BUNDESPATENTGERICHT

IM NAMEN DES VOLKES

URTEIL

An Verkündungs Statt
Zugestellt am
06.04.2021
…

7 Ni 28/19 (EP)
verbunden mit
7 Ni 35/19 (EP)

(Aktenzeichen)

In der Patentnichtigkeitssache

…

ECLI:DE:BPatG:2021:060421U7Ni28.19EP.0

betreffend das europäische Patent EP 1 177 531
(DE 600 07 521)

hat der 7. Senat (Juristischer Beschwerdesenat und Nichtigkeitssenat) des Bundespatentgerichts aufgrund der mündlichen Verhandlung vom 19. November 2020 durch die Richterin Püschel als Vorsitzende sowie den Richter Dipl.-Ing. Baumgardt, die Richterin Dr. Schnurr und die Richter Dipl.-Phys. Univ. Dr. Forkel und Dipl.-Phys. Univ. Dr. Städele

für Recht erkannt:

I. Das europäische Patent 1 177 531 wird mit Wirkung für das Hoheitsgebiet der Bundesrepublik Deutschland in vollem Umfang für nichtig erklärt.

II. Von den Kosten des Rechtsstreits trägt die Beklagte 3/4 der Gerichtskosten sowie die außergerichtlichen Kosten der Klägerinnen zu 1, 2 und 5. Die Klägerin zu 3 trägt 1/4 der Gerichtskosten, im Übrigen tragen die Parteien ihre Kosten selbst.

III. Das Urteil ist hinsichtlich der Kosten gegen Sicherheitsleistung in Höhe von 120% des jeweils zu vollstreckenden Betrages vorläufig vollstreckbar.

Tatbestand

Die Beklagte ist eingetragene Inhaberin des in englischer Verfahrenssprache mit Wirkung auch für den Hoheitsbereich der Bundesrepublik Deutschland erteilten europäischen Patents 1 177 531 (Streitpatent) mit der Bezeichnung „Method and system for providing programmable texture processing“ (Verfahren und Vorrichtung

zur programmierbaren Texturverarbeitung). Im Deutschen Patent- und Markenamt wird das inzwischen erloschene Streitpatent, das aus der internationalen Anmeldung PCT/US00/11907 vom 2. Mai 2000 (veröffentlicht als Druckschrift WO 00/68891 A1 = Anlage **NK2**) hervorgegangen ist und die Priorität der US-Patentanmeldung 09/306,877 vom 7. Mai 1999 in Anspruch nimmt, unter dem Aktenzeichen 600 07 521.4 geführt.

Die Klägerinnen zu 1, 2 und 5 greifen das Streitpatent übereinstimmend in vollem Umfang an. Der Senat hat die Klage 7 Ni 28/19 (EP) der Klägerin zu 1 und die Klage 7 Ni 35/19 (EP) der Klägerin zu 2 zur gemeinsamen Verhandlung und Entscheidung verbunden. Die Klägerin zu 3, die den verbundenen Verfahren am 1. August 2019 beigetreten ist, hat ihre Klage am 31. Januar 2020 zurückgenommen; einen Kostenantrag hat die Beklagte insoweit nicht gestellt. Weiter hinzuverbunden war zunächst auch die Klage 7 Ni 81/19 (EP) der Klägerin zu 4. Diese Klage ist aber nach der Klagerücknahme dieser Klägerin wieder abgetrennt worden. Die Klägerin zu 5 ist dem Verfahren 7 Ni 35/19 (EP) am 14. September 2020 als weitere Klägerin beigetreten.

Das Streitpatent umfasst 13 Patentansprüche, die alle mit den vorliegenden Klagen angegriffen werden. Patentanspruch 1 mit darauf unmittelbar oder mittelbar rückbezogenen Unteransprüchen 2 bis 7 betrifft ein System zur Verarbeitung von Strukturen für eine Grafikabbildung auf einer Anzeige, Patentanspruch 8 mit darauf unmittelbar oder mittelbar rückbezogenen Unteransprüchen 9 bis 13 ein Verfahren zur Strukturverarbeitung für eine Grafikabbildung auf einer Anzeige.

Die nebengeordneten Patentansprüche 1 und 8 haben in ihrer erteilten Fassung folgenden Wortlaut:

1. A system (150) for processing textures for a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments, the system comprising:

a plurality of texture processors (154; 190; 300) for processing a portion of the plurality of fragments; characterised by:

a memory (156; 160) for storing a portion of a program for processing a plurality of texture portions for the plurality of fragments, the portion of the program including a plurality of instructions; said plurality of texture processors (154; 190; 300) coupled with the memory (156; 160), each of the plurality of texture processors (154; 190; 300) for processing a portion of the plurality of texture portions for a fragment of the plurality of fragments in accordance with the program, the plurality of texture processors (154; 190; 300) capable of processing a second portion of the plurality of texture portions in parallel, the plurality of texture processors (154; 190; 300) implementing a portion of the plurality of instructions.

8. A method for processing textures of a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments, the system comprising a plurality of texture processors (154; 190; 300) for processing a portion of the plurality of fragments; the method characterised by the steps of:

(a) providing a plurality of texture portions for the plurality of fragments to a plurality of texture processors (154; 190; 300), and

(b) processing the plurality of texture portions in the plurality of texture processors (154; 190; 300) in parallel based on at least one program,

the at least one program including a plurality of instructions, the plurality of

texture processors (154; 190; 300) implementing a portion of the plurality of instructions, the at least one program thereby controlling processing of the plurality of texture portions by the plurality of texture processors (154; 190; 300).

Die deutsche Übersetzung lautet gemäß der Streitpatentschrift EP 1 177 531 B1:

1. Ein System (150) zur Verarbeitung von Strukturen für eine Grafikabbildung auf einer Anzeige (1; 104), wobei die Grafikabbildung ein Objekt umfaßt, das Objekt eine Mehrzahl von Fragmenten umfaßt und das System beinhaltet:

eine Mehrzahl von Strukturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der Mehrzahl der Fragmente; gekennzeichnet durch:

einen Speicher (156; 160) zum Speichern eines Teils eines Programms zur Verarbeitung einer Mehrzahl von Strukturabschnitten für die Mehrzahl der Fragmente, wobei der Abschnitt des Programms eine Mehrzahl von Anweisungen umfaßt; wobei die Mehrzahl der Strukturprozessoren (154; 190; 300) mit dem Speicher (156; 160) gekoppelt ist und jeder der Mehrzahl der Strukturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der Mehrzahl der Strukturabschnitte für ein Fragment der Mehrzahl der Fragmente gemäß dem Programm eingerichtet ist, wobei die Mehrzahl der Strukturprozessoren (154; 190; 300) in der Lage ist, einen zweiten Abschnitt der Mehrzahl der Strukturabschnitte parallel zu verarbeiten und die Mehrzahl der Strukturprozessoren (154; 190; 300) ein Abschnitt der Mehrzahl der Anweisungen implementiert.

8. Ein Verfahren zur Strukturverarbeitung für eine Grafikabbildung auf einer Anzeige (1; 104), wobei das Grafikabbild ein Objekt beinhaltet, das Objekt eine Mehrzahl von Fragmenten beinhaltet und das System eine Mehrzahl von Strukturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der

Mehrzahl der Fragmente umfaßt und das Verfahren durch die Mehrzahl der Schritte gekennzeichnet ist;

(a) Liefern einer Mehrzahl von Strukturabschnitten für die Mehrzahl von Fragmenten an eine Mehrzahl von Strukturprozessoren (154; 190; 300), und

(b) paralleles Verarbeiten der Mehrzahl von Strukturabschnitten in der Mehrzahl von Strukturprozessoren (154; 190; 300) auf der Grundlage von wenigstens einem Programm, wobei

das wenigstens eine Programm eine Mehrzahl von Anweisungen aufweist, die Mehrzahl von Strukturprozessoren (154; 190; 300) einen Abschnitt der Mehrzahl der Anweisungen implementiert und wenigstens ein Programm auf diese Weise das Verarbeiten der Mehrzahl der Strukturabschnitte mittels der Mehrzahl der Strukturprozessoren (154; 190; 300) steuert.

Wegen des Wortlauts der Unteransprüche 2 bis 7 und 9 und 13 in englischer bzw. deutscher Sprache wird auf die Streitpatentschrift EP 1 177 531 B1 Bezug genommen.

Die Klägerinnen zu 1, 2 und 5 machen übereinstimmend gegenüber der erteilten Fassung des Streitpatents den Nichtigkeitsgrund der mangelnden Patentfähigkeit, die Klägerinnen zu 2 und 5 zusätzlich den Nichtigkeitsgrund der unzulässigen Erweiterung geltend (Art. II § 6 Abs. 1 Nr. 1 und 3 IntPatÜG i. V. m. Art. 138 Abs. 1 Buchst. a) und c) EPÜ), wobei die Klägerinnen die fehlende Patentfähigkeit in erster Linie auf fehlende Neuheit stützen.

Sie berufen sich auf folgende Unterlagen (wobei im weiteren Text die fett gedruckten Anlagenbezeichnungen verwendet werden; Anlagenbezeichnungen **NK1a** und **NK6** bis **NK11** und **NK48** bis **NK65** aus 7 Ni 28/19 (EP), restliche Anlagenbezeichnungen aus 7 Ni 35/19 (EP)).

NK1	EP 1 177 531 B1 (Streitpatent)
NK1a	DE 600 07 521 T2 (deutsche Übersetzung)
NK2	WO 00/68891 A1 (ursprüngliche Anmeldung)
NK3	US 09/306,877 (Prioritätsdokument)
NK6	US 5,230,039 A
NK7	US 5,586,234 A
NK8 = NK28	US 5,808,690 A
NK9	EP 0 821 339 A1
NK10	US 5,821,944 A
NK11	James D. Foley et al., The Systems Programming Series - Computer Graphics, Principles and Practice, Second Edition, S. 855, 887-890;
NK12	John Eyles et al., PixelFlow: The Realization, Siggraph/Eurographics Workshop on Graphics Hardware, 1997, S. 57-68;
NK13	Anselmo Lastra et al., Real-Time Programmable Shading, Symposium on Interactive 3D Graphics, 1995, S. 59–66, 207;
NK14	US 5,481,669 A
NK15	Auszug aus dem Web-Archiv „Wayback Machine": „http://web.archive.org/web/19980613135532/http://www.voodoo2.com:80/3dblaster.html“ betr. die Webseite „Creative Labs 3D Blaster Voodoo2 Review“, März 1998, Ausdruck vom 16. Juli 2018, S. 1-13;
NK16	Auszug aus dem Web-Archiv „Wayback Machine": „http://web.archive.org/web19980613135630/http://www.voodoo2.com:80/m3d2.htm“ betr. die Webseite „Diamond Multimedia’s Monster 3D II“, März 1998, Ausdruck vom 16. Juli 2018;
NK17	Zeitschrift PC-Player, Ausgabe 4/98, S. 1, 5-7, 192, 193;
NK18	Zeitschrift Maximum PC, Ausgabe Oktober 1998 (13 Seiten Auszug);

NK19 Internet-Veröffentlichung („https://www.win.tue.nl/~aeb/ftpdocs/linux-local/docs/HOWTO/3Dfx-HOWTO“) Bernd Kreimeier, The Linux 3Dfx HOWTO, 6. Februar 1998;

NK20 Glide Programming Guide, Programming the 3Dfx Interactive Glide Rasterization Library 2.4, Document Release 018, 3Dfx Interactive, Inc., San Jose, Kalifornien (USA), 25. Juli 1997, Druckdatum 30. Juli 1997;

NK20a Auszug aus dem Web-Archiv „Wayback Machine“ vom 22. April 1998 betreffend die Webseite „http://www.3dfx.com:80/software/download_glide.html“;

NK21 Zeitschrift Next Generation, Ausgabe Januar 1998, Artikel „3Dfx Vodoo^{2}“;

NK22 Produktübersicht „N… RIVA TNT“ mit Copyright-Vermerk 1998;

NK22a Zeitschrift Microprocessor Report, Ausgabe vom 3. August 1998;

NK23 Handbuch „ELSA ERAZOR II^{TM} ELSA VICTORY Erazor LT^{TM}“ mit Datumsangabe September 1998;

NK23a Handbuch „ASUS V3400TNT Series, AGP Graphics Cards” mit Copyright-Vermerk Januar 1999;

NK24 Pressemitteilung von Microsoft, „Microsoft Ships Final Release of DirectX 6.0”, Redmond, Washington (USA), 7. August 1998.

NK25 Mark Segal et al., The OpenGL®Graphics System: A Specification, Version 1.2.1, 1. April 1999, S. 238;

NK26 Internet-Veröffentlichung (https://www.tomshardware.com/...) Thomas Pabst, New 3D Chips - Banshee, G200, RIVA TNT and Savage3D, 18. August 1998;

NK27 Zeitschrift Maximum PC, Ausgabe Januar 1999, Auszug mit Artikel „ERAZOR II“;

NK28 = **NK8** US 5,808,690 A

NK29 Macmillan Dictionary of Microcomputing, 3. Aufl. 1985, S. 150, 223, 224, 230, 231, 232, 307, 308, 310;

NK30 Jerry C. Whitaker/Joy S. Shetler (California Polytechnic State University), (Hrsg.), The Electronics Handbook, 1996, S. 1859, 1863, 1864;

NK31 Steven Molnar, The Pixel Flow Texture and Image Subsystem, veröffentlicht in “Proceedings of the Tenth Eurographics Workshop on Graphics Hardware (EGGH’95)”, 1995, S. 3-13 (Internetveröffentlichung unter “http://dx.doi.org/10.2312/EGGH/ EGGH95/003-013”);

NK32 Benutzeranleitung „MONSTER 3D II“ mit Vermerk „Copyright 1995, 1996, 1997, 1998“;

NK33 Zeitschrift MAXIMUM PC, Ausgabe November 1998 (Auszug).

NK34 Wikipedia-Artikel zum Stichwort „3dfx Voodoo Banshee“, Bearbeitungsstand 13. Mai 2017;

NK35 Ordner-Index, im Internet veröffentlicht unter „https://www.win.tue.nl/~aeb/ftpdocs/linux-local/docs/HOWTO“, Ausdruck vom 13. Dezember 2018;

NK37 Auszug aus dem Web-Archiv „Wayback Machine": „http://web.archive.org/web/19990508173428/http://www.review-zone.com:80/hardware/vi...“ betr. die Webseite „REVIEW ZONE“ vom 20. Oktober 1998;

NK38 Internet-Veröffentlichung „Image Quality Analysis Fall 2003: A Glance Through the Looking Glass” von Derek Wilson („https://www.anandtech.com/print/1205/“) vom 10. Dezember 2003;

NK39 Jahresbericht (10-Kform) der N… Corporation (Kalifornien/USA) an die Securities and Exchange Commission SEC betreffend das am 1. Januar 1999 endende Fiskaljahr, veröffentlicht im Internet unter

„https://www.sec.gov/Archives/edgar/data/1045810/000092962 4-99-000772.txt“;

NK40 Auszug aus der N…-Finanzdatenbank betreffend die Firmenquartale 2/1999 bis 1/2000;

NK41 Klageschrift der S… Incorporated gegen die N… Corporation vom 9. November 1998;

NK43 Rechnung und Lieferschein vom 24. Februar 1999 der E… Co., Ltd, Taiwan, an N… Corporation betr. RIVA-TNT;

NK44 Wikipedia-Artikel zum Stichwort „RIVA TNT“, Bearbeitungs-stand 20. September 2018;

NK45 Zeitschrift PC Magazine, Ausgabe vom 4. Mai 1999 (Auszug);

NK48 Wikipedia-Artikel zum Stichwort „Arithmetisch-logische Einheit“, Bearbeitungsstand 28. August 2018;

NK49 Programmieranleitung der 3D-Verarbeitung von DirectX 6, veröffentlicht als Teil des DirectX 6 SDK am 12. Dezember 1998;

NK50 Auszug aus dem Web-Archiv „Wayback Machine“ betr. die Webseite „http://www.microsoft.com/DirektX/default.asp“, Stand 16. Januar 1999;

NK51 Ausdruck der Webseite „http://www.gamasutra.com/view/feature/131698/multitexturing _in_dir...“ unter der Überschrift „Gamasutra - Multitexturing in DirectX 6“ vom 9. Oktober 1998;

NK52 Auszug aus dem Web-Archiv „Wayback Machine“ betreffend die Webseite „http://www.opengl.org/Products/Accelerators.html“ vom 9. Februar 1999;

NK53 Pressemitteilung N… vom 23. September 1998;

NK54 Auszug aus dem Web-Archiv „Wayback Machine“ betreffend die Webseite

„http://www.opengl.org/Documentation/Extensions.html“ vom 6. Mai 1999;

NK55 OpenGL-Programmieranleitung der Version 1.2.1 vom 14. Oktober 1998;

NK56 Auszug aus dem Web-Archiv „Wayback Machine“ betreffend die Webseite „http://www.opengl.org/Documentation/Specs.html“, Mai 1999;

NK57 Auszug aus: Ute Claussen, Programmieren mit OpenGL, Springer-Verlag, 1997, S. 183-212;

NK58 Ausdruck der Webseite „https://isbnsearch.org/isbn/9783540579779“ vom 12. März 2019 betreffend die Veröffentlichung **NK57**;

NK59 Auszug aus dem „OpenGL Programming Guide“, 2. Aufl., 1997, S. 354 bis 357;

NK60 Ausdruck der Webseite „https://isbnsearch.org/isbn/9783540579779“ vom 12. März 2019 betreffend die Veröffentlichung **NK59**;

NK61 Ausdruck der Webseite „https://www.khronos.org/registry/OpenGL/extensions /NV/NV_texture_env_combine4.txt“ vom 18. Januar 2001;

NK62 Programmheft der Game Developers Conference 1999, 15. bis 19. März 1999;

NK63 Michael I. Gold et al., Advanced OpenGL Game Development, 1999, Unterlagen eines auf der dem Programmheft **NK62** entsprechenden Konferenz gehaltenen Tutorials;

NK64 Ausdruck der Webseite „https://paroj.github.io/gltut/History TNT.html“ mit Artikel „Dynamite Combiners“ vom 3. Juli 2019;

NK65 Ausdruck der Webseite „https://www.khronos.org/opengl/wiki/History_ of_Programmability“ vom 15. Januar 2015.

Die Beklagte verweist in ihrem Vorbringen u. a. auf die folgenden Unterlagen:

G1	Wikipedia-Artikel zu DirectX, Stand 22. Februar 2019;
G2	Angelo Corana, Architectural Evolution of N… GPUs for High-Performance Computing, Technical Report, February 2015;
G3	Chris McClanahan et al., History and Evolution of GPU Architecture – A Paper Survey, Copyright 2010;
G4	Wikipedia-Artikel zum Begriff „Software Rendering", Stand 26. Juni 2018;
G5	David B. Kirk et al.: Programming Massively Parallel Processors – A Hands-on Approach, Elsevier, 2010;
G6	Kayvon Fatahalian: How a GPU Works, 2011;
G7	N… Corporation (Hrsg.), Technical Brief, Microsoft DirectX 8: Raising the Ante for Realism in Graphics, mit Vermerk "Confidential";
G8	Elektronik Kompendium: Microsoft DirectX, im Internet abrufbar unter „https://www.elektronik-kompendium.de/sites/com/0811171 htm?https://app.box.com/f...“;
G9	N… Corporation (Hrsg.): GeForce 256 and RIVA TNT Combiners, How to best utilize the per-pixel operations using OpenGL on N… Graphics Processors, Copyright 1999;
G10	Internetdokument „https://www.khronos.org/registry/OpenGL/extensions/NV/ NV_register_combiners.txt", Ausdruck vom 16. Mai 2019.

Nach Meinung der Klägerin zu 1 sind die Gegenstände der Patentansprüche 1 und 8 am Prioritätstag durch die Entgegenhaltungen **NK6**, **NK7**, **NK8**, **NK9** oder **NK10** (mit dem zusätzlichen Offenbarungsgehalt von **NK11**) neuheitsschädlich vorweggenommen, zumindest aber dem Fachmann nahegelegt. Die Klägerin zu 1

stützt ihre Klage zusätzlich auch auf den von der Klägerin zu 2 in Gestalt des PixelFlow-Systems geltend gemachten Stand der Technik.

Nach Ansicht der Klägerinnen zu 2 und 5 ist der Gegenstand des Patentanspruchs 1 nicht neu gegenüber den Druckschriften **NK12**, **NK13** (i. V. m. **NK29**), **NK14**, **NK28** oder **NK31**.

Ferner berufen sich die Klägerinnen zu 1, 2 und 5 darauf, dass der Gegenstand von Patentanspruch 1 des Streitpatents durch die Grafikkarten-Chipsätze RIVA TNT und Voodoo² offenkundig vorbenutzt worden sei. Auch die Gegenstände des Patentanspruchs 8 und der Unteransprüche sind nach Auffassung der Klägerinnen zu 1, 2 und 5 nicht patentfähig.

Der Nichtigkeitsgrund der unzulässigen Erweiterung wird von den Klägerinnen zu 2 und 5 damit begründet, dass die Merkmale M1.1, M1.3, M1.7, M8.1, M8.3, M8.3.1, M8.3.2 und M8.3.3 (zur Merkmalsgliederung siehe unter Abschnitt II.3 der Gründe) gegenüber der ursprünglich eingereichten Anspruchsfassung und Beschreibung (PCT/US00/11907, Seite 15, Zeilen 1 bis 15) wegen Fehlens der unabdingbaren Programm ID eine unzulässige Zwischenverallgemeinerung darstellten. Dies habe zur Folge, dass das Streitpatent die Priorität vom 7. Mai 1999 nicht wirksam in Anspruch nehme, weshalb ihm als Zeitrang sein Anmeldetag vom 2. Mai 2000 zukomme.

Die Klägerinnen zu 1, 2 und 5 beantragen,

> das europäische Patent EP 1 177 531 mit Wirkung für das Hoheitsgebiet der Bundesrepublik Deutschland in vollem Umfang für nichtig zu erklären.

Die Beklagte beantragt zuletzt,

die Klage abzuweisen,

hilfsweise die Klage abzuweisen, soweit sie sich gegen das Streitpatent in der Fassung der in der Reihenfolge ihrer Nummerierung gestellten Hilfsanträge I bis VII, eingereicht mit Schriftsatz vom 21. September 2020, richtet.

Außerdem verteidigt sie die Ansprüche 1 und 8 des Hilfsantrags VII, eingereicht mit Schriftsatz vom 21. September 2020, einzeln.

Darüber hinaus stellt sie – in dieser Reihenfolge – den in der mündlichen Verhandlung überreichten Hilfsantrag VIIa vom 19. November 2020 und den Hilfsantrag VIII aus dem Schriftsatz vom 21. September 2020.

Ausgehend von der untenstehenden Merkmalsgliederung und bei im Übrigen angepassten Rückbezügen erhalten die Patentansprüche in der Fassung der Hilfsanträge I bis VIII, eingereicht mit Schriftsatz vom 21. September 2020, folgende Fassung, bei welcher jeweils der Patentanspruch 12 der erteilten Fassung entfällt:

Patentanspruch 1 gemäß Hilfsantrag I unterscheidet sich von Patentanspruch 1 der erteilten Fassung durch Merkmal **M1´**, das Merkmal **M1** ersetzen soll, und durch das neu hinzugekommene Merkmal **M1.8** (Änderungen gegenüber der erteilten Fassung sind unterstrichen):

M1´ “A system (150) for ~~processing textures~~ providing programmable texture processing for a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments, the system comprising:”

M1.8 "wherein the processing a portion of the plurality of texture portions for a fragment in accordance with the program comprises texture blending."

Dementsprechend enthält Patentanspruch 8 gemäß Hilfsantrag I anstelle des Merkmals **M8** das Merkmal **M8´**, wobei außerdem noch Merkmal **M8.3.4** auf Merkmal **M8.3.3** folgen soll:

M8´ "A method for ~~processing textures~~ providing programmable texture processing of a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments,"

M8.3.4 "wherein the processing step (b) further includes the step of:
(b1) blending the plurality of texture portions in the plurality of texture processors (154; 190; 300) based on the at least one program."

Patentanspruch 1 gemäß Hilfsantrag II unterscheidet sich von Patentanspruch 1 gemäß Hilfsantrag I in der eingereichten englischsprachigen Fassung durch Merkmal **M1.9**, das sich an Merkmal **M1.8** anschließen soll:

M1.9 "wherein the texture processors can adapt to different blending operations on different inputs."

Patentanspruch 8 gemäß Hilfsantrag II unterscheidet sich von Patentanspruch 8 gemäß Hilfsantrag I durch Merkmal **M8.3.5**, das auf Merkmal **M8.3.4** folgen soll:

M8.3.5 "wherein the texture processors can adapt to different blending operations on different inputs."

Patentanspruch 1 gemäß Hilfsantrag III unterscheidet sich von Patentanspruch 1 gemäß Hilfsantrag II durch Merkmal **M1.9´**, das Merkmal **M1.9** ersetzen soll:

M1.9´ "wherein the texture processors are adaptable to perform a variety of texture blends."

Das entsprechende im Patentanspruch 8 gemäß Hilfsantrag III hinzugekommene Merkmal **M8.3.5´**, das Merkmal **M8.3.5** ersetzt, lautet:

M8.3.5´ "wherein the texture processors are adaptable to perform a variety of texture blends."

Patentanspruch 1 gemäß Hilfsantrag IV unterscheidet sich von Patentanspruch 1 gemäß Hilfsantrag I durch die Merkmale **M1.2´**und **M1.5´**, die an die Stelle der Merkmale **M1.2** und **M1.5** treten sollen:

M1.2´ "a memory (156; 160) for storing a portion of a program used by the plurality of texture processors for processing a plurality of texture portions for the plurality of fragments,"

M1.5´ "each of the plurality of texture processors (154; 190; 300) for importing at least a portion of the program and processing a portion of the plurality of texture portions for a fragment of the plurality of fragments in accordance with the program,"

Patentanspruch 8 gemäß Hilfsantrag IV beruht auf Patentanspruch 8 gemäß Hilfsantrag I, wobei Merkmal **M8.3.1´** das Merkmal **M8.3.1** ersetzt:

M8.3.1´ "wherein a portion of the program is imported and used by the plurality of texture processors for processing the plurality of texture portions, the at least one program including a plurality of instructions,"

Patentanspruch 1 gemäß Hilfsantrag V unterscheidet sich von Patentanspruch 1 gemäß Hilfsantrag IV durch Merkmal **M1.7a**, das zwischen den Merkmalen **M1.7** und **M1.8** eingefügt werden soll. Weiterhin wurde Merkmal **M1.5´** durch Merkmal **M1.5** ersetzt.

M1.7a "wherein the fragments, including the texture portions to be processed, and the portion of the plurality of instructions to be performed by a texture processor are provided to one or more texture processors for processing,"

Patentanspruch 8 gemäß Hilfsantrag V geht aus von Patentanspruch 8 gemäß Hilfsantrag IV, wobei zwischen den Merkmalen **M8.3.3** und **M8.3.4** das Merkmal **M8.3.3a** eingefügt wird. Außerdem wird Merkmal **M8.3.1´** durch Merkmal **M8.3.1´´** ersetzt:

M8.3.1´´ "wherein a portion of the program is used by the plurality of texture processors for processing the plurality of texture portions, the at least one program including a plurality of instructions,"

M8.3.3a "wherein the fragments, including the texture portions to be processed, and the portion of the plurality of instructions to be performed by a texture processor are provided to one or more texture processors for processing,"

Patentanspruch 1 gemäß Hilfsantrag VI unterscheidet sich von Patentanspruch 1 gemäß Hilfsantrag V durch Merkmal **M1.10**, das auf Merkmal **M1.8** folgen soll:

M1.10 "wherein each texture processor performs all of the texture blending for the fragment(s) which it receives."

In Patentanspruch 8 gemäß Hilfsantrag VI ist gegenüber Hilfsantrag V nach Merkmal **M8.3.4** noch Merkmal **M8.3.6** angefügt worden:

M8.3.6 "wherein each texture processor performs all of the texture blending for the fragment(s) which it receives."

Patentanspruch 1 gemäß Hilfsantrag VII geht aus von Patentanspruch 1 gemäß Hilfsantrag IV, wobei die Merkmale **M1.11** und **M1.12** hinzugefügt worden sind:

M1.11 "wherein each fragment, including a program identification, is provided to one or more texture processors for processing,"

M1.12 "wherein the at least a portion of the program is imported by each texture processor using the program identification."

Patentanspruch 8 gemäß Hilfsantrag VII beruht auf Patentanspruch 8 gemäß Hilfsantrag IV, wobei noch die Merkmale **M8.3.7** und **M8.3.8** hinzugekommen sind:

M8.3.7 "wherein each fragment includes a program identification, and wherein the method further includes the step of:"

M8.3.8 "(c) fetching a portion of the program using the program identification."

Zusätzlich zu Patentanspruch 12 ist auch Patentanspruch 11 der erteilten Fassung bei im Übrigen angepassten Rückbezügen gestrichen.

Patentanspruch 1 gemäß Hilfsantrag VIII beruht auf Patentanspruch 1 gemäß Hilfsantrag VI, wobei das neue Merkmal **M1.13** auf Merkmal **M1.10** folgen soll:

M1.13 "wherein each texture processor only provides an output when it has completed processing for the fragment."

Patentanspruch 8 gemäß Hilfsantrag VIII geht aus von Patentanspruch 8 gemäß Hilfsantrag VI, wobei noch Merkmal **M8.3.9** hinzugekommen ist:

M8.3.9 "wherein each texture processor only provides an output when it has completed processing for the fragment."

In der Fassung des erstmals in der mündlichen Verhandlung gestellten Hilfsantrags VIIa vom 19. November 2020 erhalten die Patentansprüche folgende Fassung:

Patentanspruch 1 gemäß Hilfsantrag VIIa geht aus von Patentanspruch 1 gemäß Hilfsantrag VII, wobei Merkmal **M1.12** durch Merkmal **M1.12´** ersetzt werden soll:

M1.12´ "wherein the at least a portion of the program is used by each texture processor using the program identification."

Patentanspruch 8 gemäß Hilfsantrag VIIa ist mit Patentanspruch 8 gemäß Hilfsantrag VII identisch.

Zusätzlich zu Patentanspruch 12 ist auch Patentanspruch 11 der erteilten Fassung bei im Übrigen angepassten Rückbezügen gestrichen.

Die Beklagte widerspricht dem Vortrag der Klägerinnen in allen Punkten. Die behaupteten offenkundigen Vorbenutzungen hält sie für nicht nachgewiesen; u. a. stellt sie sich auf den Standpunkt, dass ein Auszug aus einem Internetarchiv (wie im Fall von **NK20** mittels der Internet-Seite **NK20a**) nicht ausreichend sei.

Außerdem ist die Beklagte der Ansicht, dass den Klägerinnen zu 1 und 2 für ihre Klagen ein Rechtsschutzinteresse fehle: Das Streitpatent sei inzwischen erloschen. Das Verletzungsverfahren, an dem die Klägerin zu 1 als Nebenintervenientin der Verletzungsbeklagten, der B… AG, beteiligt gewesen sei, sei aufgrund der Einigung der Beklagten mit der B… AG nicht mehr anhängig. Es sei nicht ersichtlich, unter welchen Umständen eine Inanspruchnahme der Klägerin zu 1 noch in Frage kommen könnte. Die Klägerin zu 2 sei weder Verletzungsbeklagte gewesen noch einem hinsichtlich des Streitpatents anhängigen Verletzungsverfahren beigetreten noch sei sie Lieferantin der Klägerin zu 5. Die Klägerin zu 2 könne unter keinem rechtlichen Gesichtspunkt aus dem Streitpatent in Anspruch genommen werden, ebenso wenig könnten sich Regressansprüche der Klägerin zu 5 unmittelbar gegen die Klägerin zu 2 richten.

Die Klägerinnen zu 1 und 2 vertreten die Auffassung, für ihre jeweiligen Klagen bestehe auch nach dem Erlöschen des Streitpatents ein Rechtsschutzinteresse. Auch nach Rücknahme der Verletzungsklage der Beklagten gegen die B… AG, an welcher die Klägerin zu 1 als Nebenintervenientin beteiligt war, sei nicht auszuschließen, dass die Klägerin zu 1 von der Beklagten oder der B… AG im Wege der Freistellung aus dem Streitpatent in Anspruch genommen werde; vom Inhalt des zwischen der B… AG und der Beklagten geschlossenen Vergleichs habe die Klägerin zu 1 keine Kenntnis, sie sei nicht in den Vergleich einbezogen. Die Klägerin zu 2 habe es im N…-Konzern übernommen, die Nichtigkeitsklage zu betreiben und damit die Klägerin zu 5 als Beklagte im parallelen Verletzungsverfahren zu verteidigen. Die Klägerin zu 2 verfüge über eine Rechtsabteilung für Rechtsangelegenheiten des N…-Konzerns für Europa.

Die jeweiligen Gegenstände der unabhängigen Patentansprüche nach den Hilfsanträgen I bis VIII der Beklagten in der Fassung vom 21. September 2020 halten die Klägerinnen zu 1, 2 und 5 für nicht patentfähig. Die Hilfsanträge IV bis VIII in der Fassung vom 21. September 2020 halten sie überdies für unzulässig erweitert. Die Klägerin zu 1 sieht die Patentfähigkeit durch die Druckschrift **NK28** in

Frage gestellt, die Klägerinnen zu 2 und 5 beziehen sich auf die Druckschriften zum Thema „PixelFlow“ (**NK12**, **NK13**, **NK31**) und auf die behauptete offenkundige Vorbenutzung des RIVA TNT-Grafikkarten-Chipsatzes.

Die Einreichung des Hilfsantrags VIIa in der Fassung vom 19. November 2020 rügen die Klägerinnen zu 1, 2 und 5 als verspätet.

Der Senat hat den Parteien mit Schreiben vom 19. August 2020 einen qualifizierten gerichtlichen Hinweis zukommen lassen.

Einen Klageverzicht auf Ansprüche aus dem Streitpatent hat die Beklagte nicht erklärt.

Wegen des Vorbringens der Parteien im Übrigen wird auf deren Schriftsätze mit sämtlichen Anlagen und auf das Protokoll der mündlichen Verhandlung vom 19. November 2020 verwiesen.

Entscheidungsgründe

Die Klagen der Klägerinnen zu 1, 2 und 5 sind zulässig und begründet. Denn das Streitpatent hat weder in der erteilten Fassung noch in der Fassung einer der Hilfsanträge Bestand, da ihm der geltend gemachte Nichtigkeitsgrund der fehlenden Patentfähigkeit (Art. II § 6 Abs. 1 Nr. 1 IntPatÜG, Art. 138 Abs. 1 lit. a) EPÜ i. V. m. Art. 54 Abs. 1, 2 und Art. 56 EPÜ) entgegensteht.

Es bedarf daher keiner Entscheidung, ob dem Streitpatent auch der weiterhin geltend gemachte Nichtigkeitsgrund der unzulässigen Erweiterung (Art. II § 6 Abs. 1 Nr. 3 IntPatÜG i. V. m. Art. 138 Abs. 1 lit. c) EPÜ) entgegensteht.

I.

Die Klagen der Klägerinnen zu 1, 2 und 5 sind auch nach dem Erlöschen des Streitpatents aufgrund Zeitablaufs am 2. Mai 2020 weiterhin zulässig. Auch der Beitritt der Klägerin zu 5 ist wirksam erfolgt.

1. Ist ein Patent durch Zeitablauf erloschen, bedarf es eines schutzwürdigen Interesses des Klägers an der Durchführung des Nichtigkeitsverfahrens, da ein Interesse der Allgemeinheit an einer Überprüfung der Rechtsbeständigkeit nicht mehr besteht. Die Frage, ob ein solches eigenes Rechtsschutzinteresse vorliegt, darf nicht nach allzu strengen Maßstäben beurteilt werden. Es ist gegeben, wenn der Kläger wegen Verletzung des Patents in Anspruch genommen wird oder für den Kläger Grund zu der Besorgnis besteht, er könne aus dem Patent wegen Handlungen in der Zeit vor dessen Erlöschen in Anspruch genommen werden. Ein Rechtsschutzinteresse ist in solchen Fällen nur zu verneinen, wenn eine solche Inanspruchnahme ernstlich nicht mehr in Betracht kommt (st. Rspr., zuletzt BGH GRUR 2021, 42, Rn. 7 – *Truvada* m. w. N.). Letzteres ist nicht der Fall.

1.1 Die Verletzungsklage vor dem Landgericht Mannheim (…) gegen die B… AG (das ist die vormalige Klägerin zu 4 aus dem zeitweise hier hinzuverbundenen Nichtigkeitsverfahren 7 Ni 81/19 (EP)), dem die Klägerin zu 1, weil sie Zulieferin war, als Nebenintervenientin beigetreten ist, ist zwar von der Verletzungsklägerin und hiesigen Beklagten zurückgenommen worden. Jedoch besteht das Rechtsschutzinteresse an einer Patentnichtigkeitsklage grundsätzlich auch für den Fall fort, dass ein Patentinhaber eine bereits erhobene Verletzungsklage zurücknimmt, einen Verzicht auf eventuelle Ansprüche aus dem Patent aber nicht erklärt (vgl. BGH GRUR 2010, 1084, Rn. 10 – *Windenergiekonverter*; GRUR 2020, 1074, Rn. 29 – *Signalübertragungssystem*).

Ein vergleichbarer Fall liegt hier vor. Die Beklagte hat durch ihre oben bezeichnete, bei Erhebung der Nichtigkeitsklage noch anhängige Verletzungsklage zum Ausdruck gebracht hat, dass sie gewillt ist, ihr nach ihrer Auffassung zustehende Ansprüche wegen Verletzung des Streitpatents durchzusetzen. Hiervon ist sie bis zum Schluss der mündlichen Verhandlung nicht abgerückt. Rechtlich gehindert wäre sie nur in einem Fall des Klageverzichts auf Ansprüche aus dem Streitpatent. Einen solchen hat die Beklagte aber gegenüber der Klägerin zu 1 auf Nachfrage des Senats in der mündlichen Verhandlung nicht erklärt. Bei dieser Ausgangslage ist die Besorgnis der Klägerin zu 1 als Zulieferin der B… AG nicht unbegründet, dass sie von der Beklagten wegen Handlungen in der Zeit vor Erlöschen des Streitpatents wegen Verletzung in Anspruch genommen wird. Hiervon abgesehen könnte die Klägerin zu 1 auch Regressansprüchen der B… AG ausgesetzt sein, zumal der Inhalt des Vergleichs aus dem Verletzungsverfahren gegen die B… AG nicht bekannt ist.

1.2 Auch der Klägerin zu 2, die zwar selbst weder unmittelbar wegen Verletzung des Streitpatents in Anspruch genommen wird noch Zulieferin ist, kann ein Rechtsschutzinteresse an der Fortführung der Nichtigkeitsklage nicht abgesprochen werden. Nach der BGH-Entscheidung „Tafelförmige Elemente“ (BGH GRUR 1995, 342), auf die die Beklagte verweist, lässt sich das erforderliche

Rechtsschutzinteresse nicht allein damit begründen, dass der Kläger Mehrheitsgesellschafter einer GmbH ist, die wegen Verletzung des Schutzrechts in Anspruch genommen wird. Die vorliegenden Umstände sind damit aber nicht vergleichbar, da es um arbeitsteiliges Vorgehen im Konzern geht.

Die Klägerin zu 2 gehört zum N…-Konzern und verfügt nach eigenen Angaben, an deren Richtigkeit der Senat keinen Anlass hatte zu zweifeln, über eine Rechtsabteilung für Rechtsangelegenheiten des Konzerns für Europa, und hat es im N…-Konzern übernommen, die vorliegende Nichtigkeitsklage zu betreiben, um die wegen Patentverletzung in Anspruch genommene Klägerin zu 5 als Beklagte im parallelen Verletzungsverfahren vor dem Landgericht Mannheim (Aktenzeichen …) zu verteidigen. Die hiesige Beklagte hat als Verletzungsklägerin in ihrer Klageschrift vom 25. Mai 2018 (Anlage NK7 zum Verfahren 7 Ni 35/19 (EP)) insoweit selbst vorgetragen (Seiten 17, 18), dass in die Spielkonsole „Nintendo Switch“, als deren Vertreiberin in Deutschland die Klägerin zu 5 verklagt wird, das Ein-Chip-System N… T210 (N… Tegra X1) des „Chipherstellers N… verbaut“ sei und das genannte Ein-Chip-System alle Merkmale des Streitpatents verwirkliche. Ungeachtet einer näheren Auseinandersetzung mit der im konkreten Einzelfall gewählten Form arbeitsteiligen Zusammenwirkens im Konzern ist das auch nach Patentablauf bestehende rechtliche Interesse des Chipherstellers, die Inanspruchnahme einer Abnehmerin abzuwenden – die Verletzungsklage gegen die Abnehmerin ist weiterhin anhängig -, der Klägerin zu 2 hier zuzurechnen, da sie als mit dem Chiphersteller verbundenes Unternehmen nur zu diesem Zweck die Nichtigkeitsklage erhoben hat (vgl. zur Berücksichtigung verbundener Unternehmen BGH GRUR 2021, 42, Rn. 9 – *Truvada*). Eine Verzichtserklärung, die das Rechtsschutzinteresse in der gegebenen Situation hätte entfallen lassen können, hat die Beklagte nicht abgegeben (vgl. BGH GRUR 2020, 1284, Rn. 51 – *Datenpaketumwandlung*; GRUR 2010, 1084 – *Windenergiekonverter*).

1.3. Hinsichtlich der Klägerin zu 5 ergibt sich das erforderliche Rechtsschutzinteresse ohne Weiteres daraus, dass sie die Verletzungsbeklagte in dem oben unter 1.2 bezeichneten Verletzungsverfahren vor dem Landgericht Mannheim ist.

2. Die Klägerin zu 5 ist auch wirksam der Nichtigkeitsklage 7 Ni 35/19 (EP) als weitere Klägerin beigetreten.

Die Parteierweiterung auf Klägerseite ist nach ständiger Rechtsprechung wie eine Klageänderung nach § 263 ZPO zu behandeln (vgl. BGHZ 65, 264; BGH NJW 1989, 3225; Zöller/Greger, ZPO, 33. Aufl., § 263 Rn. 27). Nachdem die Beklagte dem Klagebeitritt nicht zugestimmt hat, ist die Klageänderung nur zulässig, wenn sie sachdienlich ist. Hiervon ist aus prozessökonomischen Gründen auszugehen. Da gegen die Beitretende eine auf Grundlage des Streitpatents geführte Verletzungsklage anhängig ist, wird damit ein neuer Prozess, d. h. eine eigene Klage der Beitretenden, die sie jederzeit (und zwar auch zu einem späten Zeitpunkt) selbst erheben kann, vermieden (vgl. BPatGE 32, 204, 205; BPatG, Urteil vom 8. Dezember 2016 – 2 Ni 5/15 (EP) unter Gründe I.1, juris Rn. 113).

II.

1. Das Streitpatent betrifft das Anzeigen einer Grafikabbildung auf einem Computersystem und insbesondere ein Verfahren und System, um eine programmierbare Texturverarbeitung für eine Grafikabbildung bereitzustellen (Streitpatentschrift, Abs. [0001]).

Die Bezeichnung „Struktur“ in der deutschen Fassung des Streitpatents (DE 600 07 521 T2) steht für „texture“ in der englischen Sprachfassung und stellt eine missverständliche Übersetzung dar. Im Folgenden wird im Deutschen der technisch zutreffende Begriff „Textur“ anstelle von „Struktur“ verwendet.

Laut Beschreibungseinleitung stellen herkömmliche Computergrafiksysteme grafische Bilder von Objekten auf einer Anzeige bzw. einem Display dar. Das Display umfasst eine Mehrzahl von Displayelementen, sogenannten Pixeln, die üblicherweise in einem Gitter angeordnet sind. Um Objekte anzuzeigen, unterteilt ein herkömmliches Grafiksystem jedes Objekt in eine Mehrzahl von Polygonen, welche dann in einer bestimmten Reihenfolge „gerendert“ werden (Streitpatentschrift, Abs. [0003]).

Jedes Polygon überdeckt oder schneidet eine gewisse Anzahl von Pixeln des Displays. Die Daten für den Teil eines Polygons, der ein bestimmtes Pixel schneidet, werden „Fragment“ für dieses Polygon und dieses Pixel genannt. Sie betreffen typischerweise die Farbe, den Mischungsmodus („blending mode“) und die Textur. Um ein Fragment eines Polygons darzustellen (zu „rendern“), muss das Grafikverarbeitungssystem die Farbe und Textur für das Fragment verarbeiten (Streitpatentschrift, Abs. [0004]). Dies geschieht beispielsweise durch ein Mischen der Farbe und der Textur, um einen endgültigen Farbwert für das Fragment an dem entsprechenden Pixel zu erhalten (Streitpatentschrift, Abs. [0005]).

Das Streitpatent beschreibt eine Textur als eine andere Farbe, die gewöhnlich von einer Texturkarte abgeleitet wird (Streitpatentschrift, Abs. [0005]). Eine solche Textur wird in der Grafikverarbeitung verwendet, um die Oberfläche eines Objekts mit einem „Überzug“ zu versehen, welcher etwa die Farben der Oberfläche, deren optische Beschaffenheit oder auch weitere Effekte wie Lichtreflexe definiert und auf das Objekt aufbringt (Streitpatentschrift, Abs. [0005], [0006]). Dabei werden auch mehrere Texturen für ein Objekt verwendet, um unterschiedliche Effekte schichtweise auf die Oberfläche aufzubringen. Dies bezeichnet das Streitpatent als Mehrfachtexturabbildung (Streitpatentschrift, Abs. [0006]).

Im Streitpatent geht es grundsätzlich darum, den Fragmenten eines abzubildenden Objekts entsprechende Texturwerte zuzuordnen. Ein Objekt wird letztlich auf einer Anzeigevorrichtung dargestellt, die eine Mehrzahl von Pixeln aufweist. Deshalb ist

es zum Zwecke der Darstellung erforderlich, die Daten eines Fragments so zu verarbeiten, dass daraus ein Wert für das jeweilige Pixel der Anzeigevorrichtung erhalten wird. Wie die für die mehrfache Texturabbildung notwendigen Berechnungen im Stand der Technik ausgeführt werden konnten, wird im Streitpatent u. a. anhand der Figuren 3 und 4 erläutert: Eine Lösung bestand darin, eine einzelne Texturmischeinheit („Texture Blending Unit“) zu verwenden, welche sämtliche Texturberechnungen für ein Fragment nacheinander, also seriell ausführt, wobei das Ergebnis jeder Berechnung in einer Schleife an dieselbe Texturmischeinheit zurückgeführt wird, um dann die nächste Textur darauf anzuwenden (Streitpatentschrift, Abs. [0018]). Der zugehörige Aufbau ist in Figur 4 gezeigt. Eine andere Variante bestand darin, mehrere Texturmischeinheiten kaskadiert in Reihe zu schalten, so dass jede Texturmischeinheit das Ergebnis der vorhergehenden Stufe verwendet, um dann anhand einer weiteren Textur Berechnungen für ein Fragment auszuführen (Streitpatentschrift, Abs. [0017]). Diese Variante ist in Figur 3 des Streitpatents wiedergegeben.

An solchen und ähnlichen herkömmlichen Grafiksystemen wird kritisiert, dass sie entweder in ihrer Funktion zu beschränkt und zu unflexibel seien, oder aber einen komplexen Aufbau erforderten. Dies liege daran, dass sowohl mehrere Fragmente als auch mehrere Texturen nur hintereinander abgearbeitet werden könnten und die Texturmischeinheiten speziell auf bestimmte Mischoperationen ausgebildet werden müssten (Streitpatentschrift, Abs. [0008]).

2. Ausgehend vom bekannten Stand der Technik stellt sich das Streitpatent daher die Aufgabe, ein System und ein Verfahren bereitzustellen, das einen „adaptierbaren Mechanismus zum Bereitstellen einer Texturverarbeitung ohne nachteilige Auswirkung auf die Leistung des Systems“ aufweist (Streitpatentschrift, Abs. [0009]). Die Texturverarbeitung soll in vergleichsweise kleinen Einheiten stattfinden können und dabei gleichzeitig flexibel, d. h. adaptierbar in Hinblick auf die ausgeführten Funktionen sein, um den genannten Problemen zu begegnen (Streitpatentschrift, Abs. [0008], [0009], [0022])

3. Um diese Aufgabe zu lösen, beschreiben die unabhängigen Patentansprüche 1 und 8 ein System und ein Verfahren zur Texturverarbeitung für eine Grafikabbildung auf einer Anzeige, mit folgendem Wortlaut (englisch: gemäß Streitpatentschrift, deutsch: teilweise verbesserte Übersetzung):

Patentanspruch 1 (in Anlehnung an die von der Klägerin zu 1 vorgeschlagene Merkmalsgliederung):

M1	A system (150) for processing textures for a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments, the system comprising:	Ein System (150) zur Verarbeitung von Texturen für eine Grafikabbildung auf einer Anzeige (1; 104), wobei die Grafikabbildung ein Objekt umfasst, und das Objekt eine Mehrzahl von Fragmenten umfasst, und das System beinhaltet:
M1.1	a plurality of texture processors (154; 190; 300) for processing a portion of the plurality of fragments; [characterised by:]	eine Mehrzahl von Texturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der Mehrzahl der Fragmente; [gekennzeichnet durch:]
M1.2	a memory (156; 160) for storing a portion of a program for processing a plurality of texture portions for the plurality of fragments,	einen Speicher (156; 160) zum Speichern eines Teils eines Programms zur Verarbeitung einer Mehrzahl von Texturabschnitten für die Mehrzahl der Fragmente,
M1.3	the portion of the program including a plurality of instructions;	wobei der Teil des Programms eine Mehrzahl von Anweisungen umfasst;

M1.4	said plurality of texture processors (154; 190; 300) coupled with the memory (156; 160),	wobei die Mehrzahl der Texturprozessoren (154; 190; 300) mit dem Speicher (156; 160) gekoppelt ist;
M1.5	each of the plurality of texture processors (154; 190; 300) for processing a portion of the plurality of texture portions for a fragment of the plurality of fragments in accordance with the program,	wobei jeder der Mehrzahl der Texturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der Mehrzahl der Texturabschnitte für ein Fragment der Mehrzahl der Fragmente gemäß dem Programm eingerichtet ist;
M1.6	the plurality of texture processors (154; 190; 300) capable of processing a second portion of the plurality of texture portions in parallel,	wobei die Mehrzahl der Texturprozessoren (154; 190; 300) in der Lage ist, einen zweiten Teil der Mehrzahl der Texturabschnitte parallel zu verarbeiten;
M1.7	the plurality of texture processors (154; 190; 300) implementing a portion of the plurality of instructions.	wobei die Mehrzahl der Texturprozessoren (154; 190; 300) einen Teil der Mehrzahl der Anweisungen implementiert.

Patentanspruch 8 (hier mit einer denkbaren Gliederung versehen):

M8	A method for processing textures of a graphical image on a display (1; 104), the graphical image including an object, the object including a plurality of fragments,	Ein Verfahren zur Texturverarbeitung für eine Grafikabbildung auf einer Anzeige (1; 104), wobei das Grafikabbild ein Objekt umfasst und das Objekt eine Mehrzahl von Fragmenten beinhaltet

M8.1	the system comprising a plurality of texture processors (154; 190; 300) for processing a portion of the plurality of fragments;	und das System eine Mehrzahl von Texturprozessoren (154; 190; 300) zur Verarbeitung eines Teils der Mehrzahl der Fragmente umfasst;
	the method characterised by the steps of:	wobei das Verfahren gekennzeichnet ist durch die Schritte:
M8.2	(a) providing a plurality of texture portions for the plurality of fragments to a plurality of texture processors (154; 190; 300), and	(a) Bereitstellen einer Mehrzahl von Texturabschnitten für die Mehrzahl von Fragmenten an eine Mehrzahl von Texturprozessoren (154; 190; 300); und
M8.3	(b) processing the plurality of texture portions in the plurality of texture processors (154; 190; 300) in parallel based on at least one program,	(b) paralleles Verarbeiten der Mehrzahl von Texturabschnitten in der Mehrzahl von Texturprozessoren (154; 190; 300) auf Grundlage von wenigstens einem Programm;
M8.3.1	the at least one program including a plurality of instructions,	wobei das wenigstens eine Programm eine Mehrzahl von Anweisungen aufweist,
M8.3.2	the plurality of texture processors (154; 190; 300) implementing a portion of the plurality of instructions,	die Mehrzahl von Texturprozessoren (154; 190; 300) einen Teil der Mehrzahl der Anweisungen implementiert
M8.3.3	the at least one program thereby controlling processing of the plurality of texture portions by the	und wenigstens ein Programm auf diese Weise das Verarbeiten der Mehrzahl der Texturabschnitte durch die Mehrzahl der

	plurality of texture processors (154; 190; 300).	Texturprozessoren (154; 190; 300) steuert.

4. Als Fachmann, der mit der Aufgabe betraut wird, ein System bzw. Verfahren zur Texturverarbeitung in der Grafikverarbeitung zu verbessern, ist ein Diplomingenieur der Fachrichtung Elektrotechnik oder Informatik anzusehen, der mehrjährige Berufserfahrung auf dem Gebiet des Designs von Grafikchips zur Anwendung im Bereich von Grafikkarten hat und der insbesondere über fundierte Kenntnisse in der Bildsynthese verfügt.

5. Dieser Fachmann legt den Merkmalen des Patentanspruchs 1 folgendes Verständnis zugrunde:

a) Zum Begriff Textur bzw. Struktur

Für den englischen Begriff *texture* im Original des Streitpatents verwendet die zugehörige T2-Schrift den Begriff *Struktur*. Beide Begriffe betreffen denselben Bedeutungsinhalt: eine *Textur* bezeichnet in der Computergrafik in der Regel einen „Überzug“ für ein 3D-Modell eines Objekts, der die Beschaffenheit der Objektoberfläche beschreibt und dem Objekt eine Oberflächenstruktur verleiht. Insoweit entspricht eine *Textur* gleichzeitig einer *Struktur*. Ausgehend vom Streitpatent in der englischen Originalfassung wird im Folgenden abweichend von der T2-Schrift der technisch zutreffendere Begriff *Textur* anstelle von *Struktur* verwendet.

b) Zum Begriff Fragment

Um Objekte auf dem Display eines Grafiksystems anzuzeigen, wird jedes Objekt in eine Vielzahl von Polygonen aufgeteilt (Streitpatentschrift, Abs. [0003]). Jedes der Polygone bedeckt oder schneidet eine bestimmte Anzahl von Pixeln in der Anzeige. Die Daten für einen Abschnitt eines Polygons, das ein bestimmtes Pixel schneidet, werden als *Fragment* für dieses Polygon und dieses Pixel bezeichnet

(Streitpatentschrift, Abs. [0004], siehe „Data for a portion of a polygon which intersects a particular pixel is termed the fragment for that polygon and that pixel.”). Jedes Polygon beinhaltet gewöhnlich eine Mehrzahl von *Fragmenten*, wobei jedes *Fragment* Daten beinhaltet, die dasjenige Pixel betreffen, in dem das jeweilige *Fragment* liegt. Laut Streitpatent umfasst ein *Fragment* Informationen bezüglich Farbe, Mischungsmodi und Textur (Abs. [0004]). Zwar korrespondiert ein *Fragment* mit einem Pixel, d. h. einem auf der Anzeige angeordneten Bildpunkt, jedoch handelt es sich dabei nicht schon um ein zur Anzeige bereites Pixel, sondern um eine Art Vorstufe, die diejenigen Daten beinhaltet, die zur Berechnung und Darstellung eines Pixels benötigt werden (Streitpatentschrift, Abs. [0004], [0014]).

c) Zum Begriff Textur- bzw. Strukturabschnitt

Einem Fragment ist gemäß dem Streitpatent eine Textur zugeordnet (Abs. [0005], siehe „texture for the fragment“), die als eine weitere Farbe für das Fragment angesehen werden kann und aus einer Texturkarte *(texture map)* abgeleitet wird. Diese Textur wird mit der Farbe des Fragments zusammengemischt, um einen endgültigen Farbwert für das Fragment zu erhalten. Zu diesem Zweck ruft das Grafiksystem einen bestimmten Abschnitt der Textur ab, der üblicherweise einem Pixel entspricht und als Elementarmuster die grundlegende Einheit der Textur in der Computergrafik bildet. Ein solcher *Texturabschnitt* – auch *Texel* genannt – beinhaltet laut Streitpatent einen Farb- und einen Alpha-Wert, der eine Transparenzinformation für einen Bildpunkt bzw. ein Pixel angibt (Streitpatentschrift, Abs. [0005]).

d) Zum Begriff Programm

Anspruchsgemäß steuert ein *Programm* die Texturverarbeitung auf einer Mehrzahl von Texturprozessoren (Streitpatentschrift, Abs. [0033]). Infolge dieses *Programms* können die Texturprozessoren an verschiedene Mischoperationen angepasst werden (Streitpatentschrift, Abs. [0036]). Der Fachmann versteht unter dem *Programm* ganz allgemein eine in einer bestimmten Programmiersprache oder in

Maschinencode festgelegte Abfolge von Anweisungen bzw. Instruktionen, die zur Texturverarbeitung auf den Texturprozessoren ausgeführt werden.

e) Zur Lehre des erteilten Patentanspruchs 1

Der Patentanspruch 1 lehrt ein System, das der Verarbeitung von Texturen für eine Grafikabbildung auf einer Anzeige dient. Dabei umfasst die Grafikabbildung ein Objekt, das sich aus einer Mehrzahl von Fragmenten zusammensetzt (Merkmal **M1**).

Im Streitpatent bildet das System einen Teil der *Image Generating Unit 120*, die eine Grafikabbildung auf einer Anzeige erzeugt (Streitpatentschrift, Abs. [0027], Fig. 6). Das System selbst wird als *Textureinheit 150 (texture unit)* bezeichnet. Das darzustellende Objekt wird durch eine Mehrzahl von Fragmenten bestimmt, wobei die Daten eines Fragments die Farbe, die Textur, den Alpha-Wert und Tiefenwerte eines bestimmten Polygons und Pixels betreffen (Streitpatentschrift, Abs. [0014]).

Merkmal **M1.1** besagt, dass das beanspruchte System über eine Mehrzahl von Texturprozessoren verfügt, die der Verarbeitung eines Teils der Mehrzahl der Fragmente dienen. In der bevorzugten Ausführungsform weist ein Texturprozessor eine bitweise Logik, einen Multiplexer/Akkumulator und eine Summierungseinheit auf (Streitpatentschrift, Abs. [0049], Fig. 11A).

Gemäß Merkmal **M1.2** umfasst das System einen Speicher, der der Hinterlegung eines Teils eines Programms zur Verarbeitung einer Mehrzahl von Texturabschnitten dient. Als Speicher kommt jede Art von Speicher unabhängig von seiner Lage innerhalb des Systems in Frage (vgl. Streitpatentschrift, Abs. [0033]). So kann es sich bei dem Speicher laut Streitpatent etwa um einen Zwischenspeicher (Fig. 9, Program Cache 160) oder einen Festplattenspeicher handeln (Fig. 6, Memory 110). Weiterhin wird in Absatz [0042] der Streitpatentschrift ausgeführt, dass der Zwischenspeicher das komplette Programm zur Texturverarbeitung beinhaltet. Alternativ wird vorgeschlagen, dort nur einen Teil des

Programms vorzuhalten. Der erteilte Patentanspruch 1 verlangt allerdings *nicht,* dass im Speicher *nur ein* Teil des Programms hinterlegt sein muss; der Patentanspruch 1 schließt nicht aus, dass der Speicher mehrere, z. B. auch sämtliche Teile des Programms umfasst und nicht bloß einen einzigen Teil davon. Zur Texturverarbeitung können die einzelnen Teile des anspruchsgemäßen Programms in gleichwirkender Weise sowohl auf einem einzigen Speicher untergebracht als auch über eine Mehrzahl von Speichern verteilt sein.

In Merkmal **M1.3** wird beansprucht, dass der Teil des Programms eine Mehrzahl von Anweisungen beinhalten soll. Bei den Anweisungen handelt es sich um beliebige Anweisungen, die an die Texturprozessoren übergeben werden und die Texturverarbeitung von Fragmenten steuern.

Merkmal **M1.4** besagt, dass die Mehrzahl der Texturprozessoren an den Speicher gekoppelt ist. Laut Streitpatent können die Texturprozessoren und der Speicher über einen *Distributor* und einen *Argument Decoder* miteinander verknüpft sein (Streitpatentschrift, Abs. [0032], [0039], Fig. 7, 9). Beide Komponenten sind dafür verantwortlich, dass die Anweisungen des im Speicher hinterlegten Programms an die Texturprozessoren verteilt werden. Allerdings ist Merkmal **M1.4** nicht auf eine solche indirekte Verknüpfung beschränkt. Vielmehr sind sämtliche physikalische Verbindungen mit umfasst, die das Laden von Anweisungen aus dem Speicher in die Texturprozessoren erst ermöglichen.

Weiterhin ist jeder der Texturprozessoren dazu ausgelegt, einen Teil der Texel zu verarbeiten, wobei diese Texel zu einem Fragment gehören (Merkmal **M1.5**). Dem Anspruchswortlaut nach kann es sich bei dem zu verarbeitenden Teil der Texel auch um nur ein einziges Texel handeln.

Außerdem sollen die Texturprozessoren in der Lage sein, die Texturverarbeitung parallel durchzuführen. So sind die Texturprozessoren gemäß Merkmal **M1.6** dazu eingerichtet, einen zweiten Teil der Texel parallel zu verarbeiten. Der Fachmann

wird Merkmal **M1.6** so verstehen, dass sich der „zweite Teil der Texel“ von dem in Merkmal **M1.5** genannten Teil dadurch unterscheidet, dass er andere Texel einer Texturkarte desselben Fragments betrifft oder aber überhaupt zu einem anderen Fragment gehört. Wie sich den Ausführungen in Absatz [0032] der Streitpatentschrift entnehmen lässt, sollen die Texturprozessoren die Texel für die entsprechenden Fragmente verarbeiten. Demnach führt jeder Texturprozessor die gesamte Texturverarbeitung für das Fragment oder die Fragmente durch, das bzw. die er erhält. Alternativ kann ein Texturprozessor auch nur einen Teil der Texturverarbeitung für ein Fragment durchführen, wobei der Rest der Texturverarbeitung von einem anderen Texturprozessor übernommen wird (Streitpatentschrift, Spalte 9, Zeilen 40 bis 45). Merkmal **M1.6** ist demnach bereits dann erfüllt, wenn die Texturprozessoren mehrere Texel parallel verarbeiten können unabhängig davon, ob sie zum selben Fragment gehören oder nicht. Allerdings ist Patentanspruch 1 so zu verstehen, dass in Übereinstimmung mit Merkmal **M1** die parallel verarbeiteten Texel zu Fragmenten desselben Objekts gehören.

In Merkmal **M1.7** wird beansprucht, dass die mehreren Texturprozessoren einen Teil der Anweisungen „implementieren“. Damit ist – entsprechend der Bedeutung des englischen Verbs „to implement“ (zu Deutsch „durchführen“, „umsetzen“, „verwirklichen“, „erfüllen“) – gemeint, dass die Texturprozessoren in der Lage sind, Programmanweisungen auszuführen, und dass ein vom Benutzer erstellter Programmcode auf die Texturprozessoren übertragen werden kann, der vorgibt, was diese ausführen sollen.

III.

Das Streitpatent hat in der erteilten Fassung keinen Bestand, weil die jeweiligen Gegenstände der unabhängigen Patentansprüche 1 und 8 nicht patentfähig sind.

1. Die Lehre des erteilten Patentanspruchs 1 ist nicht neu gegenüber dem der Druckschrift **NK12** entnehmbaren Stand der Technik.

So führt die Druckschrift **NK12** den Fachmann zu *PixelFlow*, einem System zur Bildsynthese und zur Erzeugung von Grafikabbildungen auf einer Anzeige (Seite 57, Abstract, siehe „PixelFlow is an architecture for high-speed, highly realistic image generation, based on the techniques of object-parallelism and image-composition.“; Fig. 2; Seite 67, rechte Spalte, Abschnitt „Summary“, zweiter Absatz).

PixelFlow verarbeitet u. a. Texturen, was im Rahmen des *Shadings* stattfindet, bei dem endgültige Pixelfarbwerte auf der Grundlage von Texturen und Rohpixeldaten berechnet werden (Seite 57, linke Spalte, Abschnitt „Introduction“, siehe „… to implement realistic rendering techniques such as user-programmable shading, texturing, antialiasing, and shadows.“; Seite 58, linke Spalte, Abschnitt „System Operation“, zweites Aufzählungszeichen, siehe „Shaders apply texture and lighting models to regions of raw pixel data, producing RGB color values …“; Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values …“).

Die von *PixelFlow* erzeugten Grafikabbildungen umfassen verschiedene Objekte, z. B. Buchstaben (vgl. Fig. 2), die sich aus einer Vielzahl von geometrischen *Primitiven*, z. B. Polygonen, Kugeln oder Quadriken zusammensetzen (Seite 67, rechte Spalte, Abschnitt „Summary“, zweiter Absatz, siehe „… PixelFlow can render primitives such as spheres, quadrics, and volume data with high-quality shading methods …“), die durch *Renderer* auf Bildschirmkoordinaten abgebildet worden sind (Seite 58, linke Spalte, Abschnitt „System Operation“, erstes Aufzählungszeichen).

Nachdem jeder *Renderer* von *PixelFlow* für denselben vorgegebenen Bildschirmbereich seinen Anteil an *Primitiven* gerastert hat, werden die von den einzelnen Renderern erzeugten Rohpixeldaten *(raw pixel attributes)*, die diesen

Bildschirmbereich betreffen, über ein Netzwerk *(composition network)* zusammengefasst und zur weiteren Verarbeitung an einen *Shader* übergeben (Fig. 2; Seite 58, linke Spalte, Abschnitt „System Operation“, letzter Absatz, siehe „Once a given region has been rasterized on all of the renderers, the composition network merges the pixel data together and loads the region of composited pixel data onto a shader.“).

Die berechneten Rohpixeldaten stellen noch keine endgültigen Pixelfarbwerte dar, sondern bilden geometrische und spezifische Pixel-Attribute, z. B. Flächennormalen-Vektoren und Farben von Oberflächen (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „… instead, they compute geometric and intrinsic pixel attributes, such as surface-normal vectors and surface color; these attributes, not pixel colors, are composited.“).

Bei den Rohpixeldaten handelt es sich um die anspruchsgemäßen Fragmente, d. h. um Daten für Abschnitte von Primitiven bzw. Polygonen, die bestimmte Pixel schneiden, wobei die Fragmente zusammen ein Objekt ausmachen.

Merkmal **M1** ist somit in Druckschrift **NK12** offenbart.

Die Weiterverarbeitung der Rohpixeldaten erfolgt auf dem für den jeweiligen Bildschirmbereich zuständigen *Shader*, der für die einzelnen Pixel eine Texturverarbeitung durchführt und endgültige Pixelfarbwerte berechnet (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values …“).
Die *Shading*-Berechnungen können für viele Pixel simultan ausgeführt werden (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, letzter Absatz).

Da in dem *SIMD Pixel Processor Array* eines *Shaders* jedem Verarbeitungselement *(PE)* genau ein Pixel bzw. Subpixel zugeordnet wird (Seite 61, linke Spalte, siehe Tabelle; Seite 62, linke Spalte, fünfter Absatz, siehe „The PEs are grouped into sets

of 1, 4 or 8, each group corresponding to a pixel.“), kann jedes Pixel bzw. Subpixel unabhängig auf Grundlage der zugehörigen Rohpixeldaten bzw. Fragmente errechnet werden (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „For ultimate quality rendering, every subpixel sample can be shaded independently.“). Die Verarbeitungselemente *(PE)* in Verbindung mit dem Textur-/Video-Subsystem (Fig. 4; Seite 63, rechte Spalte, Abschnitt „Texture/Video Subsystem“, siehe *TASICs* und *SDRAMs*) fungieren als Texturprozessoren, weil sie u. a. Texturen anwenden (s. o.) und dabei zugleich wenigstens auf den Teil der Rohpixeldaten bzw. Fragmente zurückgreifen, die an den *Shader* übergeben worden sind. Im Ergebnis lehrt die Druckschrift **NK12** damit ein System, das über eine Mehrzahl von Texturprozessoren verfügt und eine Mehrzahl von Fragmenten verarbeitet (Merkmal **M1.1**).

Das *Geometry Processor Board* einer *Shader Flow Unit* umfasst einen SDRAM, der als Hauptspeicher für den *Geometry Processor (GP)* und als FIFO-Speicher für Programmanweisungen an das *Rasterizer Board* mit SIMD-Prozessor und Textur-Subsystem dient (Seite 60, rechte Spalte, Abschnitt „Geometry Processor“, dritter Absatz). Der beschriebene Speicher stimmt mit dem in Figur 3 dargestellten *Program/Data Memory* überein.

Darin sind *Shader*-spezifische Programmanweisungen gepuffert, z. B. solche für das Laden einer neuen Shadingfunktion (Seite 66, linke Spalte, zweiter Absatz). Da die *Shader* mit ihren Verarbeitungselementen *(PE)* Texturen auf eine Mehrzahl von Rohpixeldaten bzw. Fragmenten eines Bildschirmbereichs anwenden (Seite 58, linke Spalte, zweites Aufzählungszeichen, siehe „Shaders apply texture and lighting models to regions of raw pixel data …“), handelt es sich wenigstens bei einem Teil dieser Programmanweisungen um solche für eine Texturverarbeitung, welche laut Druckschrift **NK12** auf der Verarbeitung einer Mehrzahl von Texeln beruht (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values …“).

Die Programmanweisungen gehen im Wesentlichen auf kleine Unterprogramme zurück, die auf einer *Host Workstation* hinterlegt sind, dort ablaufen und die Funktionen der OpenGL API aufrufen (Seite 66, linke Spalte, zweiter Absatz). Demnach ist Merkmal **M1.2** in der Lehre der Druckschrift **NK12** verwirklicht.

Dass der für die Texturverarbeitung zuständige Programmteil tatsächlich über eine Mehrzahl von Anweisungen verfügt, ergibt sich direkt aus den Ausführungen zum Textur-/Video-Subsystem (Abschnitt 3.3). Mehrfaches Nachschlagen in Texturen *(texture-lookups)*, Texturfiltern und Mipmapping (Seite 63, rechter Absatz, Abschnitt „Texture/Video Subsystem“, erster und zweiter Absatz), aber auch die Unterstützung von Funktionen wie *bump mapping* (Simulation von Oberflächen-unebenheiten) oder *environment mapping* (Simulation spiegelnder Oberflächen) (Seite 63, rechte Spalte, vorletzter Absatz) sind, wie dem hier zuständigen Fachmann geläufig ist, ohne eine Reihe von Programmanweisungen nicht umzusetzen (Merkmal **M1.3**).

Weiterhin werden die Verarbeitungselemente *(PE)* und die *TASICs* des Textur-/Video-Subsystems über die *Image Generation Controller ASICs (IGCs)* durch die Programmanweisungen im SDRAM des *Geometry Processors (GP)* gesteuert und sind somit auch mit dem Speicher gekoppelt (Fig. 4; Seite 64, linke Spalte, Abschnitt „Rasterizer Control“, erster Absatz – Merkmal **M1.4**).

Jedes Verarbeitungselement *(PE)* ist zusammen mit dem Textur-/Video-Subsystem in der Lage, eine Mehrzahl von Texeln zu verarbeiten, die sich auf die Rohpixeldaten eines Pixels und damit auf ein Fragment beziehen (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values, based on surface normal, light sources, etc.“). Die Texel werden aus einem Texturspeicher in den lokalen Speicher der Verarbeitungselemente *(PE)* geladen und anschließend durch deren *ALUs* verarbeitet (Seite 63, rechte Spalte, erster Absatz; Fig. 6). Verarbeitungselemente *(PE)* und Textur-/Video-Subsystem werden dabei

vollständig durch die Programmanweisungen des *Geometry Processor Boards* gesteuert (Seite 64, linke Spalte, Abschnitt „Rasterizer Control“). Die Texturverarbeitung verläuft gemäß einem Programm, das nach dem Client-/Server-Prinzip gestaltet ist (Seite 66, linke Spalte, zweiter Absatz – Merkmal **M1.5**).

Die Verarbeitungselemente *(PE)* arbeiten auf jedem *Shader* parallel (Seite 59, rechte Spalte, letzter Absatz, siehe „Shading calculations can be performed for many pixels simultaneously; …), so dass auch die Texturverarbeitung bzw. die Verarbeitung von Texeln für alle Pixel und Subpixel parallel erfolgt (siehe oben).

Die parallele Texturverarbeitung findet für eine Mehrzahl von Texeln statt, die zwar zu verschiedenen Rohpixeldaten bzw. Fragmenten, aber zum selben Objekt gehören (Merkmal **M1.6**).

Die zur Texturverarbeitung nötigen Programmanweisungen des *Geometry Processor Boards* werden in das *SIMD Array* und das Textur-/Video-Subsystem geladen und dort auf den einzelnen Verarbeitungselementen *(PE)* und *TASICs* des *Rasterizer Boards* ausgeführt (Seite 64, linke Spalte, Abschnitt „Rasterizer Control“). Durch Verwendung der *OpenGL* API wird die Texturverarbeitung darüber hinaus auch programmierbar (Seite 65, rechte Spalte, Abschnitt „PixelFlow OpenGL“, zweiter Absatz) (Merkmal **M1.7**).

Demnach offenbart das *PixelFlow*-System der Druckschrift **NK12** sämtliche Merkmale des erteilten Patentanspruchs 1.

2. Dem Vorbringen der Beklagten, die Druckschrift **NK12** lehre kein Programm mit Anweisungen für eine anspruchsgemäße Texturierung (Texturmischen), geschweige denn, dass entsprechende Programmanweisungen aus einem Speicher von mehreren Texturprozessoren parallel ausgeführt werden, kann nicht gefolgt werden.

2.1 In diesem Zusammenhang führt die Beklagte aus, dass die Druckschrift **NK12** zwar an manchen Stellen von „user-programmable shading“ (Seite 57, linke Spalte, letzter Absatz, Zeile 4; Seite 57, rechte Spalte, vierter Absatz, Zeile 4) spreche, die Texturierung werde dabei aber klar eigenständig/separat und ohne den Zusatz „user-programmable“ erwähnt. Bereits daraus ergebe sich, dass die Programmierbarkeit gemäß der Druckschrift **NK12** allenfalls auf ein Shading im engeren Sinne bezogen ist, also die Anwendung von Licht und Schatten. Ein konkretes Beispiel einer „programmierbaren“ Texturierung/Texturmischung sei in der Druckschrift **NK12** nicht zu finden.

Der Einwand vermag nicht zu überzeugen.

So wird zwar die Eigenschaft „user-programmable“ bzw. „benutzerprogrammierbar“ in der Druckschrift **NK12** zusammen mit „Shading“ verwendet. Jedoch vermag dies allein noch keine abschließende Aussage darüber zu machen, ob und in welchem Maß die in *PixelFlow* realisierte Texturverarbeitung programmierbar war. Hingegen zeigt die Druckschrift **NK12** sehr wohl, dass die Texturverarbeitung in den einzelnen Verarbeitungselementen (PE) und TASICs durch Programmanweisungen des *Geometry Processor Boards* gesteuert wird (Seite 64, linke Spalte, Abschnitt „Rasterizer Control“); dabei wird auf die Programm-Routinen der OpenGL-Bibliothek zurückgegriffen, welche keinen „starren“ Funktionsumfang haben, sondern grundsätzlich angepasst und erweitert werden können. In diesem Umfang ist deshalb die Texturverarbeitung „programmierbar“ (vgl. S. 65 rechte Spalte, Abschnitt „PixelFlow OpenGL“, erster Absatz „… to demonstrate programmability in the hardware pipeline“, mit Verweis auf die **NK13**).

2.2 Nach Auffassung der Beklagten ist der Druckschrift **NK12** an keiner Stelle zu entnehmen, dass speziell die Texturmischung in irgendeiner Form über die herkömmlichen, im Streitpatent gewürdigten algebraischen Operationen hinausgeht. Dieses Dokument enthalte keinerlei Details, aus denen geschlossen werden könne, dass für die Texturierung mehr als lediglich Parameterwerte bzw.

Operanden zur Berechnung übergeben werden. Ferner könne sich die Nutzer-Programmierbarkeit der Druckschrift **NK12** nur auf *OpenGL*-Version 1.0 beziehen. In dieser *OpenGL*-Version sei aber eine Programmierbarkeit der Texturierung bzw. des Texturmischens überhaupt nicht möglich gewesen.

Diese Argumentation ist ebenfalls nicht überzeugend.

Vielmehr kann aufgrund der Verwendung der *OpenGL* API auch auf eine programmierbare Texturverarbeitung geschlossen werden (Seite 65, rechte Spalte, Abschnitt „PixelFlow OpenGL“, zweiter Absatz), bei der es sich aus Sicht des Senats allerdings nicht um eine *freie* sondern eher um eine *teilweise* bzw. *eingeschränkte* Programmierbarkeit handelt, die sich maßgeblich auf die in der API implementierten Funktionen stützt. Nichts Anderes trifft auf die in der Druckschrift **NK12** angesprochene *OpenGL*-Programmierschnittstelle der Version 1.0 zu. Dabei ist davon auszugehen, dass eine Programmierbarkeit i. S. d. Streitpatents nicht notwendigerweise voraussetzt, dass die beanspruchte Vorrichtung *frei* programmierbar ist oder unbeschränkt angepasst werden kann, so dass dieser Programmierbarkeit keine technischen Grenzen gesetzt sind. Die *OpenGL* API eröffnet dem Benutzer z. B. die Möglichkeit, Funktionen für *bump mapping* und *mip mapping* in ein verteiltes Anwendungsprogramm nach dem Client-/Server-Modell einzubinden und dieses auf „PixelFlow“ auszuführen (Seite 63, rechte Spalte, vorletzter Absatz). *Bump mapping* und *mip mapping* stellen bekanntlich prominente Beispiele für Spezialfälle einer Texturverarbeitung dar. Das Einbinden der entsprechenden Softwarekomponenten aus der *OpenGL*-Programmierschnittstelle in einen Programmcode zur Darstellung komplexer Grafik belegt, dass das in Druckschrift **NK12** beschriebene System „als solches“ programmierbar war – wenn auch nicht unmittelbar aus einem beliebigen Grafikprogramm heraus, sondern nur über den Rückgriff auf die *OpenGL*-Programmbibliothek; diese wiederum war aber grundsätzlich anpassbar und änderbar.

2.3 In Hinblick auf die für die arithmetisch-logischen Einheiten (ALUs) der Verarbeitungselemente *(PE)* bereitgestellten Mikroanweisungen argumentiert die Beklagte, dass diese vom Geometrie-Prozessor ausgegeben würden (Seite 64, linke Spalte, Abschnitt „Rasterizer Control“, erster Absatz) und daher gerade nicht die Anweisungen eines in einem Speicher gespeicherten Programms gemäß den Merkmalen **M1.2** und **M1.5** sein könnten.

Auch dieser Einwand hält einer genaueren Überprüfung nicht stand.

So sind die vom *Geometry Processor (GP)* an den SIMD-Prozessor und das Textur-Subsystem ausgegebenen Programmanweisungen im SDRAM gepuffert (Fig. 4; Seite 60, rechte Spalte, dritter Absatz „Memory … buffering commands for the rasterizer“). Wenigstens bei einem Teil dieser Programmanweisungen handelt es sich um solche für eine Texturverarbeitung, welche laut Druckschrift **NK12** auf der Verarbeitung einer Mehrzahl von Texeln beruht (Seite 59, rechte Spalte, Abschnitt „Deferred Shading“, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values …“).

Der Fachmann wird unschwer erkennen, dass es sich bei den gespeicherten Programmanweisungen um eine Abfolge von Instruktionen und somit um ein Programm bzw. den Teil eines Programms handelt. Dieses geht wiederum auf kleine Unterprogramme zurück, die auf einer *Host Workstation* hinterlegt sind, dort ablaufen und Funktionen der *OpenGL* API aufrufen (Seite 66, linke Spalte, zweiter Absatz). Die Funktionsaufrufe werden an die jeweiligen Shader verteilt und auf deren *Geometry Processors (GP)* ausgeführt, wobei der Code für die jeweiligen SIMD-Prozessoren generiert wird (Seite 66, linke Spalte, zweiter und dritter Absatz). Der anspruchsgemäße Speicher gemäß Merkmal **M1.2** ist damit in der Lehre der Druckschrift **NK12** in Gestalt des SDRAM verwirklicht.

3. Ausgehend von dem im Streitpatent genannten Stand der Technik argumentiert die Beklagte, die streitpatentgemäße Lösung diene vor allem dazu,

über einen vorgefertigten Satz mathematischer Funktionen für ein Texturmischen hinauszugehen. Sie beruft sich dabei insbesondere auf die Absätze [0008] und [0021] der Streitpatentschrift, aus denen hervorgehe, dass herkömmliche Texturmischeinheiten allenfalls eine kleine Teilmenge an Mischoperationen unterstützten und deshalb nicht an beliebige Texturmischungen verschiedener Anwendungen angepasst werden konnten (Spalte 2, Zeilen 52 bis 58; Spalte 6, Zeilen 34 bis 38). Die streitpatentgemäße Lösung ermögliche demgegenüber eine solche Anpassbarkeit von Texturmischeinheiten mittels des anspruchsgemäßen Computerprogramms. Die Lehre der Druckschrift **NK12** gehe aber nicht über die zum Prioritätstag des Streitpatents bekannten und im Streitpatent beschriebenen Lösungen hinaus. Nach Auffassung der Beklagten dürfe der erteilte Patentanspruch 1 nicht derart ausgelegt werden, dass dessen Gegenstand unter den genannten Stand der Technik falle.

Der Einwand der Beklagten geht bereits deswegen fehl, weil eine beliebige Anpassbarkeit von Texturmischeinheiten – was deren *freie Programmierbarkeit durch ein beliebiges Grafik-Programm* voraussetzen würde – in Merkmal **M1.7** auch in Verbindung mit den Merkmalen **M1.2** und **M1.3** ersichtlich nicht beansprucht wird. Durch die genannten Merkmale ist lediglich festgelegt, dass die Texturprozessoren ihre Berechnungen „in accordance with the program“ ausführen, wobei dieses (Teil-) Programm mehrere Anweisungen umfasst und in einem Speicher des Systems gespeichert ist. Ob das Programm aus einer Programmbibliothek stammt, die auf einer *Host Workstation* hinterlegt ist, oder ob das Programm von einem beliebigen (Grafik-) Programm übermittelt wurde, hat keinen Einfluss auf das anspruchsgemäße System.

In diesem Zusammenhang ist zu beachten, dass eine Auslegung unterhalb des Wortlauts eines Patentanspruchs generell nicht zulässig ist; dies gilt insbesondere, wenn (wie im Fall des Streitpatents) der Beschreibung eine Schutzbegrenzung auf bestimmte Ausführungsformen nicht zu entnehmen ist (vgl. BGH GRUR 2007, 309 – *Schussfädentransport).* Dabei erlaubt ein Ausführungsbeispiel regelmäßig keine

einschränkende Auslegung eines die Erfindung allgemein kennzeichnenden Patentanspruchs (vgl. BGH GRUR 2004, 1023 – *Bodenseitige Vereinzelungseinrichtung*).

Die aus dem englischen Verb „to implement“ betreffend Merkmal **M1.7** abgeleitete Programmierbarkeitseigenschaft der Texturprozessoren – d. h. die Möglichkeit, lauffähigen Programmcode zu erstellen und auf diese zu übertragen – schließt nach fachmännischem Verständnis nicht aus, dass der Programmcode aus einer – naturgemäß in ihrem Umfang beschränkten, jedoch erweiterbaren – externen Programmbibliothek stammen könnte. Dass der erteilte Patentanspruch 1 nach einem solchen Verständnis möglicherweise Ausführungsformen mit umfasst, die unter den im Streitpatent genannten Stand der Technik fallen, ist für eine sachgerechte Auslegung ohne Belang.

4. Unter Berufung auf die von ihr eingeführten Dokumente **G1** bis **G10** macht die Beklagte weiterhin geltend, dass die zum Prioritätstag des Streitpatents bekannten Grafikchipsätze keine Programmanweisungen hätten ausführen können, weil sie lediglich eine *fixed function pipeline* zur Verfügung gestellt hätten. *Fixed function* bedeute, dass ein solcher Grafikchipsatz gerade nicht – sei es voll oder nur teilweise – programmierbar war und dass jegliche Funktion der Pipeline fest, d. h. unveränderlich in der Hardware vorgegeben gewesen sei („fest verdrahtet“). Den Druckschriften **G2** und **G3** sei diesbezüglich insbesondere zu entnehmen, dass sich der Ansatz der *fixed function pipeline* in Hinblick auf die Programmierung graphischer Effekte als zu unflexibel erwiesen habe, da die durch die Programmierschnittstellen *OpenGL* und *DirectX* festgelegten Funktionen in Hardware implementiert gewesen seien (vgl. G2, Seite 3, zweiter Absatz, siehe „During years graphics card, to answer to the increasing demand for real-time graphics, has evolved from fixed functions pipelines, i. e. non programmable hardware pipelines to fully programmable highly parallel many-core architectures …“; Seite 5, erster Absatz, siehe „… during years, an increasing number of functions is executed on the graphics card, freeing the CPU, until the whole graphics pipeline

was implemented in hardware on the card (graphics pipelines with fixed functions).”; vgl. G3, Seite 3, linke Spalte, letzter Absatz, siehe „… the main problem with the fixed function model was the inflexibility of graphical effects. … the fixed function hardware could not take advantage of the new standards.”). Die ersten auf Pixel-/Fragment-Ebene voll programmierbaren Grafikkarten seien aber erst ab dem Jahr 2002 verfügbar gewesen (vgl. G3, Seite 3, rechte Spalte, letzter Absatz, siehe „One year later, in 2002, the first fully programmable graphics cards hit the market:
N… GeForce FX, ATI Radeon 9700.“).

Auch aus Dokument **G5** gehe hervor, dass bis in die späten 90er Jahre die führende Performance-Grafikhardware durch *Fixed Function Pipelines* gebildet worden sei, die zwar konfigurierbar, aber nicht programmierbar gewesen seien (vgl. G5, Seite 22, Abschnitt 2.1.1, siehe „From the early 1980s to the late 1990s, the leading performance graphics hardware was fixed-function pipelines that were configurable but not programmable“; Seite 26, zweiter Absatz, siehe „Although each generation introduced additional hardware resources and configurability to the pipeline stages, developers were growing more sophisticated and asking for more new features than could be reasonably offered as built-in fixed functions. The obvious next step was to make some of these graphics pipeline stages into programmable processors.”).

Ferner weist die Beklagte auf das Dokument **G7** hin, das sich mit der Programmierschnittstelle *DirectX 8* (veröffentlicht im Jahr 2000) beschäftigt, die eine Programmierung von Grafikprozessoren auf Pixelebene ermöglichte (vgl. G7, Seite 5, dritter Absatz). Im Gegensatz hierzu seien die früheren Versionen von *DirectX* (z. B. *DirectX 6.0*) *Fixed Function* gewesen, so dass Entwickler ihre Hardware nicht frei programmieren konnten und sich auf die statischen Funktionen in der API verlassen mussten (vgl. G7, Seite 1, zweiter Absatz). Nach Auffassung der Beklagten können die genannten Dokumente damit belegen, dass das *PixelFlow*-System der Druckschrift **NK12** aus dem Jahr 1997 allenfalls über eine *fixed function pipeline* verfügte, so dass dessen Texturverarbeitung weder voll noch teilweise programmiert, sondern lediglich konfiguriert werden konnte.

Zwar ist der Beklagten darin zuzustimmen, dass die häufigsten grundlegenden Funktionen des *OpenGL*-Standards, der auch dem *PixelFlow*-System zugrunde liegt, rein auf konfigurierbarer Ebene nutzbar sind, d. h. Werte für die Parameter der *OpenGL* API werden an feste Funktionen übergeben, die dann die Texturierung steuern. Für Funktionalitäten, für die allerdings eine Konfiguration nicht in Frage kommt, kann eine Anwendung zusätzlich durch die Erweiterung von individuellem nativen Programmcode angereichert werden. In *OpenGL* sind hierfür sogenannte *Extensions* für neue noch nicht vom Standard unterstützte Funktionen vorgesehen. Um eine Programmierbarkeit zu unterstützen, macht man sich auch im *PixelFlow*-System der Druckschrift **NK12** die Eigenschaften solcher *Extensions* zunutze. So können unter *PixelFlow OpenGL* die von einem Programmierer erstellten Prozeduren für *Shading* und Beleuchtung mittels *OpenGL Extensions* in den Programmcode eines Shaders eingebunden und auf die zuständigen Prozessoren geladen werden. Insbesondere werden im *PixelFlow*-System *Extensions* angewandt, um benutzerdefinierte Parameter für ein *Shading* bereitzustellen, die von Primitiv zu Primitiv geändert werden können. Weitere *Extensions* betreffen u. a. die Festlegung von Texturkoordinaten, die neben anderen Textur- und Umgebungsparametern die Anwendung von Texturen steuern und infolgedessen auch das Texturmischen beeinflussen (vgl. **NK12**, Seite 65, Abschnitt 4.1, zweiter Absatz, siehe „Support for programmability includes library routines for loading user-coded shading and lighting functions. … We provide other OpenGL extensions to set the values of user-defined shading parameters that can change on a primitive by primitive basis. Examples include not only standard parameters, such as surface normal and texture coordinates, but also arbitrary others, such as noise frequency, etc.”).

Alles in allem muss dieses als eine Programmierbarkeit der Texturverarbeitung in *PixelFlow* verstanden werden. Allerdings handelt es sich hierbei nicht um eine *freie,* sondern eher um eine *teilweise* bzw. *eingeschränkte* Programmierbarkeit, die sich maßgeblich auf die in der *OpenGL* API implementierten Funktionen stützt. Der

Patentanspruch 1 des Streitpatents schließt jedoch einen Zugriff auf Anwendungsprogrammierschnittstellen als Quelle für den Programmcode, welcher jeweils die Texturprozessoren steuert, in keiner Weise aus.

5. Damit offenbart die Druckschrift **NK12** alle Merkmale des Gegenstandes nach dem erteilten Patentanspruch 1. Dem Gegenstand des Patentanspruchs 1 gemäß Hauptantrag fehlt es daher an der für die Patentfähigkeit erforderlichen Neuheit.

6. Der auf „ein Verfahren zur Strukturverarbeitung für eine Grafikabbildung auf einer Anzeige“ gerichtete, nebengeordnete Patentanspruch 8 ist nicht günstiger als Patentanspruch 1 zu beurteilen, da er inhaltlich nicht über diesen hinausgeht und somit nichts enthält, was eine Patentfähigkeit rechtfertigen würde.

7. Weder der unabhängige Systemanspruch 1 noch der unabhängige Verfahrensanspruch 8 des Streitpatents haben daher Bestand. In seiner erteilten Fassung ist das Streitpatent, dessen abhängige Unteransprüche die Beklagte nicht gesondert verteidigt hat, insgesamt für nichtig zu erklären.

IV.

Das Streitpatent ist in keiner der Fassungen der zur Akte gereichten Hilfsanträge I bis VII, VIIa und VIII patentfähig. Im Hinblick darauf kann dahin gestellt bleiben, ob die Fassungen jeweils zulässig sind.

1. Dem **Hilfsantrag I** kann nicht stattgegeben werden, weil der Gegenstand seines Patentanspruchs 1 nicht neu ist.

1.1 Gemäß Hilfsantrag I wurde Merkmal **M1** des erteilten Patentanspruchs 1 in Merkmal **M1´** mit der Maßgabe abgeändert, dass sich das beanspruchte System

auf eine *Bereitstellung von programmierbarer Texturverarbeitung* bezieht. Weiterhin wurde Merkmal **M1.8** eingeführt, wonach die *Verarbeitung eines Teils der Texel für ein Fragment entsprechend dem Programm ein Texturmischen („texture blending“)* beinhaltet.

Der unabhängige Patentanspruch 8 wurde entsprechend angepasst.

Entsprechend dem Verständnis des Streitpatents umfasst ein *Texturmischen* bzw. *texture blending* eine Operation, bei der eine Textur mit einer Farbe oder einer anderen Textur beispielsweise durch Interpolation vermischt bzw. kombiniert wird (vgl. Streitpatentschrift, Abs. [0005], [0006] und [0020]).

1.2 Die Merkmale **M1´** und **M1.8** sind aus Druckschrift **NK12** bekannt.

Bereits zum Hauptantrag ist ausgeführt, dass das *PixelFlow*-System der Druckschrift **NK12** eine – wenn auch eingeschränkte – Programmierbarkeit der Texturverarbeitung ermöglichte (Merkmal **M1´**).

Weiterhin ergibt sich aus den in der Druckschrift **NK12** genannten Techniken der Bildsynthese, z. B. *bump mapping* oder *mip mapping* (Seite 63, rechte Spalte, vorletzter Absatz), dass das Texturmischen eine von *PixelFlow* unterstützte Art der Texturverarbeitung darstellte. So entnimmt der Fachmann der Druckschrift **NK12**, dass die genannten Techniken Operationen vorsehen, bei denen Texturen mit den Farbwerten von Fragmenten gemischt werden, um endgültige Farbwerte zu bekommen (vgl. **NK12**, Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz, siehe „The shaders look up texture values for the pixels and compute final pixel color values, based on surface normal, light sources, etc.“ – Merkmal **M1.8**).

1.3 Die Beklagte argumentiert, die Druckschrift **NK12** gebe keinen Hinweis, wie ein Texturmischen durchgeführt, geschweige denn wie ein solches programmiert werden könne. Das unter Abschnitt 3.3 der Druckschrift **NK12** beschriebene *mip*

mapping betreffe allenfalls *texture lookups*, zeige aber kein programmierbares Texturmischen.

Zwar ist der Beklagten darin zuzustimmen, dass *mip mapping* für sich gesehen kein programmierbares Texturmischen umfasst. Um die Darstellungsqualität von Bildern bzw. Texturen zu verbessern, wenn diese kleiner als in der ursprünglichen Größe dargestellt werden sollen, sieht *mip mapping* nach fachmännischem Verständnis vielmehr vor, verschiedene Versionen eines originalen Bildes, sogenannte *MipMap Levels* in Form von Texturkarten im Speicher abzulegen, welche in der Regel jeweils um einen Faktor 2 verkleinert sind. Durch anschließende *texture lookups* werden dann ausgehend von Texturkoordinaten aus den *MipMap*-Texturkarten Farbwerte ermittelt, die einer weiteren Texturverarbeitung bereitgestellt werden. Dass diese Farbwerte als Texturwerte einem Texturmischen zugeführt werden, ergibt sich bereits aus der oben genannten Textstelle auf Seite 59 unter Abschnitt 2.5 der Druckschrift **NK12**. Dort wird beschrieben, dass spezifische Pixelattribute, z. B. Farben von Oberflächen, von den Verarbeitungselementen *(PE)* der *Shader* zusammen mit Texturwerten zu endgültigen Pixelwerten bzw. -farben verarbeitet – also gemischt – werden (siehe „PixelFlow rasterizers do not compute pixel colors directly; instead, they compute geometric and intrinsic pixel attributes, such as surface-normal vectors and surface color; these attributes, not pixel colors, are composited. … The shaders look up texture values for the pixels and compute final pixel color values, based on surface normal, light sources etc.”). Im Übrigen ist dem Fachmann geläufig, dass die von *texture lookups* unterstützten Funktionen, wie *bump mapping* (Simulation von Oberflächenunebenheiten) oder *environment mapping* (Simulation spiegelnder Oberflächen) (vgl. **NK12**, Seite 63, rechte Spalte, vorletzter Absatz) gewöhnlich mit einem Texturmischen verknüpft sind.

Unter Berücksichtigung der Ausführungen zum Hauptantrag unterstützt *PixelFlow* demnach nicht nur eine programmierbare Texturverarbeitung (vgl. Merkmal **M1´**), sondern auch ein Texturmischen (vgl. Merkmal **M1.8**). Dass die Druckschrift **NK12** nicht beschreibt, wie ein Texturmischen durchgeführt oder gar programmiert werden

kann, ist in diesem Zusammenhang ohne Belang, da mit Merkmal **M1.8** allenfalls beansprucht wird, dass ein Texturmischen überhaupt stattfinden soll.

1.4 Mit Rücksicht auf die Ausführungen zum Hauptantrag ist die Lehre des Patentanspruchs 1 gemäß Hilfsantrag I nicht neu und daher nicht patentfähig. Mit dem Patentanspruch 1 fällt der gesamte Hilfsantrag.

Beantragt der Patentinhaber, das Patent in beschränktem Umfang mit einem bestimmten Anspruchssatz oder bestimmten Anspruchssätzen aufrechtzuerhalten, rechtfertigt es grundsätzlich die Ablehnung des gesamten Antrags, wenn sich auch nur der Gegenstand eines Patentanspruchs aus dem vom Patentinhaber verteidigten Anspruchssatz als nicht patentfähig erweist *(vgl.* BGH GRUR 2007, 862 – *Informationsübermittlungsverfahren II)*. Allerdings ist das Gericht gehalten, aufzuklären, in welchem Verhältnis die Hilfsanträge zu einem nicht ausdrücklich formulierten Petitum stehen sollen, einem formal vorrangigen Antrag nur teilweise zu entsprechen (BGH GRUR 2017, 57 - *Datengenerator)*.

Im vorliegenden Fall hat die Beklagte nur die Patentansprüche 1 und 8 des Hilfsantrags VII, eingereicht mit Schriftsatz vom 21. September 2020, ausdrücklich einzeln vereidigt. Im Übrigen hat sie sich der Erklärung des Senats in der mündlichen Verhandlung angeschlossen, er verstehe die von der Beklagten vorgelegten Hilfsanträge grundsätzlich im Sinne geschlossener Anspruchssätze, die sie jeweils in ihrer Gesamtheit beanspruche. Vorbehaltlich der für Hilfsantrag VII beantragten Ausnahme, also für alle übrigen Hilfsanträge einschließlich des Hilfsantrags I, schließt dies eine separate Betrachtung einzelner Patentansprüche aus, wenn sich ein Patentanspruch des betroffenen Anspruchssatzes, wie hier, als nicht patentfähig erweist.

2. Der **Hilfsantrag II** kann nicht günstiger beurteilt werden, weil das zum Patentanspruch 1 hinzugekommene Merkmal ausgehend von Druckschrift **NK12** nahegelegt ist.

2.1 Der Hilfsantrag II beruht auf dem Hilfsantrag I, wobei Merkmal **M1.9** bzw. **M8.3.5** in Patentanspruch 1 bzw. 8 aufgenommen worden ist. Demnach sollen die Texturprozessoren in der Lage sein, sich auf unterschiedliche Eingaben hin an unterschiedliche Mischoperationen anzupassen.

Ausgehend von Seite 12, Zeilen 26 bis 29 der Anlage **NK2** und den Absätzen [0035] und [0036] der Streitpatentschrift ist Merkmal **M1.9** bzw. **M8.3.5** so zu verstehen, dass aufgrund des anspruchsgemäßen Programms zur Texturverarbeitung die Texturprozessoren in der Lage sein sollen, in Abhängigkeit von unterschiedlichen Eingaben unterschiedliche Mischoperationen auszuführen.

2.2 Die Lehre des Patentanspruchs 1 gemäß Hilfsantrag II hat ausgehend von der Druckschrift **NK12** nahegelegen.

So gibt die *OpenGL*-Programmieranleitung **NK55** der Version 1.2.1 vom 14. Oktober 1998 Hinweise, wie ein „Blending“ (Seiten 146ff) programmiert werden kann oder welche Funktionen für ein Multitexturing angewandt werden können (Seiten 240ff).

Insbesondere werden in Druckschrift **NK55** auf den Seiten 136 und 137 diejenigen Funktionen beschrieben, die in Abhängigkeit von der Anzahl der Farbanteile der Textur und dem Texturmodus *(REPLACE, MODULATE, DECAL, BLEND)* verschiedene Mischoperationen ausführen können, in denen Fragmentfarbe und Texturfarbe kombiniert werden. Zum besseren Verständnis sei an dieser Stelle außerdem auf die Druckschrift **NK57** hingewiesen, die auf Seite 204 die unter der *OpenGL*-Version 1.1 verfügbaren Funktionen zur Texturmischung näher erläutert.

Die Druckschrift **NK55** offenbart damit zumindest die zum Prioritätstag des Streitpatents unter *OpenGL* verfügbaren Funktionen zur Texturverarbeitung, die ganz konkret verschiedene Mischoperationen in Abhängigkeit von verschiedenen Eingaben realisieren. Da der Fachmann stets bestrebt ist, in einem System zur

Bildsynthese die aktuellsten Softwarebibliotheken zu verwenden, lag es für ihn auf der Hand, auch im *OpenGL*-basierten *PixelFlow*-System der Druckschrift **NK12** die neueste Version der *OpenGL* API mit der größtmöglichen Funktionalität anzuwenden. Die Verarbeitungselemente eines solchen angepassten *PixelFlow*-Systems, die als Texturprozessoren fungieren, sind dann in der Lage, verschiedene Mischoperationen an Fragmenten und Texturen durchzuführen, und zwar in Abhängigkeit von verschiedenen Eingaben, wie etwa Farbanteilen der Textur und Texturmodi (Merkmal **M1.9**).

2.3 Die Beklagte macht geltend, dem *PixelFlow*-System der Druckschrift **NK12** liege allenfalls *OpenGL*-Version 1.0 zugrunde, das keine benutzerprogrammierbare Texturverarbeitung erlaube. Letzteres gelte ebenso für die *OpenGL*-Version 1.2.1 aus Druckschrift **NK55**.

Der Einwand der Beklagten geht bereits deswegen fehl, weil *PixelFlow OpenGL* um *Extensions* erweitert worden ist, die eine Programmierbarkeit dadurch unterstützten, dass benutzercodierte Funktionen für Shading- und Beleuchtungsmodelle in Anwendungen eingebunden werden konnten. Die *Extensions* ermöglichen insbesondere, Werte für benutzerdefinierte Parameter, z. B. Texturkoordinaten festzulegen, die sich von Primitiv zu Primitiv ändern können (vgl. **NK12**, Seite 65, rechte Spalte, Abschnitt 4.1, zweiter Absatz), was sich wiederum auf die Texturverarbeitung und das Texturmischen auswirkt. Nichts Anderes gilt nach Auffassung des Senats für ein *PixelFlow*-System, das sich auf *OpenGL*-Version 1.2.1 mit *Extensions* stützt. Dass eine Verwendung von *OpenGL*-Version 1.2.1 unter *PixelFlow* zum Prioritätstag des Streitpatents grundsätzlich möglich gewesen sein muss, ergibt sich bereits aus der Tatsache, dass nahezu alle Komponenten von *PixelFlow* programmierbar waren (vgl. **NK12**, Seite 67, rechte Spalte, Abschnitt 6, zweiter Absatz, siehe „Virtually all of the components of PixelFlow are programmable: …“) und *PixelFlow* auch auf Rechenanlagen mit vielen Ressourcen implementiert werden konnte (vgl. **NK12**, Seite 67, rechte Spalte,

Abschnitt 6, dritter Absatz, siehe „Coupled to a parallel supercomputer, it can serve as a visualization subsystem for immediate-mode rendering.“).

2.4 Die Beklagte argumentiert, der Spezifikation **NK55** sei allenfalls ein begrenzter mathematischer Funktionensatz für ein Texturmischen zu entnehmen (vgl. **NK55**, Seite 137), der keine Programmierbarkeit der Texturverarbeitung gestatte.

Zwar ist die Anzahl der unter der *OpenGL*-Version 1.2.1 für ein Texturmischen zur Verfügung stehenden mathematischen Texturfunktionen ersichtlich begrenzt (vgl. **NK55**, Seite 137, Tabelle 3.19, siehe „Decal and blend texture functions“). Die Programmierbarkeit der Texturverarbeitung beruht aber – entgegen der Auffassung der Beklagten – zum einen auf der Möglichkeit, *OpenGL Extensions* auch unter *OpenGL*-Version 1.2.1 bereitzustellen, die entsprechend der Lehre der Druckschrift **NK12** benutzerindividuellen Programmcode einbinden können, der die Texturverarbeitung betrifft (vgl. **NK12**, Seite 65, Abschnitt 4.1). Zum anderen resultiert die Programmierbarkeit der Texturverarbeitung schon aus der Tatsache, eine Mehrzahl von *OpenGL*-Aufrufen in eine Anwendung einbinden zu können, die die Definition von Texturen (vgl. **NK55**, Seite 112ff, Abschnitt 3.8.1), die Festlegung von Texturparametern (vgl. **NK55**, Seite 123ff, Abschnitt 3.8.3) und die Anwendung von Texturen steuern (vgl. **NK55**, Seite 135ff, Abschnitt 3.8.9), wodurch eine Auswahl aus den oben genannten Funktionen für das Texturmischen getroffen wird. Dabei gehört es zum fachmännischen Wissen und Können, im *PixelFlow*-System der Druckschrift **NK12** dafür Sorge zu tragen, dass die bei jedem *OpenGL*-Aufruf anfallende Arbeitslast zwischen *Host Workstation* und modularem Grafiksystem unter dem Gesichtspunkt der Performanz in geeigneter Weise aufgeteilt wird.

2.5 Damit ist auch der Patentanspruch 1 in der Fassung des Hilfsantrags II nicht patentfähig. Mit dem Patentanspruch 1 fällt der gesamte Hilfsantrag.

3. Hilfsantrag III hat keinen Erfolg, weil der Gegenstand seines Patentanspruchs 1 durch die Druckschrift **NK12** zumindest nahegelegt ist.

3.1 Der Hilfsantrag III basiert auf dem Hilfsantrag I, wobei das zusätzliche Merkmal **M1.9´** bzw. **M8.3.5´** in die unabhängigen Patentansprüche aufgenommen worden ist. Demnach sollen die Texturprozessoren derart angepasst werden können, dass sie eine Vielfalt von Texturmischungen durchführen können.

3.2 Aus der Druckschrift **NK12** folgt, dass das dort offenbarte *PixelFlow*-System aufgrund der *OpenGL* API programmierbar war. Außerdem war das bekannte System nicht auf die Anwendung einer einzigen Textur auf Fragmente eingeschränkt (vgl. **NK12**, Seite 63, rechte Spalte, Abschnitt 3.3, erster und zweiter Absatz, siehe „Mip-map texture lookups“; Seite 63, rechte Spalte, Abschnitt 3.3, vorletzter Absatz, siehe „bump mapping“ u. a.). Weiterhin war ein auf der *OpenGL*-Version 1.2.1 basierendes *PixelFlow*-System dank der im *OpenGL*-Standard bereitgestellten Texturfunktionen (s. o.) derart programmierbar bzw. anpassbar, dass nicht nur eine, sondern eine Vielfalt an Texturmischungen ausgeführt werden konnte (Merkmal **M1.9´**).

3.3 Die Beklagte führt aus, dass aus den Druckschriften **NK12** und **NK55** keine Vielzahl von Texturmischungen i. S. d. Merkmals **M1.9´** hervorgehe.

Der Einwand überzeugt nicht.

So offenbart die Druckschrift **NK12**, dass in *PixelFlow* grundsätzlich eine Mehrzahl von Texturen mit *mip mapping* angewendet werden konnte (vgl. **NK12**, Seite 63, Abschnitt 3.3, zweiter Absatz, siehe „Prefiltered (Mip-map) texture maps can be interleaved across the banks so that eight texels required for one pixel are stored in the eight separate banks.“), was bei vorgefilterten *MipMap Levels* je nach Textur und Auflösung zu unterschiedlichen Texturmischungen führte. Außerdem geht aus der Spezifikation **NK55** hervor, dass die mathematischen Mischfunktionen der

Tabellen 3.18 und 3.19 auf den Seiten 136 und 137 für ein und dieselbe auf ein Fragment angewandte Textur unterschiedliche Texturmischungen hervorbringen. In diesem Sinne hat ein auf *OpenGL*-Version 1.2.1 basierendes *PixelFlow*-System die Eigenschaft, eine Vielfalt von Texturmischungen durchzuführen.

3.4 Mit Rücksicht auf die Ausführungen zu Hilfsantrag I ist der Gegenstand des Patentanspruchs 1 gemäß Hilfsantrag III nicht patentfähig. Mit dem Patentanspruch 1 fällt der gesamte Hilfsantrag. Der Hilfsantrag III ist nach allem nicht günstiger zu beurteilen als der Hilfsantrag I.

4. Hilfsantrag IV hat keinen Erfolg, weil der Gegenstand seines Patentanspruchs 1 nicht neu ist.

4.1 Der Hilfsantrag IV beruht auf Hilfsantrag I, wobei die Merkmale **M1.2** und **M1.5** durch die Merkmale **M1.2´** und **M1.5´** ersetzt worden sind. Laut Merkmal **M1.2´** soll der in einem Speicher hinterlegte Teil eines Programms von der Mehrzahl der Texturprozessoren zur Texturverarbeitung genutzt werden. Gemäß Merkmal **M1.5´** ist jeder der Texturprozessoren dazu vorgesehen, wenigstens einen Teil des Programms zu importieren und einen Teil der Mehrzahl der Texturabschnitte bzw. Texel für ein Fragment gemäß dem Programm zu verarbeiten.

Der Fachmann wird das Teilmerkmal *„importing at least a portion of the program“* so verstehen, dass wenigstens ein Teil eines Programms von den Texturprozessoren eingelesen bzw. geladen wird, um dann eine Texturverarbeitung auszuführen. Eine solche Auslegung findet ihre Stütze in Absatz [0043] (siehe „Using instructions stored in the program cache 160, a texture processor 190-1 through 190-k processes a fragment.“) und Absatz [0033] der Streitpatentschrift (siehe „… the program used by a texture processor 154-1 through 154-k may be imported from the memory 110.“).

4.2 Diese Maßnahmen können jedoch eine Patentfähigkeit nicht begründen.

So offenbart Druckschrift **NK12** in Figur 5, dass ein EMC-Chip und damit auch die jeweiligen Verarbeitungselemente *(PE)* (die ja als Texturprozessoren fungieren) Programmanweisungen („instructions“) erhalten. Die jeweils zu verarbeitende Textur wird von einem Verarbeitungselement *(PE)* über den „Input Buffer“ geladen (vgl. **NK12**, Fig. 5, 6; Seite 63, rechte Spalte, erster und zweiter Absatz).

Weiterhin geht – wie bereits zum Hauptantrag ausgeführt - aus der Druckschrift **NK12** hervor, dass die vom *Geometry Processor (GP)* erzeugten und im SDRAM zwischengespeicherten Programmanweisungen über die Controller-*IGCs* an die Verarbeitungselemente *(PE)* ausgegeben werden. Die Anweisungen werden von diesen geladen bzw. importiert und demnach auch zur Texturverarbeitung verwendet (vgl. **NK12**, Seite 64, linke Spalte, Abschnitt 3.4, erster und zweiter Absatz, siehe „It provides the SIMD micro-instruction to the PEs.“).

Mit Rücksicht auf die Ausführungen zum Hauptantrag bzw. Hilfsantrag I sind die Merkmale **M1.2´** und **M1.5´** damit aus der Druckschrift **NK12** bekannt.

4.3 Die Beklagte führt aus, in der Lehre der Druckschrift **NK12** werde keine Mehrzahl von Programmanweisungen an die Verarbeitungselemente *PEs* übergeben. Dementsprechend werde dort auch kein Programmteil geladen und ausgeführt.
Der Einwand greift nicht durch. So geht aus Druckschrift **NK12** unmittelbar hervor, dass die vom *Geometry Processor GP* ausgegebenen Mikrobefehle mittels eines Controller-*IGC* an die *SIMD*-Prozessoren bzw. deren Verarbeitungselemente *(PE)* weitergeleitet werden. Ein Mikrobefehl beinhaltet dabei u. a. Steuerfelder für die arithmetisch-logische Einheit ALU eines Verarbeitungselements *(PE)* sowie die Adresse von dessen lokalem Speicher (vgl. **NK12**, Seite 64, linke Spalte, Abschnitt 3.4, zweiter Absatz). Nach fachmännischem Verständnis stellt der Mikrobefehlsstrom eine Abfolge von Programmanweisungen dar, die zusammen

einen Programmteil bilden und die auf den Verarbeitungselementen *(PE)* der *SIMD*-Prozessoren zur Ausführung gebracht werden.

4.4 Da sich auch die Merkmale **M1´** und **M1.8** aus der Druckschrift **NK12** ergeben (siehe Ausführungen zu Hilfsantrag I), fehlt es dem Gegenstand gemäß Hilfsantrag IV an der für die Patentfähigkeit erforderlichen Neuheit, weswegen Hilfsantrag IV nicht günstiger als der Hauptantrag bzw. Hilfsantrag I beurteilt werden kann.

5. **Hilfsantrag V** hat keinen Erfolg, da die Lehre seines Patentanspruchs 1 mit Rücksicht auf den der Druckschrift **NK12** entnehmbaren Stand der Technik nicht neu ist.

5.1 Der Hilfsantrag V beruht auf dem Hilfsantrag IV, wobei in Patentanspruch 1 Merkmal **M1.5´** gegen Merkmal **M1.5** ausgetauscht und der Patentanspruch 1 um Merkmal **M1.7a** ergänzt wurde. Patentanspruch 8 wurde dementsprechend angepasst.

Das ergänzte Merkmal **M1.7a** sieht vor, Fragmente mitsamt den zu verarbeitenden Texturabschnitten und den Instruktionen, die auf einem Texturprozessor ausgeführt werden sollen, für einen oder mehrere Texturprozessoren bereitzustellen.

5.2 Das neu hinzugekommene Merkmal **M1.7a** kann eine Patentfähigkeit des Patentanspruchs 1 gemäß Hilfsantrag V nicht begründen, da es aus Druckschrift **NK12** bekannt ist.

Druckschrift **NK12** zeigt, dass den Verarbeitungselementen *(PE)*, die die Aufgaben von Texturprozessoren übernehmen, Programmanweisungen und zu verarbeitende Texturen bereitgestellt werden (vgl. **NK12**, Seite 64, linke Spalte, Abschnitt 3.4, erster und zweiter Absatz, siehe „It provides the SIMD micro-instruction to the PEs.“; Fig. 6, siehe „Data in from TASIC“; Seite 63, rechte Spalte, erster und zweiter Absatz). Die den Verarbeitungselementen *(PE)* zugeführten Rohpixeldaten, die die

anspruchsgemäßen Fragmente repräsentieren (vgl. **NK12**, Seite 58, linke Spalte, vorletzter Absatz, siehe „raw pixel attributes“), setzen sich aus geometrischen und spezifischen Pixel-Attributen zusammen (Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz, siehe „… instead, they compute geometric and intrinsic pixel attributes, such as surface-normal vectors and surface color …“). Die Bereitstellung von Fragmenten, Texturabschnitten und Programmanweisungen für Texturprozessoren i. S. d. Merkmals **M1.7a** geht damit aus der Druckschrift **NK12** hervor.

5.3 Die Beklagte argumentiert, in Abschnitt 3.4 der Druckschrift **NK12** sei allenfalls ein Strom aus Mikroinstruktionen offenbart, jedoch keinerlei Fragmente, die zudem laut Merkmal **M1.7a** Texturabschnitte beinhalten müssten.

Zwar ist der Beklagten darin zuzustimmen, dass sich Abschnitt 3.4 in erster Linie mit der Übergabe von Mikrobefehlen an den *SIMD*-Prozessor und das Textur-/Video-Subsystem beschäftigt und dort die Verarbeitung von Fragmenten nicht direkt angesprochen wird. Allerdings lehrt Druckschrift **NK12**, dass Rohpixeldaten, welche geometrische und spezifische Pixel-Attribute umfassen und welche der Fachmann als Fragmente i. S. d. Streitpatents verstehen wird, in die Shader und damit auch in die Verarbeitungselemente *(PE)* geladen werden, die letztendlich als Texturprozessoren fungieren (vgl. **NK12**, Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz). Dabei ist zu beachten, dass Merkmal **M1.7a** entsprechend der Präposition „including“ (zu Deutsch „mit“, „mitsamt“, „einschließlich“) nicht notwendigerweise verlangt, dass die Fragmente selbst Texturabschnitte umfassen müssen. Vielmehr ist Merkmal **M1.7a** so zu verstehen, dass Fragmente zusammen mit Texturabschnitten an Texturprozessoren übergeben werden, wobei offenbleibt, ob die Texturabschnitte in den Fragmenten enthalten sind oder nicht.

5.4 Mit Blick auf die Ausführungen zum Hilfsantrag IV ist der Gegenstand des Patentanspruchs 1 gemäß Hilfsantrag V nicht neu, so dass das Streitpatent in seiner Fassung nach Hilfsantrag V, der ebenfalls als geschlossener Anspruchssatz zu verstehen ist, keinen Bestand hat.

6. Gleiches gilt für das Streitpatent in seiner Fassung nach **Hilfsantrag VI**, weil das gegenüber dem Hilfsantrag V einschränkende Merkmal aus der Druckschrift **NK12** bekannt ist.

6.1 Der Hilfsantrag VI beruht auf dem Hilfsantrag V, wobei Merkmal **M1.10** in Patentanspruch 1 aufgenommen worden ist. Demnach soll jeder Texturprozessor für die Fragmente, die er erhält, das gesamte Texturmischen durchführen. Der nebengeordnete Patentanspruch 8 wurde entsprechend angepasst.

6.2 Das neu hinzugekommene Merkmal **M1.10** geht aus Druckschrift **NK12** hervor.

So war das *PixelFlow*-System in der Lage, alle Texturmischungen für ein Fragment in einem Texturprozessor durchzuführen. Dies wird in Druckschrift **NK12** am Beispiel eines *mip mappings* verdeutlicht, bei dem ein Verarbeitungselement *(PE)* acht Texturwerte *(texel)* anwendet (vgl. **NK12**, Seite 63, rechte Spalte, Abschnitt 3.3, zweiter und dritter Absatz). Dabei werden alle acht Texturwerte in den Puffer des Verarbeitungselements *(PE)* geladen, vom Verarbeitungselement *(PE)* verarbeitet und gemäß Abschnitt IV. 1.2 (s. o.) mit Farbwerten gemischt.

Weiterhin ist Druckschrift **NK12** nicht nur die Zuordnung der Verarbeitungselemente *(PE)* zu den einzelnen Pixeln (vgl. **NK12**, Seite 62, linke Spalte, fünfter Absatz, siehe „The PEs are grouped into sets of 1, 4 or 8, each group corresponding to a pixel.“) zu entnehmen, sondern auch der Hinweis, dass jedes Subpixel (und somit bei 1 *PE* pro Pixel auch jedes Pixel) unabhängig (und vollständig) zusammen mit den Shading-Informationen verarbeitet werden kann (vgl. **NK12**, Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz, siehe „For ultimate quality rendering, every subpixel sample can be shaded independently.“). Hieraus folgt, dass ein einzelnes Verarbeitungselement *(PE)* dazu ausgelegt ist, für ein einzelnes Pixel (d. h. für ein zugeführtes Fragment) das gesamte *Shading* zu übernehmen, was aber gleichzeitig bedeutet, dass dort auch die gesamte Texturverarbeitung für das Pixel ablaufen

kann. Dabei ist zu beachten, dass Merkmal **M1.10** bereits dann erfüllt ist, wenn die gesamte Texturverarbeitung für eine einzige Textur und ein einziges Pixel auf einem der Verarbeitungselemente *(PE)* erfolgt.

Merkmal **M1.10** ist somit aus Druckschrift **NK12** bekannt.

6.3 Die Beklagte führt aus, Druckschrift **NK12** offenbare nicht, dass sämtliche Operationen einer Texturverarbeitung bzw. eines Texturmischens an einem Fragment auf ein und demselben Verarbeitungselement *(PE)* stattfinden.

Dem Einwand kann nicht zugestimmt werden; denn der Fachmann entnimmt Abschnitt 2.5 der Druckschrift **NK12** nicht nur, dass ein *Shading* für sämtliche Subpixel bzw. Pixel ausgeführt werden kann, sondern dass dieses für jedes Subpixel bzw. Pixel unabhängig stattfindet (vgl. **NK12**, Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz). Demnach ist aber auch die ganze Texturverarbeitung bzw. das ganze Texturmischen eines Subpixels bzw. Pixels auf dem ihm zugeordneten Verarbeitungselement *(PE)* unabhängig vom *Shading* auf den anderen Verarbeitungselementen *(PE)*, d. h. eine Verteilung der Texturverarbeitung bzw. des Texturmischens an einem einzigen Subpixel bzw. Pixel auf mehrere Verarbeitungselemente *(PE)* lässt sich aus der genannten Textstelle gerade nicht ableiten. Vielmehr ist in der Lehre der Druckschrift **NK12** davon auszugehen, dass das gesamte *Shading* und somit auch die gesamte Texturverarbeitung an einem Subpixel bzw. Pixel auf einem Verarbeitungselement *(PE)* ausgeführt wird.

6.4 Angesichts der Ausführungen zum Hilfsantrag V erweist sich auch der Gegenstand des Patentanspruchs 1 gemäß Hilfsantrag VI als nicht neu. Mit Patentanspruch 1 fällt der gesamte Hilfsantrag.

7. **Hilfsantrag VII** kann nicht günstiger beurteilt werden, da die jeweiligen Lehren seiner unabhängigen Patentansprüche 1 und 8, die von der Beklagten

ausdrücklich auch einzeln verteidigt werden, nicht auf erfinderischer Tätigkeit beruhen.

7.1 Der Hilfsantrag VII beruht auf dem Hilfsantrag IV, wobei in Patentanspruch 1 die Merkmale **M1.11** und **M1.12** hinzugekommen sind. Der nebengeordnete Patentanspruch 8 wurde mit den Merkmalen **M8.3.7** und **M8.3.8** entsprechend angepasst. Demgemäß soll über die Lehre des Hilfsantrags IV hinaus jedes Fragment eine Programm ID umfassen, wobei der zur Texturverarbeitung notwendige „Teil des Programms“ anhand dieser Programm ID bestimmt und geholt bzw. importiert wird.

7.2 Der Beklagten ist zuzustimmen, dass der Gegenstand des Patentanspruchs 1 nach Hilfsantrag VII nicht mehr ohne weiteres als durch die Lehre der Druckschrift **NK12** neuheitsschädlich getroffen oder allein durch diese nahegelegt bezeichnet werden kann. Dennoch ist die beanspruchte Lehre nicht patentfähig, weil sie ausgehend von Druckschrift **NK12** in Verbindung mit dem aus der Druckschrift **NK13** entnehmbaren Stand der Technik nahegelegt ist.

7.2.1 Die Druckschrift **NK12** zeigt alle Merkmale des Patentanspruchs 1 nach Hilfsantrag VII mit dem einzigen Unterschied, dass die Lehre der Druckschrift **NK12** nicht ausdrücklich vorsieht, dass das Fragment mitsamt einer Programm ID an die Texturprozessoren übermittelt wird (Merkmal **M1.11**) und, dass der mindestens ein Programmteil mittels der Programm ID von den Texturprozessoren importiert wird (Merkmal **M1.12**). Die neu hinzugekommenen Merkmale können jedoch keine erfinderische Tätigkeit begründen.

Dies begründet sich z. B. durch die Druckschrift **NK13**, welche eine etwas frühere Realisierungsstufe desselben *PixelFlow*-Systems betrifft (**NK13** Seite 59 „Abstract“, dritter Absatz, siehe „*PixelFlow*, an experimental graphics engine under construction“) und in der Druckschrift **NK12** zitiert wird (**NK12** Seite 68 „LAST95“).

So offenbart die Druckschrift **NK13** die Anwendung verschiedener Programmiermodelle auf das *PixelFlow*-System der Druckschrift **NK12**, insbesondere auf dessen *Shader Flow Units* (vgl. **NK13**, Seite 64, Abschnitt „Shader programming model“). Auch anhand der Druckschrift **NK13** wird klar, dass eine Programmierung der Texturverarbeitung vor dem Prioritätsdatum des Streitpatents bereits bekannt war (vgl. **NK13**, Seite 64, linke Spalte, Abschnitt „Shader programming model“, erster und zweiter Absatz; Absatz „High-level model“ ff), und der damit generierte Code jedenfalls spätestens während der in der Druckschrift **NK12** beschriebenen Forschungsarbeiten in die Verarbeitungselemente *(PE)* des *SIMD*-Prozessors geladen und dort zur Ausführung gebracht werden konnte. Mit anderen Worten: die Verarbeitungselemente *(PE),* die die Aufgaben von Texturprozessoren übernehmen, waren programmierbar.

In diesem Zusammenhang offenbart die Druckschrift **NK13** eine Programmidentifikation i. S. d. Hilfsantrags VII, die als *Shader ID* bezeichnet wird und die Bestandteil einer Datenstruktur für „Appearance“-Parameter ist (vgl. **NK13**, Fig. 8). Die *Shader ID* wird vom Steuerprogramm für das Shading benutzt, um für jedes Pixel einen *Shader Code*, d. h. einen Programmteil auszuwählen (vgl. **NK13**, Seite 65, linke Spalte, Abschnitt 4.1, zweiter Absatz). Dass dieser *Shader Code* u. a. für die Texturverarbeitung auf den Shadern bzw. deren Verarbeitungselementen *(PE)* zuständig ist, ergibt sich z. B. aus Figur 9, wo auf die Anwendung verschiedener Texturen Bezug genommen wird. Der Fachmann wird obige Datenstruktur für „Appearance“-Parameter als das anspruchsgemäße Fragment verstehen, das u. a. eine *Shader ID* als Programmidentifikation beinhaltet und das an die Verarbeitungselemente *(PE)* zur Texturverarbeitung übergeben werden muss. Anhand der *Shader ID*, die mitsamt dem Fragment übermittelt wird, wird der zum Fragment passende Shading-Programmcode für das Verarbeitungselement (PE) ausgewählt. Die Merkmale **M1.11** und **M1.12** sind demnach aus der Druckschrift **NK13** bekannt.

Da die Lehre der Druckschrift **NK13** auf das 1995 bekannte experimentelle Grafiksystem *PixelFlow* zugeschnitten und auf diesem demonstriert worden war (**NK13** Seite 59 „Abstract“, dritter Absatz, siehe „*PixelFlow*, an experimental graphics engine under construction“), war es ohne Weiteres naheliegend, diese Lehre auch noch bei dem konkret zwei Jahre später beschriebenen *PixelFlow*-System gemäß Druckschrift **NK12** anzuwenden.

7.2.2 Die Beklagte weist darauf hin, dass in der Lehre der Druckschrift **NK13** nicht vorgesehen sei, ein Fragment mit Programmidentifikation an einen Texturprozessor zu übergeben, mit deren Hilfe der Texturprozessor einen Programmteil importiert.
Dem Einwand kann nicht zugestimmt werden. So lehrt die Druckschrift **NK13**, dass die „Appearance“- bzw.- „Shader-Parameter“ aus Figur 8, die zusammen ein Fragment bilden, im *PixelFlow*-System auf zwei verschiedenen Übertragungswegen an einen Shader kommuniziert werden: einerseits kann die Übertragung über das „Geometry Network“, andererseits über das „Composition Network“ erfolgen (vgl. **NK13**, Seite 64, linke Spalte, Abschnitt „Shader parameters“, erster und zweiter Absatz, siehe „There are two ways to communicate parameters to a shader node. One is to send the parameters over the composition network. The other is to send the parameters over the front-end geometry network.“; Seite 63, linke Spalte, Abschnitt 3.2, zweiter Absatz, siehe „When the rasterizers have finished with a particular region, they send appearance parameters and depth values for each pixel onto the image-composition network, where they are merged and loaded into a shader.“). Insbesondere können die Parameter in den lokalen Speicher eines Verarbeitungselements *(PE)* geschrieben werden, wenn sie dort zur Verarbeitung benötigt werden (vgl. **NK13**, Seite 64, linke Spalte, Abschnitt „Shader parameters“, zweiter Absatz, siehe „If the parameter is needed in the local memory of the pixel processors, it can be broadcast locally at a shading node.“). Der lokale Speicher eines Verarbeitungselements *(PE)* wird indes in Druckschrift **NK12** näher beschrieben (vgl. **NK12**, Seite 62, Abschnitt „Local Memory“ und Fig. 6). Für den Fachmann ist ohne Weiteres ersichtlich, dass dementsprechend auch die *Shader ID* als Programmidentifikation an ein Verarbeitungselement *(PE)* geschickt werden

muss, wenn anhand von Mikrobefehlen des Steuerprogramms ein *Shader code* für ein Pixel ausgewählt und abgearbeitet werden soll (vgl. **NK13**, Seite 65, linke Spalte, Abschnitt 4.1, zweiter Absatz, siehe „The shader ID is used by the shading control program to select the shader code for each pixel.“).

7.3 Der nebengeordnete Patentanspruch 8 kann nicht günstiger beurteilt werden.

In den im Patentanspruch 8 gemäß Hilfsantrag VII neu hinzugekommenen Merkmalen **M8.3.7** und **M8.3.8** wird im Wesentlichen beansprucht, dass jedes Fragment eine Programmidentifikation beinhaltet und unter Verwendung dieser Programmidentifikation ein Programmteil (in den Texturprozessor) geholt wird.

Da die Druckschrift **NK13** mit der *Shader ID* gerade eine solche Programmidentifikation offenbart, die darüber hinaus noch in einem Fragment enthalten ist (siehe oben), können die neuen Merkmale allein eine Patentfähigkeit nicht begründen. Da außerdem der Patentanspruch 8 gemäß Hilfsantrag IV, auf dem der Patentanspruch 8 gemäß Hilfsantrag VII beruht, inhaltlich nicht über den Patentanspruch 1 gemäß Hilfsantrag IV hinausgeht, ist auch die Lehre des Patentanspruchs 8 gemäß Hilfsantrag VII mit Rücksicht auf den den Druckschriften **NK12** und **NK13** entnehmbaren Stand der Technik nahegelegt.

7.4 Die Patentansprüche 1 und 8 gemäß Hilfsantrag VII haben daher auch jeder für sich allein betrachtet keinen Bestand. Mit ihnen fällt der gesamte Hilfsantrag.

8. **Hilfsantrag VIIa** in der Fassung vom 19. November 2020 kann nicht günstiger beurteilt werden, da die Lehre seines Patentanspruchs 1 nicht auf erfinderischer Tätigkeit beruht.

8.1 Patentanspruch 1 gemäß Hilfsantrag VIIa beruht auf Patentanspruch 1 gemäß Hilfsantrag VII, wobei Merkmal **M1.12** durch Merkmal **M1.12´** ersetzt worden

ist, unter Änderung der Formulierung „a portion of the program is *imported* by each texture processor“ in „a portion of the program is *used* by each texture processor“.

Der Senat versteht das neue Merkmal **M1.12´** so, dass unter Verwendung der Programmidentifikation wenigstens ein Teil des Programms von jedem Texturprozessor genutzt wird.

Insoweit geht Patentanspruch 1 gemäß Hilfsantrag VIIa inhaltlich nicht über Patentanspruch 1 gemäß Hilfsantrag VII hinaus, weswegen er nicht anders als dieser beurteilt werden kann. Zur Vermeidung von Wiederholungen sei an dieser Stelle lediglich auf die entsprechenden Ausführungen zum Hilfsantrag VII hingewiesen.

8.2 Mit dem Patentanspruch 1 gemäß Hilfsantrag VIIa fällt der gesamte Hilfsantrag.

8.3 Unter den vorgenannten Umständen kann dahinstehen, ob der Hilfsantrag VIIa schon wegen Verspätung nach § 83 Abs. 4 Satz 1 PatG zurückzuweisen war.

9. In seiner Fassung nach **Hilfsantrag VIII** hat das Streitpatent keinen Bestand, weil der Gegenstand seines Patentanspruchs 1 nicht neu ist.

9.1 Der Hilfsantrag VIII beruht auf dem Hilfsantrag VI, wobei der Patentanspruch 1 um das Merkmal **M1.13** ergänzt wurde. Das neue Merkmal schränkt die Verarbeitung der Texturprozessoren dahingehend ein, dass nur dann eine Ausgabe erzeugt wird, wenn die Verarbeitung des Fragments abgeschlossen ist.

Der unabhängige Patentanspruch 8 wurde entsprechend angepasst.

9.2 Das in Patentanspruch 1 gemäß Hilfsantrag VIII hinzugekommene Merkmal kann eine Patentfähigkeit nicht begründen.

So ist aus Druckschrift **NK12** bekannt, dass nachdem alle Verarbeitungen für ein Pixel abgeschlossen sind, der endgültige Pixelwert an das „Image Composition Network“ übergeben wird, um ein endgültiges Bild zu erhalten. Die Druckschrift **NK12** beschreibt, dass für die vollständig verarbeiteten Pixel („final shaded pixel“) der „Unload“-Modus des „Image Composition Networks“ verwendet wird, damit der Pixelwert in den Bildpuffer gelangt (vgl. **NK12**, Seite 63, linke Spalte, letzter Absatz, siehe „Unload mode is used, to dump final shaded pixels out of the shader …“). Dabei dürfte klar sein, dass ein Pixelwert von dem Verarbeitungselement (PE) nur dann an das „Composition Network“ weitergegeben wird, wenn für die Rohpixeldaten das gesamte *Shading* und damit auch die gesamte Texturverarbeitung ausgeführt worden ist.

Merkmal **M1.13** ist demnach in der Druckschrift **NK12** offenbart.

9.3 Die Beklagte argumentiert, aus der Druckschrift **NK12** gehe nicht hervor, dass ein Texturprozessor erst dann eine Ausgabe erzeuge, wenn ein Fragment vollständig verarbeitet ist.

Der Einwand greift nicht durch. So ist bereits zu Hilfsantrag VI ausgeführt, dass im *PixelFlow-System* der Druckschrift **NK12** ein Verarbeitungselement *(PE)* das komplette *Shading* bzw. die gesamte Texturverarbeitung eines Pixels übernehmen kann. Dies gilt insbesondere dann, wenn auch nur eine einzige Textur verarbeitet werden soll. Erst wenn das *Shading* eines Pixels auf einem Verarbeitungselement *(PE)* abgeschlossen ist, wird dieses an den Framebuffer zur Anzeige ausgegeben (vgl. **NK12**, Seite 59, rechte Spalte, Abschnitt 2.5, erster Absatz, siehe „After shading, regions of shaded pixels are forwarded to a frame buffer for display.“).

9.4 Unter Berücksichtigung der Ausführungen zum Hilfsantrag VI ist der Gegenstand des Patentanspruchs 1 gemäß Hilfsantrag VIII durch die Druckschrift **NK12** neuheitsschädlich vorweggenommen. Damit ist auch der Patentanspruch 1 in der Fassung des Hilfsantrags VIII nicht patentfähig. Mit dem Patentanspruch 1 fällt der gesamte Hilfsantrag.

10. Aus diesen Gründen war das Streitpatent, das somit in keiner seiner durch die Beklagte verteidigten Fassungen Bestand hatte, insgesamt für nichtig zu erklären.

V.

Nach dem Vorstehenden kommt es auf die Frage einer offenkundigen Vorbenutzung des sogenannten Voodoo^2-Chipsatzes und des N… RIVA TNT im Ergebnis nicht mehr an.

VI.

Die Kostenentscheidung beruht auf § 84 Abs. 2 Satz 1 und 2 PatG, § 91 Abs. 1, § 269 Abs. 3 Satz 2 ZPO.

Die Klägerin zu 3, die bereits im Januar 2020 die Klage zurückgenommen hat, ist im Kostenausspruch des Urteils hinsichtlich der Gerichtskosten noch zu berücksichtigen, weil die Klagerücknahme nicht den gesamten Streitgegenstand betrifft (vgl. BGH GRUR 2020, 599, Rn. 69 – *Rotierendes Menü*; BGH NJW-RR 1999, 1741). Einen Antrag nach § 269 Abs. 4 ZPO, über die Kostenverpflichtung bei Klagerücknahme nach § 269 Abs. 3 Satz 2 ZPO zu entscheiden, hat die Beklagte nicht gestellt; über die außergerichtlichen Kosten ist daher insoweit keine Entscheidung veranlasst. Hinsichtlich der Gerichtskosten war hingegen gemäß § 84

Abs. 2 Satz 1 PatG, § 308 Abs. 2 ZPO auch ohne Antrag der Beklagten aus den eingangs genannten Gründen eine einheitliche Entscheidung im Urteil zu treffen.

Der Ausspruch über die vorläufige Vollstreckbarkeit beruht auf § 99 Abs. 1 PatG i. V. m. § 709 ZPO.

VII.

Rechtsmittelbelehrung

Gegen dieses Urteil ist das Rechtsmittel der Berufung gegeben.

Die Berufungsschrift muss von einer in der Bundesrepublik Deutschland zugelassenen Rechtsanwältin oder Patentanwältin oder von einem in der Bundesrepublik Deutschland zugelassenen Rechtsanwalt oder Patentanwalt unterzeichnet und innerhalb eines Monats beim Bundesgerichtshof, Herrenstraße 45a, 76133 Karlsruhe eingereicht werden.

Die Berufungsfrist beginnt mit der Zustellung des in vollständiger Form abgefassten Urteils, spätestens aber mit dem Ablauf von fünf Monaten nach der Verkündung. Die Berufungsfrist kann nicht verlängert werden.

Die Berufungsschrift muss die Bezeichnung des Urteils, gegen das die Berufung gerichtet wird, sowie die Erklärung enthalten, dass gegen dieses Urteil Berufung eingelegt werde. Mit der Berufungsschrift soll eine Ausfertigung oder beglaubigte Abschrift des angefochtenen Urteils vorgelegt werden.

Püschel Baumgardt Dr. Schnurr Dr. Forkel Dr. Städele

Fi