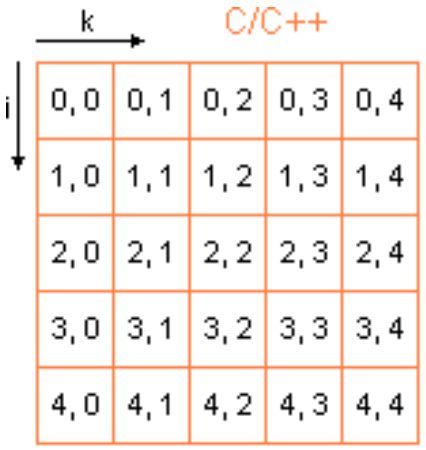
Name	Vorzeichen	Exponent	Mantisse	größter Wert	kleinster Wert	vollständiger Zahlenbereich
nVIDIA 16 Bit	15 (1)	14:10 (5)	9:0 (10)	± 65.504	$\pm 2^{-14} \approx 10^{-5}$	± 2048
ATi 16 Bit	15 (1)	14:10 (5)	9:0 (10)	± 131.008	$\pm 2^{-15} \approx 10^{-5}$	± 2048
ATi 24 Bit	23 (1)	22:16 (7)	15:0 (16)	$\pm \sim 2^{64} \approx 10^{19}$	$\pm 2^{-62} \approx 10^{-19}$	± 131.072
nVIDIA 32 Bit	31 (1)	30:23 (8)	22:0 (23)	$\pm \sim 2^{128} \approx 10^{38}$	$\pm 2^{-126} \approx 10^{-38}$	± 16.777.216

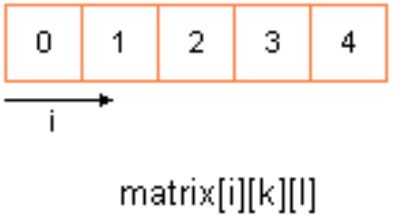
Kodierung auf den Bereich [0; 1] (2)

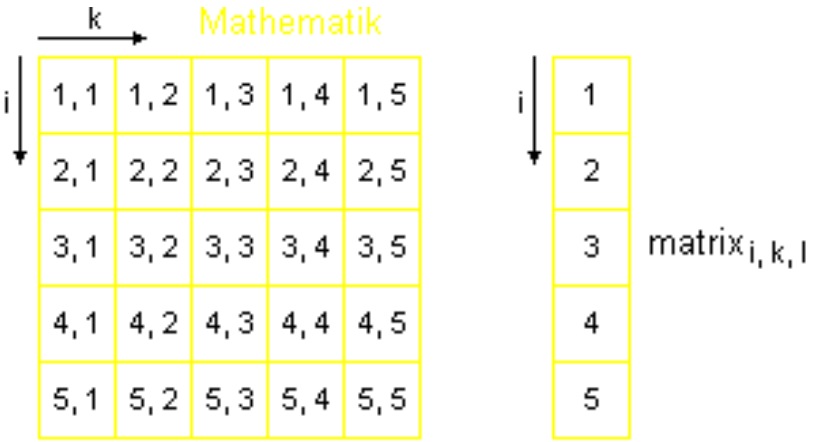
Kodierungsfunktion

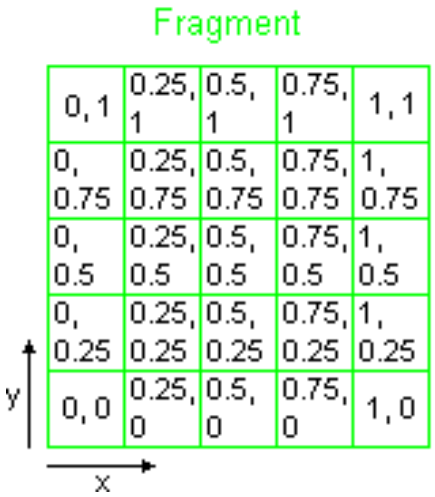


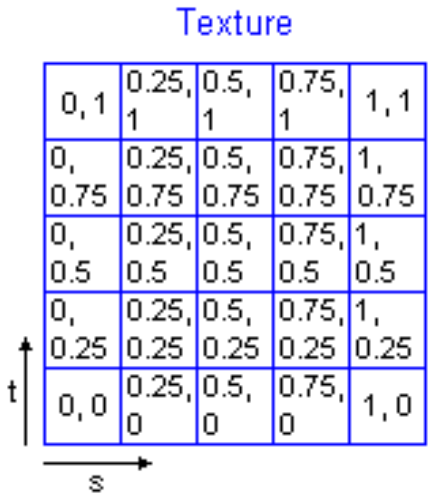
Matrizen und Vektoren der unterschiedlichen Systeme

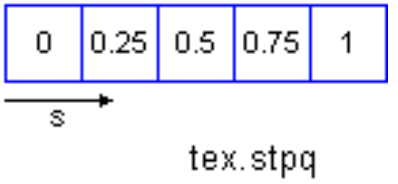




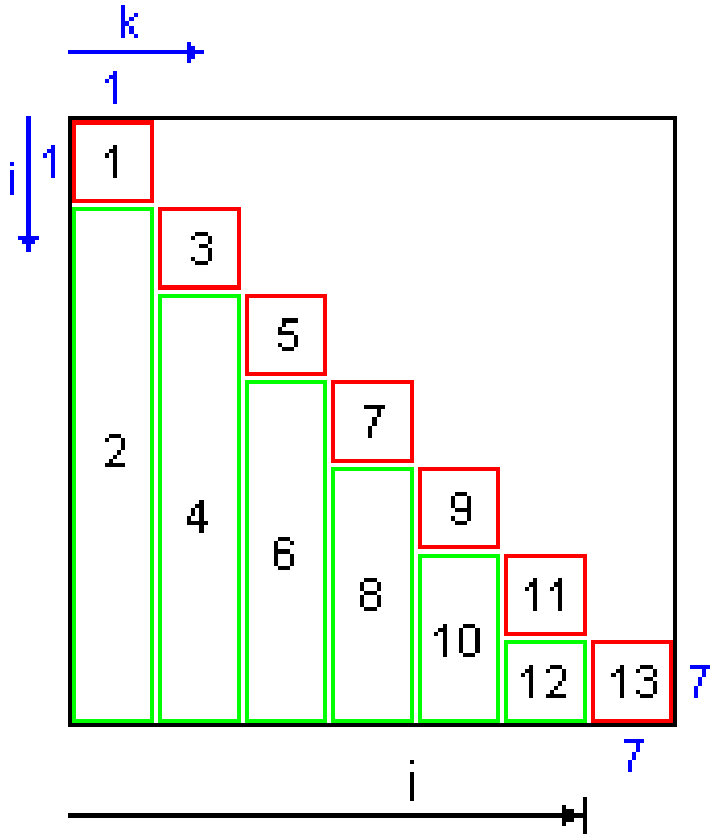


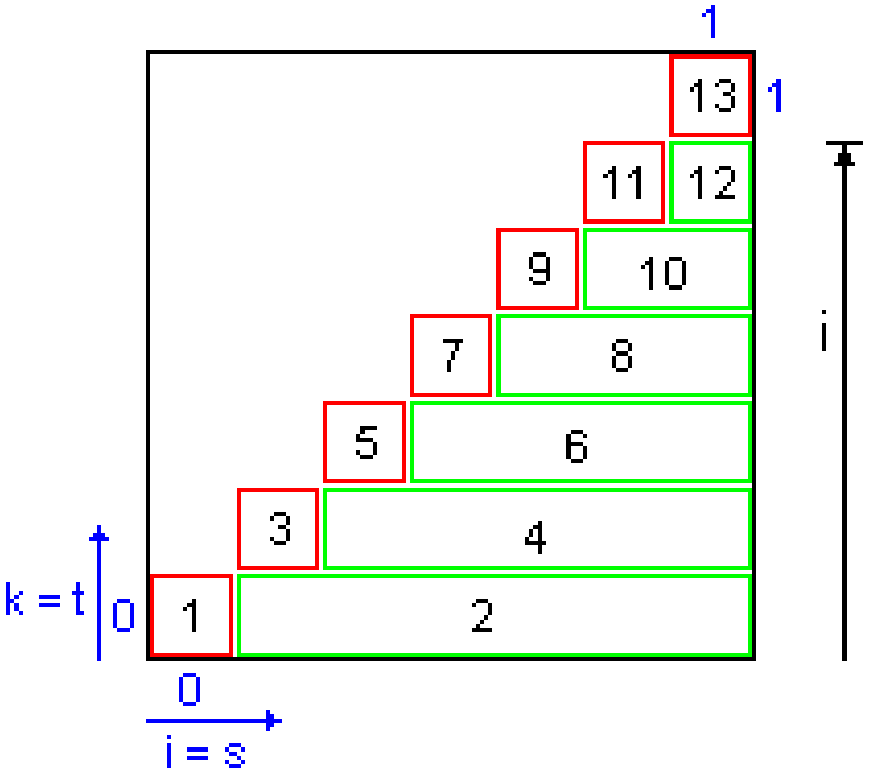




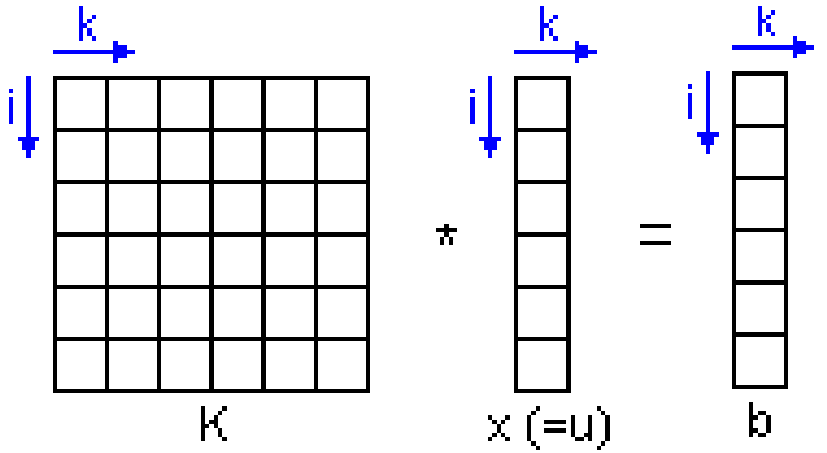


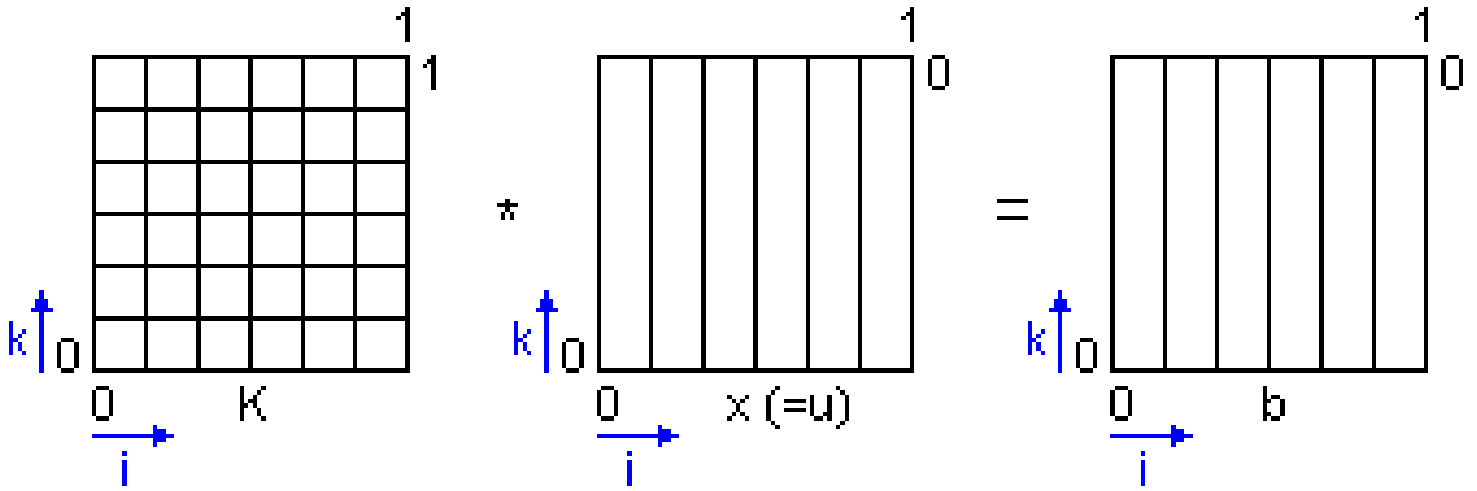
Cholesky-Verfahren



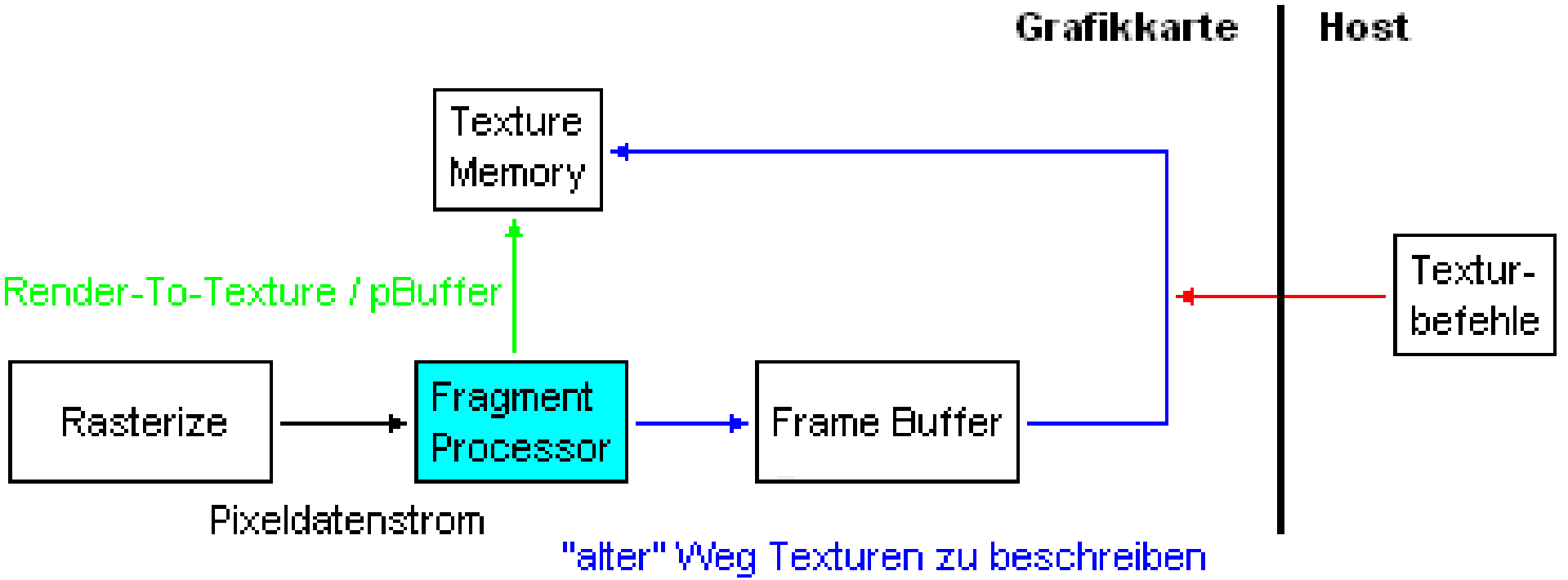


Jacobi-Verfahren

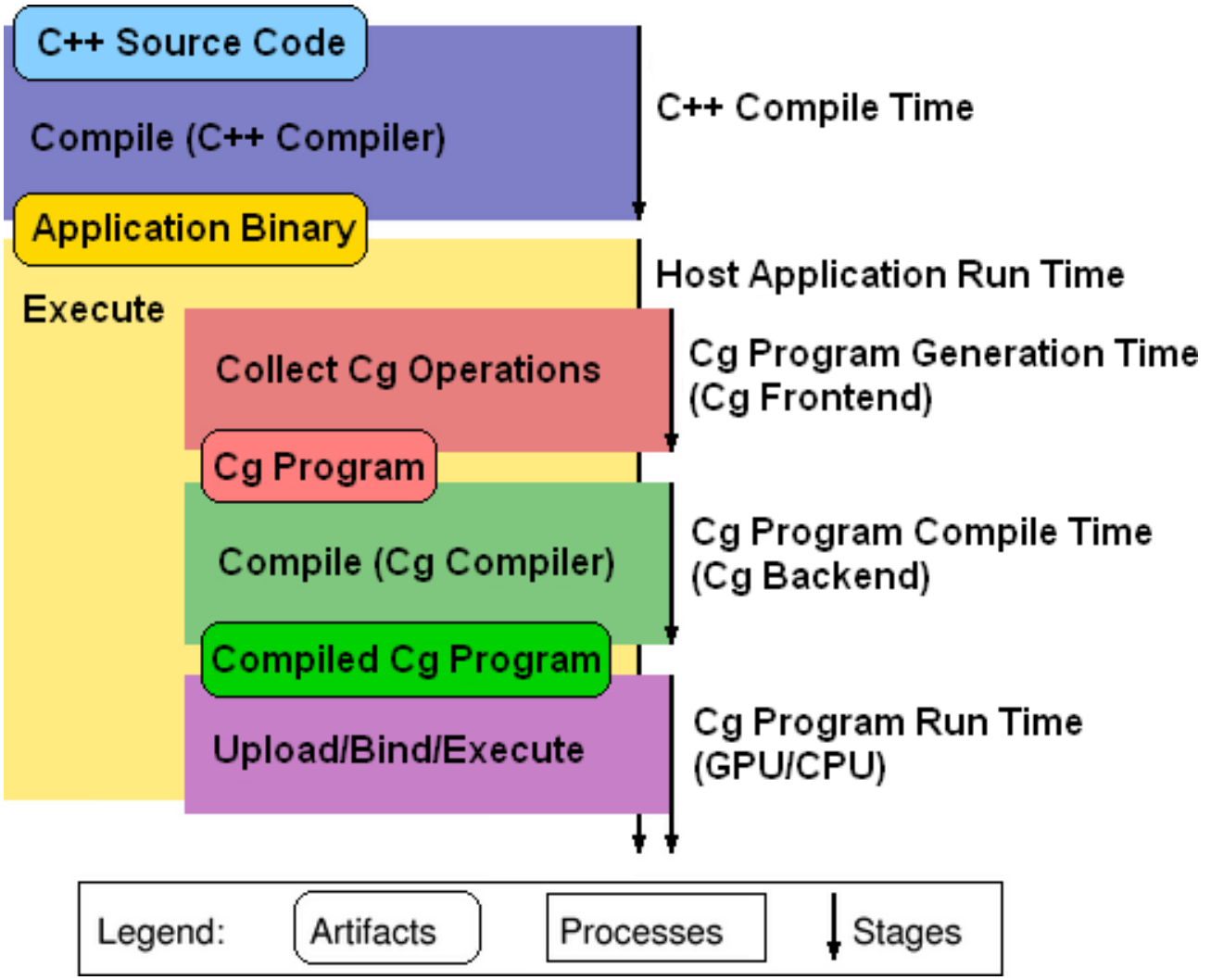




Render-to-Texture



Ablauf von Cg



Grundlegender Programmaufbau

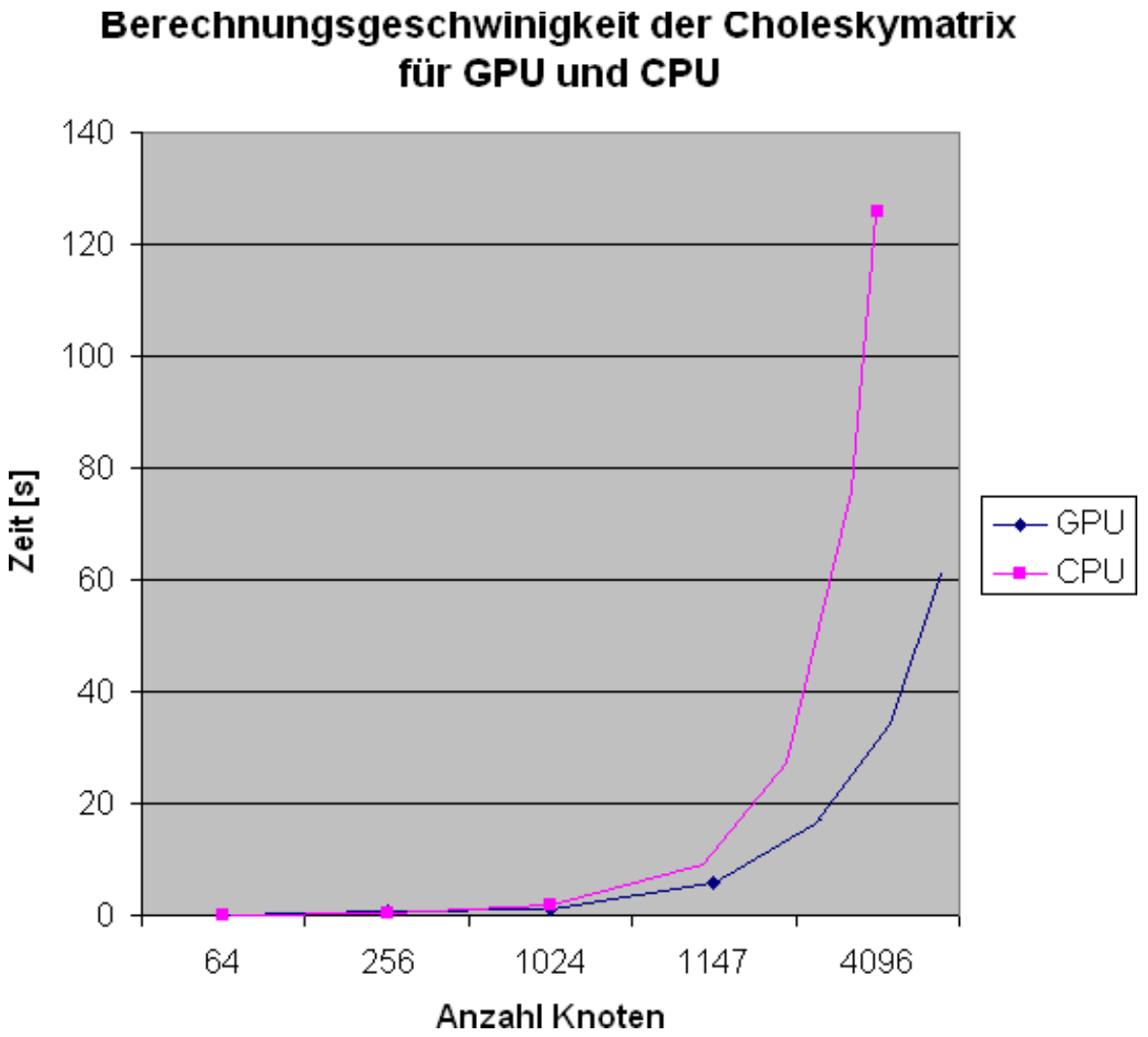
- GLUT-Funktionen initialisieren
- Views initialisieren (glOrtho, glViewport usw.)
- Cg initialisieren
  - Fehlerroutine festlegen
  - Vertex- und Fragmentprofile laden
  - Vertex-Programm erstellen, kompilieren und laden
  - Fragment-Programm erstellen, kompilieren und laden
  - Shaderparameter extrahieren und mit Werten belegen
- Texturen laden
- Zeichnen (Display-Funktion)
  - Vertex-Programm einbinden
  - Fragment-Programm einbinden
  - Texturen einbinden
  - Vertex-Strukturen (GL_POINTS usw.) zeichnen

Zeitmessungen für das Cholesky-Verfahren (1)

GPU (nVIDIA GeForce 6800 GT)	
Anzahl Knoten	Zeit [s]
256	0,5 – 0,8
1024	1
1147	< 6

CPU (Intel Xeon 2,7 GHz)	
16	0
64	0
256	0
1024	2
4096	126

Zeitmessungen für das Cholesky-Verfahren (2)



Rechenungenauigkeit zwischen Float und Double

	Abweichung	
Anzahl Knoten	absolut	relativ [%]
16	0,000002195	0,000003658
64	0,000013751	0,000017382
256	0,000061564	0,000090462
1024	0,000193816	0,000323066
4096	0,001005299	0,001289099
16384	0,131485308	0,219139712
65536	2,000139280	3,333557305
262144	13,931472853	23,219044629

Rechenungenauigkeit und Einfluss der Schlüssel

	CPU			
	Float		Double	
Anzahl Knoten	Diagonalwert	Wert darunter	Diagonalwert	Wert darunter
256	-0,500030458	2,000000000	-0,4999549995	2,0000000000
1024	-0,500030458	2,000000000	-0,4999549995	2,0000000000

	GPU (nur Float)					
	kodiert		dekodiert		Schlüssel	
	Diagonalwert	Wert darunter	Diagonalwert	Wert darunter	unterer	oberer
256	0,100009047	0,372549027	-0,499954765	0,862745135	-1	4
1024	0,099993907	0,482352942	-0,500030465	1,411764710		
256	0,187505662	0,356862753	-4,999547000	0,854902028	-2	6
1024	0,187496185	0,427450984	-0,500030517	1,419607878		
256	0,300003022	0,392156869	-0,499546700	0,882353035	-5	10
1024	0,299997955	0,427450984	-0,500030675	1,411764760		
256	0,487501144	0,521568656	-0,499954224	0,862746240	-20	20
1024	0,487499237	0,533333361	-0,500030516	1,333334444		
256	0,663333654	0,670588255	-0,499951900	0,588238250	-100	50
1024	0,663333118	0,674509823	-5,000323000	1,176473450		
256	0,570450783	0,572549045	-0,499955480	0,235695177	-200,5	150,1
1024	0,570450544	0,576470614	-0,500039274	1,610597268		

Bilder der Cholesky-Rechnung (1)

Bilder der Cholesky-Rechnung (2)