Computer Animation Competition 2024

Aufgabe

- Teilnahme ist Voraussetzung für Klausurzulassung!
- Verwenden Sie Ihre Grafikengine, um eine 3D-Animation zu erzeugen
- Teilnahme einzeln oder (sehr empfohlen!) in Teams von 2–3 Personen
- Implementieren Sie weitere Funktionen/Features aus dem beigefügten Katalog (Punktezahl korreliert grob mit Aufwand). Mindestumfang: **100 Punkte pro Person**. Die zuerkannten Punkte je Feature sind in Einzelfällen verhandelbar.
- Features müssen aus mindestens zwei (Einzelbeiträge), drei (Zweierteam) bzw. vier (Dreierteam) Kategorien stammen
- Eigene Features sind ebenfalls möglich und erwünscht!
- Modellierungstools wie Blender sowie fertige 3D-Modelle, Texturen usw. sind selbstverständlich erlaubt (aber bitte Lizenzen beachten!)

Ablauf

- Besprechung Teilnahmeregeln + Featurekatalog (Freitag 31. Mai)
- Abgabe eines Konzepts (bis Sonntag 9. Juni, 18:00 Uhr)
  - Kurzvorstellung des Teams
  - Priorisierte Liste zu implementierender Komponenten (mit Zuständigkeit der Teammitglieder)
  - Beschreibung der Story
    - Abstract / Kurze verbale Darstellung der Geschichte (2-3 Sätze)
    - Handgezeichnetes Storyboard (Websuche “how to draw a storyboard” o.ä.)
    - Wie wird die Story von den ausgewählten technischen und gestalterischen Elementen “getragen” bzw. unterstützt?
  - Technisches Konzept
    - Flussdiagramm / Zusammenspiel der Komponenten untereinander
    - Mit welchen Herausforderungen/Schwierigkeiten rechnen Sie? Wie gehen Sie mit Risiken um?
- Project Pitch und Diskussion (Montag 10. Juni)

  - Stellen Sie Ihr Konzept (künsterisch und technisch) in 5 Minuten vor
  - Konstruktive Kritik

- Vorschauvideo (Frist: Dienstag, 16. Juli, 23:59)
  - Videodatei in beliebiger Qualität und Auflösung
  - Letzter Baustein zum "Übungserfolg" (Zulassungsvoraussetzungen Klausur)
  - Screengrab, Teilergebnisse, kommentiertes Handyvideo ... alles vollkommen OK!

- Abgabe der finalen Animation (Frist: Donnerstag, 18. Juli, 23:59)
  - MPEG4-codierte Videodatei in (mindestens) 1080p30 Auflösung
  - Compilier- und lauffähiger Code mit allen Abhängigkeiten und Eingabedaten
  - Bebilderte Dokumentation als HTML-Dokument oder PDF
  - Ein fertiger Film ist natürlich schön, aber auch "gut dokumentertes Scheitern" kann für alle sehr interessant und wertvoll sein

- Vorführung und Preisverleihung (Freitag, 19. Juli)
  - Präsentation der Ergebnisse samt "Making-Of" (Video + max. 5 Minuten Demo / Präsentation je Team)

Featurekatalog

Kategorie 0: Modellierung

- Algorithmisch erzeugte (prozedurale) Geometrie (30P)
  Berge, Wellen, etc. — gerne auch implizit via SDFs und Ray Marching im Fragment Shader [1]

- Boolesche Geometrie (60P)
  Komplexe Geometrien aus einfachen Grundformen erzeugen [2]

- Physikalische Simulation (50P)
  Textilien, Wasser, Feuer, Haare, Textilien, Gummiobjekte, usw.

- Rigging und Blending (50P)
  Komplexe 3D-Modelle durch Skelette deformieren/animieren [3]

- Subdivision Surfaces (40P)
  Glatte Oberflächen durch Verfeinerung grober Kontrollnetze [4]

- Partikelsysteme (40P)
  Erzeugen Sie große Mengen Partikel (z.B. Feuer, Rauch, "Feenstaub", ...) mittels CPU oder GPU (z.B. mit geometry/tessellation shader)

- Displacement Mapping (40P)
  Erzeugen Sie Gebirge, Wasserwellen, Hautfalten usw. aus planaren Flächen mit Höhentextur
- Fell, Haare (40P)
  Erzeugen und schattieren Sie Haare als 1D-Primitives [5]

Kategorie 1: Kameraeffekte

- Fokusunschärfe (20P)
  Rendern Sie mehrere Lochkamera-Bilder von verschiedenen Augpunkten und mitteln Sie diese, um ein Bild mit Tiefenunschärfe zu erhalten
- Bewegungsunschärfe (20P)
  Mitteln Sie Bilder von unterschiedlichen Zeitpunkten innerhalb eines Frames, um "motion blur" zu erhalten.
- HDR-Effekte (Glare, ...) (20P)
  Filtern Sie einen Framebuffer hohen Dynamikumfangs mit einem Unschärfekernel.
- Filmrauschen (10P)
  Statten Sie Ihre Pipeline mit "Dreckeffekten" wie Filmkörnung aus.
- Lichtfeld-Rendering (30P)
  Rendern Sie 4D-Bilder zur Darstellung auf einem "holografischen" Display. "Looking Glass" Display steht zur Verfügung!
- Physikalisch realistische Linseneffekte (20P)
  Aberrationen, usw.

Kategorie 2: Lokale Beleuchtung

- **Shadow Mapping** (30P)
  Schattenwurf von Punktlichtquellen
- Flächenlichtquellen 1 (20P)
  Verteilen Sie Punktlichtquellen auf flächigen Strahlern, um weiche Schatten und Glanzlichter zu erhalten.
- Spherical Harmonics Illumination (60P)
  Effiziente Darstellung komplexer BRDFs in Umgebungslicht [6]
- Analytische Flächenlichtquellen (60P)
  Ausgedehnte Lichtquellen in BRDF-Modell backen [7]
- Datengetriebene Materialmodelle (40P)
  Gemessene Materialien, z.B. aus MERL-Datenbank oder Bonner BTFs [8, 9]
- Weiche/Gefilterte Schatten (80P)
  mehrere mathematische Ansätze möglich [10, 11, 12]
- Prozedurale Texturen (10P)

Kategorie 3: Globale Beleuchtung

Doppelte Punktzahlen beziehen sich auf GPU-Vorberechnung bzw. echte CPU-Implementierung.

- Umgebungsverdeckung (Ambient Occlusion) (30P/80P)
  Realistische Abdunklung verdeckter Regionen (Hohlräume usw.) [13]
- Radiosity (50P/80P)
  Globale Beleuchtung für diffuse Oberflächen [14, 15]
- Instant Radiosity (30P/50P)
  Interreflexionen durch virtuelle Punktquellen[16]
- Screen-space Globale Beleuchtung (SSAO/SSDO) (70P)
  Näherungsweise Berechnung von Ambient Occlusion oder Interreflexionen in 2D [17]
- Volumetrische Einfachstreuung (50P)
  [18]
- Raytracing-Effekte (50P/80P)
  Erweitern Sie Ihre optischen Effekte um Spiegelung, Brechung, usw. mittels Raytracing. Entweder vorberechnet auf CPU oder live berechnet auf GPU (NVIDIA Optix ...)

Kategorie 4: Interaktive Features und Sonstiges

- **Cinematic Engine** (50P)
  Asset Management, Keyframe-Interpolation (Splines), Szenen, Kamerafahrten, usw.
- Datengetriebene Echtzeit-Modelle (60P)
  z.B. RGB-D Datenstrom von Kinect-Kamera als Input
- Interaktive Anwendung (20P)
  Animation reagiert in Echtzeit auf User-Eingaben
- Stilisiertes Rendering (30P)
  Verbiegen Sie die Pipeline für nicht-fotorealistische Ergebnisse (künstlerisch stilisiert, Cartoon-Stil/Cel-Shading [19], ...)
- Sound (20P/40P)
  Unterlegen Sie Ihre Animation mit Ton, entweder als vorberechneter Audiotrack oder in Echtzeit synthetisiert.

References

[1] Inigo Quilez. Tutorials. `https://iquilezles.org/articles/`.

[2] Nigel Stewart, Geoff Leach, and Sabu John. An improved z-buffer csg rendering algorithm. In *Workshop on Graphics Hardware*, pages 25–30, 1998.

[3] Skeletal animation. https://www.khronos.org/opengl/wiki/Skeletal_Animation.

[4] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, The University of Utah, 1987.

[5] Cem Yuksel and Sarah Tariq. Advanced techniques in real-time hair rendering and simulation (siggraph 2010 course. http://developer.download.nvidia.com/presentations/2010/SIGGRAPH/HairCourse_SIGGRAPH2010.pdf.

[6] J Kautz, J Snyder, and P-PJ Sloan. Fast arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *Rendering Techniques*, volume 28, pages 291–296. Eurographics Association, 2002.

[7] James Arvo. *Analytic methods for simulated light transport*. PhD thesis, Yale University, 1995.

[8] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[9] https://www.merl.com/brdf/.

[10] Thomas Annen, Tom Mertens, Philippe Bekaert, Hans-Peter Seidel, and Jan Kautz. Convolution shadow maps. In *Proceedings of the 18th Eurographics Symposium on Rendering Techniques*, pages 51–60. Eurographics Association, 2007.

[11] Thomas Annen, Tom Mertens, Hans-Peter Seidel, Eddy Flerackers, and Jan Kautz. Exponential shadow maps. In *Proceedings of Graphics Interface 2008*, GI '08, pages 155–161, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.

[12] Christoph Peters and Reinhard Klein. Moment shadow mapping. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, i3D '15, pages 7–14, New York, NY, USA, 2015. ACM.

[13] Michael Bunnell. Dynamic ambient occlusion and indirect lighting. *GPU Gems*, 2(2):223–233, 2005.

[14] Cindy M Goral, Kenneth E Torrance, Donald P Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *ACM SIGGRAPH computer graphics*, volume 18, pages 213–222. ACM, 1984.

[15] Donald P Greenberg, Michael F Cohen, and Kenneth E Torrance. Radiosity: A method for computing global illumination. *The Visual Computer*, 2(5):291–297, 1986.

[16] Alexander Keller. Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[17] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 75–82. ACM, 2009.

[18] Kenny Mitchell. Volumetric light scattering as a post-process. In Hubert Nguyen, editor, *GPU Gems 3*. 2007. `https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch13.html`.

[19] Raul Luque. The cel shading technique. `http://raulreyesfinalproject.files.wordpress.com/2012/12/dissertation_cell-shading-raul_reyes_luque.pdf`.