

preschern.org

Klaus Preschern

Version 2025 *

mail@preschern.org

email

Contents

1 Introduction	3
2 Projects	4
2.1 glTF - JPEG of 3D	4
2.2 WebAssembly	4
2.3 MaterialX with WebGL	5
2.4 Java interface for WebGL	5
2.5 Content Management with \LaTeX	6
2.6 Solid Modeling with Implicit Surfaces	6
2.7 Universal 3D Interface	7
2.8 3D Reconstruction with Implicit Surfaces	8
2.9 2D und 3D Delaunay Triangulation	8
2.10 JavaScript Image Linker	9
2.11 WEB-Dock	9
2.12 JaRT - a Java Ray Tracer	9
2.13 3D Desktop Browser	10
2.14 3D Desktop	10
2.15 Generating 3D Icons in Real-Time	11
2.16 Real-Time Rendering	12
2.17 Generating 3D Icons with Ray Tracing	12
2.18 Modula-3 for OS/2 and MS-DOS	13
2.19 Modula-2 to C Translator	13

3	Details of selected projects	14
3.1	3D Reconstruction with Implicit Surfaces	14
3.2	2D and 3D Delaunay Triangulation	17
3.3	JavaScript Image Linker	22
3.4	3D Desktop	24
4	About the Author	26
5	Dictionary of PDF Documents	27
6	Disclaimer	28
6.1	Content	28
6.2	Referrals and Links	28
6.3	Copyright	29
6.4	Privacy policy	29
6.5	Legal validity of this disclaimer	29
7	References	30

1 Introduction

preschern.org does describe some software projects of the author. This document is the result of one of these projects (see Section 2.5 for details). 3D models are visualized in current Internet browsers with WebGL (see section 2.4).

The web site is powered by  (Section 2.10).

2 Projects

2.1 glTF - JPEG of 3D



glTF (Graphics Library Transmission Format or GL Transmission Format and formerly known as WebGL Transmissions Format or WebGL TF) is a standard file format for three-dimensional scenes and models. An open standard [12] developed and maintained by the Khronos Group, it supports 3D model geometry, appearance, scene graph hierarchy and animation. To support glTF on Windows desktops as well as in browser environments with WebGL, Visual Studio C++ and Emscripten have been used to implement a glTF-Viewer, which does support HDRI (high dynamic range imaging) with glTF 2.0 PBR (physically based rendering) and MaterialX (Section 2.3).

glTF PBR examples: Damaged Helmet, Suzanne, dragon model, Voronoi diagram on the sphere, Chess Set, Box Animation.

MaterialX examples: Brick Pattern, Copper, Suzanne, BMW-M6.

Some loading time required: MaterialX Chess Set.

2.2 WebAssembly



WebAssembly (abbreviated Wasm) is a safe, portable, low-level code format designed for efficient execution and compact representation. Its main goal is to enable high performance applications on the Web, but it does not make any Web-specific assumptions or provide Web-specific features, so it can be employed in other environments as well. WebAssembly is an open standard developed by a W3C Community Group [24].

The code generator of JaSIL (Section 2.10) has been improved to generate C/C++ code, which is then translated to WebAssembly with the help of Emscripten. With the following examples it is possible to compare the performance of Javascript and WebAssembly: preschern.org, dragon model, CSG example, Voronoi diagram on the sphere, WebGL-Demo, MaterialX-Demo, BMW-M6 (the BMW example does use a sim-

ple mark and sweep garbage collector).

2.3 MaterialX with WebGL



MaterialX is an open standard for representing rich material and look-development content in computer graphics, enabling its platform-independent description and exchange across applications and renderers. The support of the OpenGL ES Shading Language (GLSL Version 3.00 ES) enabled the integration of MaterialX in the Java-WebGL interface (Section 2.4).

The following links [MaterialX-Demo](#) and [BMW-M6](#) show examples with MaterialX. This demos take some time to load and require a browser which does support the „Compression Stream API” (in Firefox since version 113).

2.4 Java interface for WebGL



JaSIL (Section 2.10) has been used to implement a WebGL [27, 28] interface with Java. It has been used to build a viewer for 3D models stored with Universal 3D (see Section 2.7).

The following links can be used to open the U3D viewer with the dragon model, the CSG example or the Voronoi diagram on the sphere.

The mouse wheel is used for zooming in the U3D viewer and dragging with the mouse does rotate the camera. A Phong shader is used to simulate light.

2.5 Content Management with \LaTeX

\LaTeX

The content of preschern.org is described with \LaTeX [18, 19] and can be translated to different formats. In this way it is possible to create documents with a table of contents, a list of references, citations and more the like, with tools used for further processing. It is in example possible to create this PDF document and to embed 3D models with the help of the Universal 3D interface, described in Section 2.7.

A compiler, written with Java, does translate the \LaTeX subset, used for the description of the content, including some special \LaTeX macros (called "hints"), to XML. This XML description is then translated to DHTML, with the help of XML layout descriptions (so called "parts"). Via these layout descriptions, the content is connected to JavaScript, which has been created with JaSIL (Section 2.10) from Java.

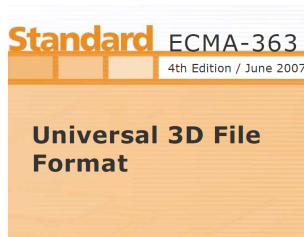
2.6 Solid Modeling with Implicit Surfaces



A signed distance field function assigns to every point $|x,y,z| \in \mathbf{R}^3$ a distance with a positive sign for points outside the region enclosed by the represented surface, the value 0 for points on the surface and a negative sign for points inside the surface. Operations like point location or boolean operations for solid modeling (Constructive Solid Geometry [CSG]) can be implemented quite efficiently with such functions.

Dual Contouring [15] has been used for the feature preserving extraction (polygonization) of surfaces. The following WebGL link shows a 3D model created with implicit functions and extracted with *Dual Contouring*: [WebGL](#).

2.7 Universal 3D Interface



Visualization includes the notion of pluralism and perspectivism since there is an explicit attempt at *representing* one, often textual (i.e. mathematical), description in terms of a graphical one [11].

The standard ECMA-363 *Universal 3D File Format (U3D)* [7] defines the structures for storing 3D models. An interface, implemented with Java, does support a subset of U3D for reading and storing 3D scenes, created with the graphics library described in Section 2.16.

With the help of the \LaTeX "movie15" package multimedia content, like movies, sound and objects described with U3D, can be embedded in PDF documents, according to the Adobe PDF specification version 1.6. With appropriate software, i.e. the Acrobat Reader version ≥ 8 , such objects are displayed in 3D. This enables a high quality representation of textual and graphical content.

2.8 3D Reconstruction with Implicit Surfaces



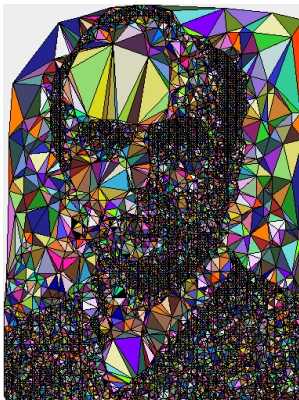
A surface can be described with an implicit function of the form $f(x, y, z) = f(p) = 0$. A point p is on the implicit surface if the result is zero when the point is inserted into the implicit function f .

A simple example is the unit sphere, which has $f(x, y, z) = x^2 + y^2 + z^2 - 1$ as its implicit function. Blending of implicit surfaces can be used for the reconstruction of surfaces from unoriented point sets. The basic idea is to blend simple primitives, such as spheres. Each sphere can be seen as an "atom", and a "molecule" is obtained by blending its atoms.

The following WebGL link shows a 3D model of two blended spheres: [WebGL](#).

More in Section 3.1 ...

2.9 2D und 3D Delaunay Triangulation



The Delaunay triangulation and the Voronoi diagram are dual structures and contain the same information in different form. They record everything about proximity of a set of objects in space.

These structures are used in applications for nearest neighbor queries, triangulations with maximized smallest angles, largest empty circles, minimum spanning trees or traveling salesperson paths.

More in Section 3.2 ...

2.10 JavaScript Image Linker



JaSIL is a compiler/linker which does translate Java classes to JavaScript and creates a optimized program, which can be executed in current web browsers. The JDK currently used is Apache Harmony. Furthermore a number of Java packages have been developed to create GUIs in web browsers and to support WebGL. Support for preemptive Java threads is available also, but optional. JaSIL does create a compact optimized JavaScript program which contains the necessary parts only. For memory management the system does use the garbage collector of JavaScript.

More in Section 3.3 ...

2.11 WEB-Dock



The WEB-Dock provides fast and direct access to important and/or frequently used web resources. Compared to similar solutions, like the task list (Task-Dock) of the *3D Desktops* (Section 2.14) or Apple's Dock, the efficient image-based rendering approach makes it possible to use the WEB-Dock with current web browsers. The scaling of the icons, which can be used to increase the icon density, and drag & drop are realized with fast image switching, like in an animated cartoon. Stopping the cursor over one, in this way selected, icon shows a short tool tip (hover help).

2.12 JaRT - a Java Ray Tracer



JaRT is a ray tracer developed with Java, which is used to create GUI elements with 3D effects for web pages, like round corner boxes or frames with round edges. JaRT does support CSG operations for primitives like planes, boxes, spheres, cylinders, cones and for triangle meshes, including 3D text. Objects can be textured with planar, spher-

ical and cylindrical texture projections. The ray tracer does use Phong/Blinn lightning equations with reflections and shadows. Image quality can be improved with supersampling.

2.13 3D Desktop Browser



The structure and the elements of the *3D Desktop* (Section 2.14) can be described with XML. The desktop browser creates a 3D view from the XML description. Callbacks (i.e. for menus) can be written with JavaScript. The Document Object Model (DOM) of the desktop does provide an interface for accessing desktop elements.

2.14 3D Desktop



Based on the real-time rendering library, described in Section 2.16, a number of software components for a 3D desktop have been developed. The desktop components implement Swing drag and drop interfaces and context menus. Components which support circle lists, spiral lists and a 3D task list (Task Dock) to show activated functions, are available. The abstract list models are derived from the corresponding Swing models. A 3D folder instance provides a space for placing 3D objects around a center.

Screenshots in Section 3.4 . . .

2.15 Generating 3D Icons in Real-Time



The real-time rendering library, described in Section 2.16, greatly improves the performance of the 3D icon generation. A complex model, which requires several hours of processing with the ray tracer is now transformed within minutes into an image with responsive 3D elements. To improve the image quality supersampling is used for antialiasing.

To integrate 3D models and web content, layouts are defined in XML. From these descriptions HTML and JavaScript is generated with functions for camera movements and anima-

tions.

2.16 Real-Time Rendering



Based on JOGL, a Java/OpenGL [2, 25] interface, a real-time rendering library has been implemented. The components support reflections and shadows. Shaders have been developed with the OpenGL Shading Language (GLSL) [17]. A simple text system is used to translate text dynamically into 2D textures. The picking of objects is accelerated by the graphics hardware. It is possible to build 3D models with constructive solid geometry (CSG) operations from primitives. Furthermore with Blender created 3D objects can be used.

2.17 Generating 3D Icons with Ray Tracing



Some tools have been developed for generating images of 3D models with responsive areas (3D icons) which can be used in web designs. Such 3D elements are defined with the Java programming language and the ray tracer POV-Ray is used for rendering. Sensitive areas are defined by assigning "sensors" to 3D objects for which corresponding HTML code is generated automatically. Special panorama camera abstractions are used to create wide angle views without distortions.

An example generated with POV-Ray.

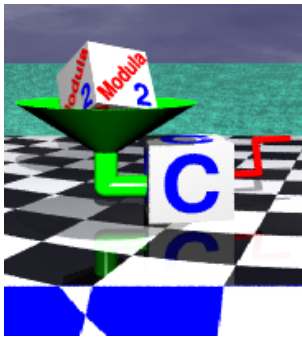
2.18 Modula-3 for OS/2 and MS-DOS





The Modula-3 system (compiler, runtime and library), which has been developed by Digital's Systems Research Center (DEC-SRC), has been ported to OS/2 (Warp and Merlin) and MS-DOS. Carsten Whimster has published an article about the Modula-3 for OS/2 project in the Electronic Development Magazine, OS/2 edition (EDM/2). The DOS port has been used for teaching.

- Carsten's article in EDM/2.
- Download Modula-3 for Warp or Merlin.
- Announcements of Modula-3 for OS/2 and MS DOS.
- More information about Modula-3 and some references [6, 14, 22].

2.19 Modula-2 to C Translator



The Modula-2 to C translator developed during this project is able to translate itself to C. The very portable translator is available for MS-DOS, UNIX and VMS.

- An introduction to parsing in German .
- A comprehensive project report in German .
- More information about Modula-2 and some references [4, 30].

3 Details of selected projects

3.1 3D Reconstruction with Implicit Surfaces

An implicit surface is defined by an implicit function which is a continuous function over the domain \mathbf{R}^3 . For example, a unit sphere may be defined using the implicit function $f(\mathbf{p}) = 1 - |\mathbf{p}|$, for points \mathbf{p} element of \mathbf{R}^3 . Points on the sphere are those locations at which $f(p) = 0$. A short introduction to implicit surfaces can be found in [1]. An important class of implicit surfaces are the "blobby surfaces" [5]. Such a surface is described by:

$$f(\mathbf{p}) = \sum_{i=0}^{n-1} h(r_i)$$

where n is the number of primitive objects and for each of these objects a distance r_i is computed, which is in example the distance from \mathbf{p} to these objects. The blending function h describes the region of influence of object i . For spheres, r_i is in example the distance from \mathbf{p} to the center of sphere i .



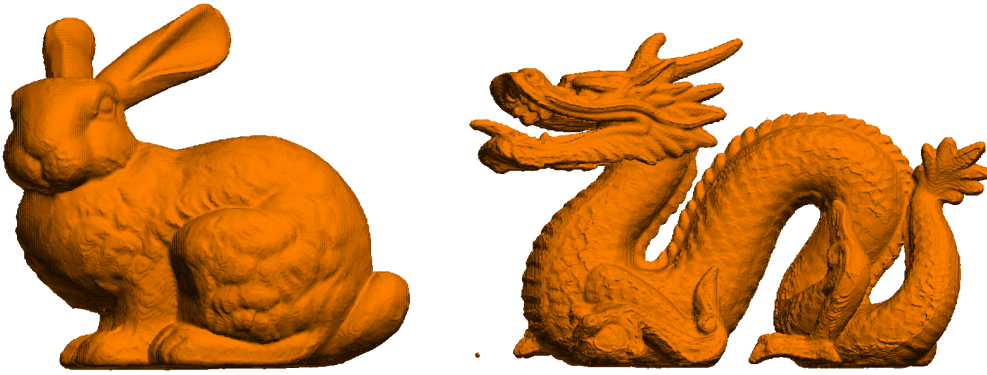
Figure 1: Two blended spheres.

In this case, in the above equation, a single function h_i describes the profile of a "blobby sphere" which is a Gaussian function that has a particular center and standard deviation. When the centers of two blobby spheres are close enough to one another, the implicit surface appears as though the two spheres have melted together.

Figure 1 shows two such blended spheres. All points on and between them, where they are blended together, are in the area of influence of the two centers and create the surface. The following WebGL link shows a corresponding 3D model: [WebGL](#).

To reconstruct the surface, a three-dimensional grid is placed over the area where the surface is located and the blending function is sampled at each grid point. A well-known method for creating a triangle mesh from such an implicit surface is the marching cubes [20] algorithm.

The following reconstruction examples (Figure 2) are based on the range data from the Stanford 3D Scanning Repository, used by courtesy of the Stanford Computer Graphics Laboratory. The rotation and translation information available with this range data has been used to move the parts (range images) together to form the starting point set. The information about mesh connectivity however has not been used for this examples. In this way the point set was used as unoriented point cloud.



(a) Bunny reconstructed from 362.272 points. (b) Dragon reconstructed from 1.235.765 points.



Figure 2: Reconstruction examples

Figure 3 shows the treatment of holes by different reconstruction approaches. What is best depends on the requirements of the application area. A hole is either intentional (in example drilled into the surface) or may be caused by low point density delivered by the scanning device. The bunny is shown from below where it has two holes and areas of low data density in the base plate.

Figure 3a is the result of Zipper [26]. The mesh produced by Zipper shows the holes and is not watertight.

The middle bunny (Figure 3b) is the one reconstructed with the above described implicit surface method. This bunny has a watertight mesh and shows the holes. It has a double sided surface.

For the right bunny (Figure 3c) the Poisson surface reconstruction [16] method has been used. This method does produce a watertight mesh with closed holes.

The following WebGL links show 3D models of the reconstruction examples¹: Zipper , Approach with implicit surfaces  and

¹Mesh quality reduced to 35K/70K vertices/faces.

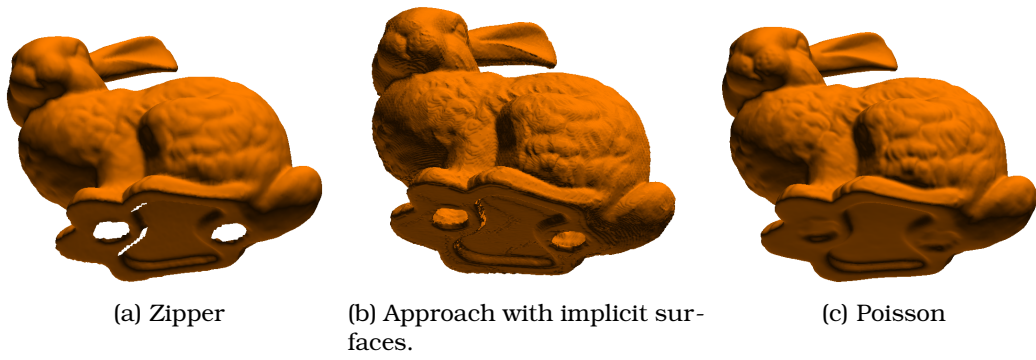


Figure 3: Reconstruction examples

Poisson 

3.2 2D and 3D Delaunay Triangulation

The Delaunay triangulation and the Voronoi diagram are dual structures and contain the same information in different form. Computing one of these structures does automatically create the other. Furthermore there is a connection between Delaunay triangulations and convex hulls in one higher dimension. The Delaunay triangulation and the Voronoi diagram in d dimensions can be constructed from a convex hull in $d+1$ dimensions. The Delaunay triangulation for a set of d -dimensional points is the projection of the points of the hull in $d+1$ dimensions. A introduction can be found in [23].

2D Delaunay Triangulation

Two algorithmic variants of the 2D Delaunay triangulation have been implemented. One is the incremental algorithm [21] and the other the recursive variant with a "divide and conquer" strategy [13]. The two following images show the Delaunay triangulation (Figure 4a) and the corresponding Voronoi diagram (Figure 4b), created from 5964 points.

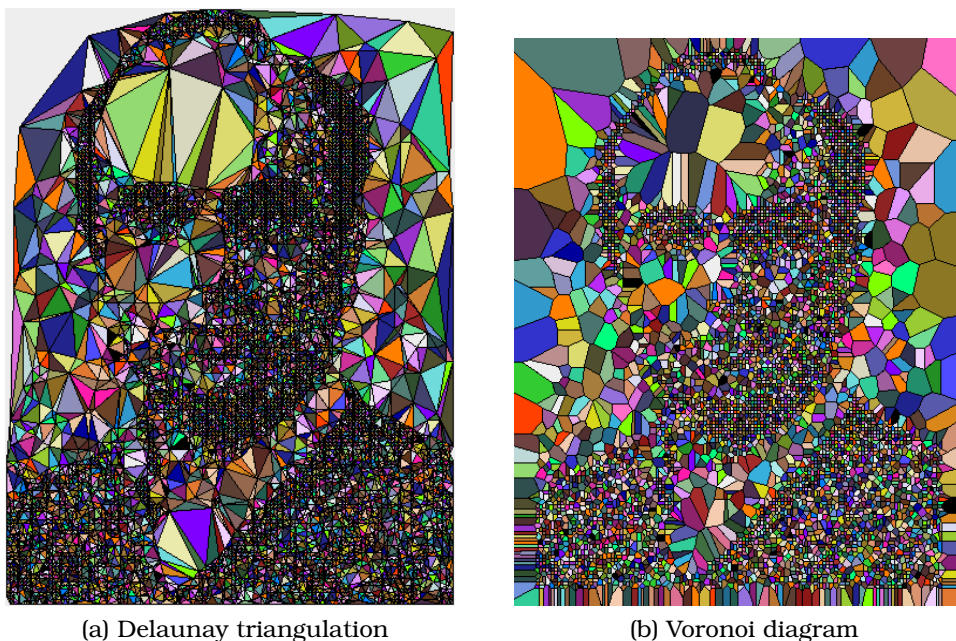


Figure 4: Delaunay triangulation or Voronoi diagram in 2D

Figure 5 shows both, the Delaunay triangulation and the Voronoi diagram, in one image, for a set of 10 points.

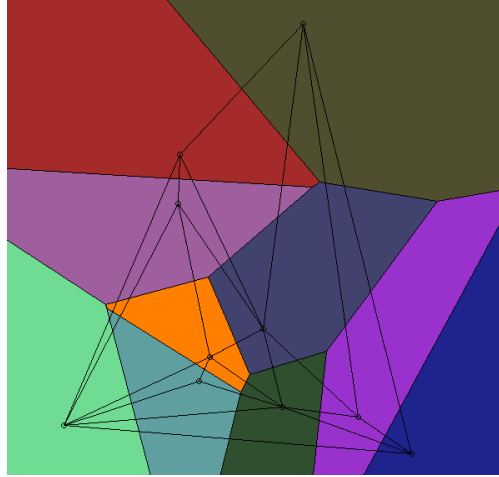


Figure 5: Delaunay triangulation and Voronoi diagram in 2D

Voronoi Diagram on the Sphere

Figure 6 shows a Voronoi diagram, created from the positions of the capitals of 228 countries (including overseas territories and the like) blended with the world map. The red dots show the GPS positions of the capitals, which have been transformed to 3D points using the WGS-84 [29] ellipsoid. The diagram has been created from the convex hull of these points. The following WebGL link shows a 3D model of this Voronoi diagram: [WebGL](#). Figure 7 shows this Voronoi diagram of the whole world in 2D.

3D Delaunay Triangulation

The implementation of the 3D Delaunay triangulation is based on the algorithm described in [9]. An extensive description (in German) can be found in [3].

The following two images (Figure 8a and Figure 8b) show the well known Stanford Bunny. The right one shows a 3D Delaunay triangulation of 71 weighted points of its point set.

The 3D Delaunay triangulation for point sets with symmetries can be generated with the help of *Simulation of Simplicity* [8]. The following two images (Figure 9a and Figure 9b) show an icosahedron. The right Figure 9b shows a possible 3D Delaunay triangulation of this platonic solid.

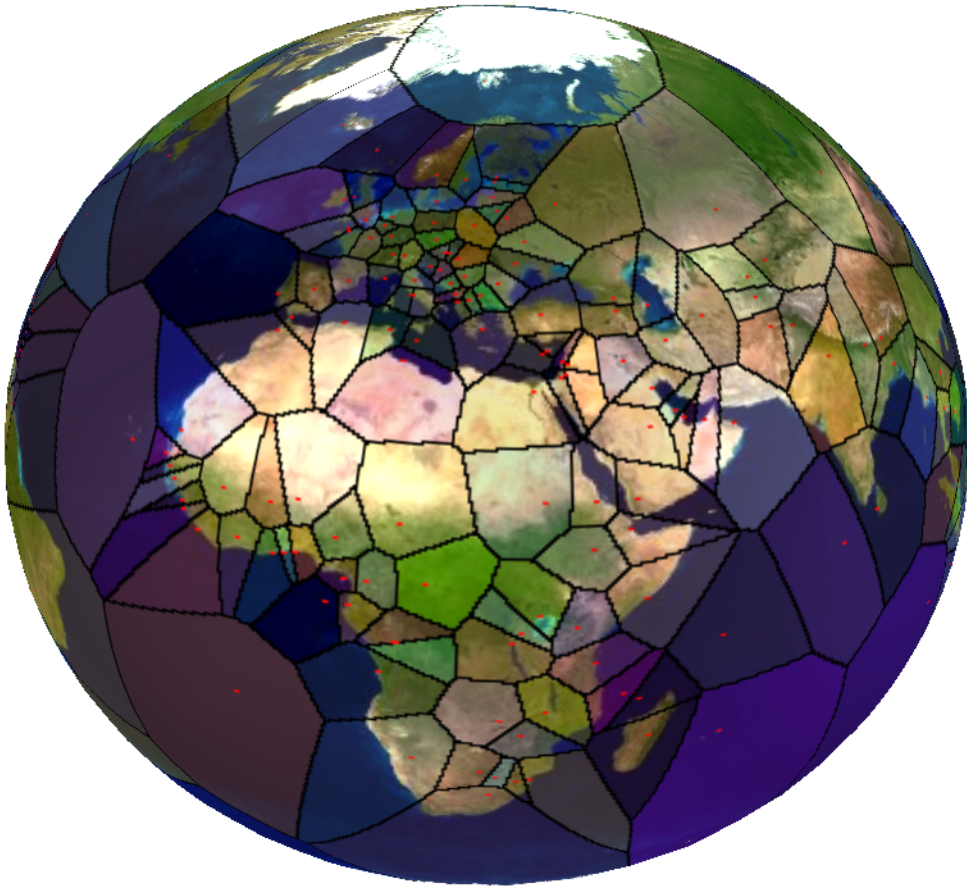


Figure 6: Voronoi diagram on the sphere

The following WebGL links show 3D models of the 3D Delaunay triangulations: Bunny [WebGL](#) and Icosahedron [WebGL](#).

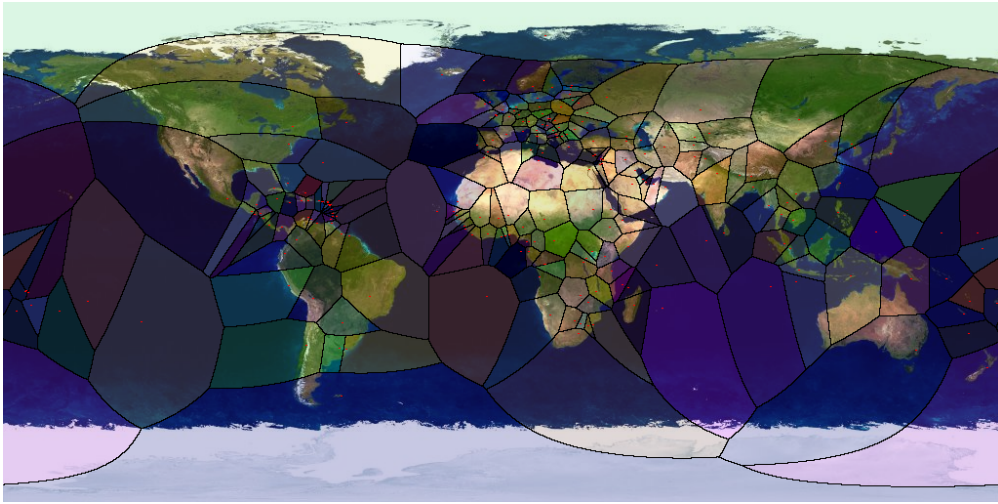
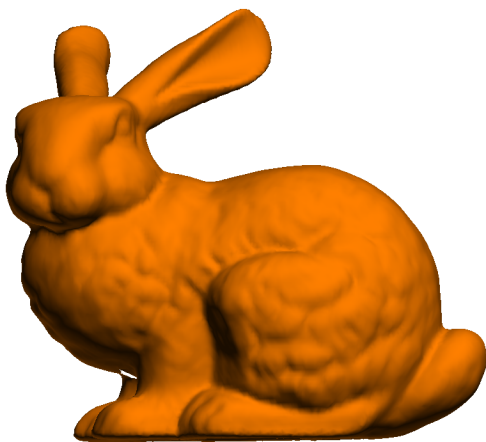
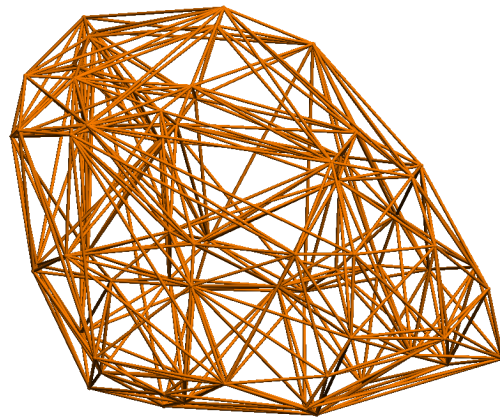


Figure 7: Voronoi diagram of the whole world

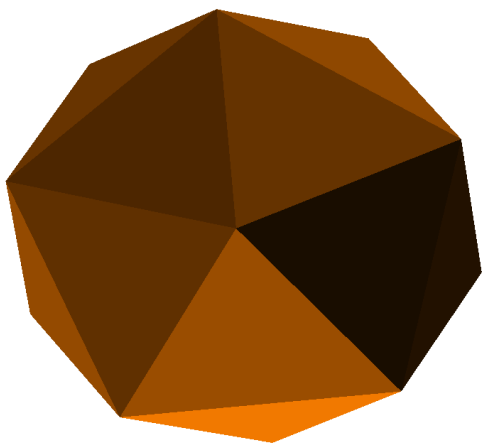


(a) Stanford Bunny

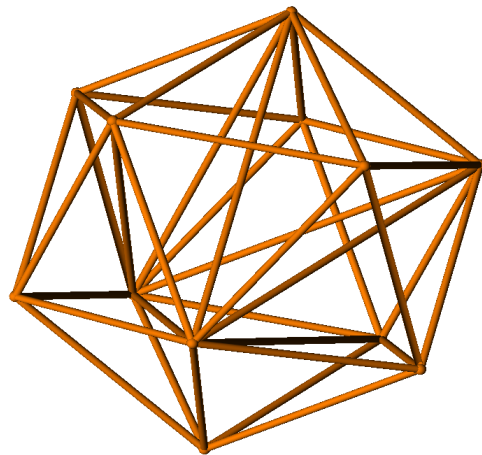


(b) 3D Delaunay triangulation of the bunny

Figure 8: Bunny 3D Delaunay Triangulation



(a) Icosahedron



(b) 3D Delaunay triangulation of an icosahedron

Figure 9: 3D Delaunay Triangulation of Icosahedron

3.3 JavaScript Image Linker



JaSIL is a compiler/linker which translates applications developed with Java to JavaScript. The currently used JDK is Apache Harmony. The optimized runtime image created by the linker does contain only the referenced classes, fields and methods. The JaSIL project offers the possibility to explore the usage of preemptive Java threads with JavaScript and their application in clients of Web-Applications.

JaSIL does translate Java bytecode to JavaScript and does therefore use a more general approach than similar projects which use special Java front-ends. The system does use the garbage collector of JavaScript for memory management.

Motivation for the Translation of Java to JavaScript

JavaScript is a functional language available on all popular current web browsers. In this way it is one of the most available and used platforms in the world. Java is a popular object oriented programming language with strong typing, which is available on many different platforms also. Very good IDE support allows developers to efficiently create and maintain complex Java applications. Using Java simplifies the development of complex web applications, but it is in general not enough to translate an existing Java program to JavaScript, because further special adaptations to the web browser environment are required.

Java/JVM Compatibility

JaSIL does provide options to provide less or more Java/JVM compatibility of the generated JavaScript code. Generating support for preemptive Java threads, including synchronization is therefore optional. These threads are realized in software and do not provide real hardware based multiprocessing. It is currently not intended to support dynamic class loading and runtime reflection of Java classes is limited. If threads are not required then jumps can be eliminated [10] optionally, to generate more efficient code. The performance of JavaScript code without jumps is around 11 % faster.

Example with Threads

Clicking on the link [Word Counting Demo](#) opens a dialog and starts a simple word counting task. It reads a file from the server, counts the words and displays the results in a table. Closing the dialog does terminate this task. A thread with low priority is used for this task.

Because threads are used, the GUI remains responsive, the window can be moved or minimized to the WEB-Dock and the table is constantly updated. Java programmers can use thread concepts they are already familiar with. The system does support threads with priorities and GUI activity can therefore slow down the word counting thread.

Example with WebGL

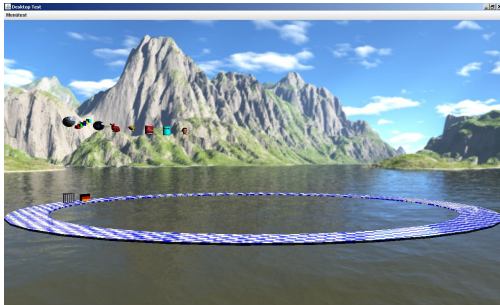
Clicking on the link [WebGL Demo](#) opens a dialog and starts a simple WebGL application. Threads are not used in this example, which means the application does run in the "GUI thread".

Related Projects

- [TeaVM](#)
- [Emscripten SDK](#)
- [Google Web Toolkit \(GWT\)](#)
- [Java2Script \(J2S\)](#)
- [Links: Web Programming Without Tiers](#)

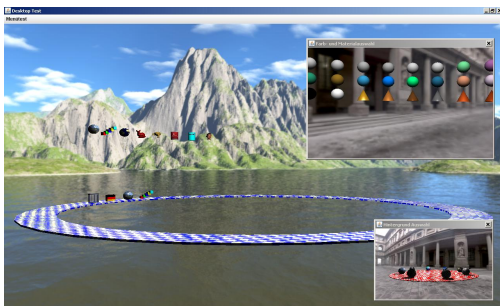
3.4 3D Desktop

Desktop with 3D Task List (Task-Dock)



The image shows the 3D task list with the Recycler and the Language Selection Box on it. Above the task list you can see a number of 3D icons. The viewer is located outside of the center of the desktop folder.

Desktop with Background and Color/Material Selection



The image shows the 3D desktop with 3D folders for background and color/material selection opened. The 3D icons of the opened folders are placed on the task list.

Background Selection View



The background selection folder shows a number of spheres with different cube maps projected on them. It is possible to change the background of a opened 3D folder by dropping one of these spheres on it.

Inside the 3D Desktop



The image shows the 3D Desktop with changed background and another 3D folder opened. The opened folder is selected in the 3D icon list. The viewer is located in the center of the desktop folder.

4 About the Author


Klaus Preschern has developed software components for Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) systems and host communication, used by banks and their computing centers. Furthermore he develops software in the area of compiler construction, runtime systems, computer aided design (CAD) and visualization.


He was a member of different development teams which performed ISO-9001 certified software life cycles with corresponding quality control. The programming languages used in these projects are mainly object oriented languages, like C/C++, C#, Smalltalk and Java.

His interests are centered around algorithms and data structures, programming languages, compilers and runtime systems, content management and web-applications, computer graphics and computational geometry, user interfaces and operating systems.


5 Dictionary of PDF Documents


preschern.org (German version): .

preschern.org (English version): .


Example of CSG operations with implicit surfaces: .


3D model of two blended implicit surfaces: .

Voronoi diagram on the sphere: .

Examples of 3D Delaunay triangulations: .

Examples of 3D reconstructions: .

An introduction to parsing in German: .

Modula-2 - C Translator Report in German: .

6 Disclaimer

6.1 Content

The author reserves the right not to be responsible for the topicality, correctness, completeness or quality of the information provided. Liability claims regarding damage caused by the use of any information provided, including any kind of information which is incomplete or incorrect, will therefore be rejected.

THIS CONTENT IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS CONTENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

All offers are not-binding and without obligation. Parts of the pages or the complete publication including all offers and information might be extended, changed or partly or completely deleted by the author without separate announcement.

6.2 Referrals and Links

The author is not responsible for any contents linked or referred to from his pages - unless he has full knowledge of illegal contents and would be able to prevent the visitors of his site from viewing those pages. If any damage occurs by the use of information presented there, only the author of the respective pages might be liable, not the one who has linked to these pages. Furthermore the author is not liable for any postings or messages published by users of discussion boards, guestbooks or mailinglists provided on his page.

6.3 Copyright

The author intended not to use any copyrighted material for the publication or, if not possible, to indicate the copyright of the respective object. All rights for any material created by the author are reserved. Any duplication or use of objects such as images, diagrams, sounds or texts in other electronic or printed publications is not permitted without the author's agreement.

6.4 Privacy policy

If the opportunity for the input of personal or business data (email addresses, name, addresses) is given, the input of these data takes place voluntarily. The use and payment of all offered services are permitted - if and so far technically possible and reasonable - without specification of any personal data or under specification of anonymized data or an alias. The use of published postal addresses, telephone or fax numbers and email addresses for marketing purposes is prohibited, offenders sending unwanted spam messages will be punished.

6.5 Legal validity of this disclaimer

This disclaimer is to be regarded as part of the Internet publication which you were referred from. If sections or individual terms of this statement are not legal or correct, the content or validity of the other parts remain uninfluenced by this fact.

7 References

- [1] Tomas Akenine-Möller and Eric Haines: *Real-Time Rendering*, 2nd edition, A.K. Peters, 2002, Pages 526-527.
- [2] Ken Arnold and James Gosling: *The Java TM Programming Language*, Addison-Wesley, 1996.
- [3] Christoph Baudson und Edgar Klein: *Berechnung und Visualisierung von Voronoi Diagrammen in 3D*, Diplomarbeit an der Rheinischen Friedrich-Wilhelms-Universität Bonn, 2006.
- [4] Günther Blaschek, Gustav Pomberger und Franz Ritzinger: *Einführung in die Programmierung mit Modula-2*, Springer Verlag, 1986.
- [5] James F. Blinn: *A Generalization of Algebraic Surface Drawing*, ACM Transactions on Graphics, Vol. 1, No. 3, July 1982, Pages 235-256.
- [6] Luca Cardelli, James Donahue, Lucille Glassman, Mick Jordan, Bill Kalsow and Greg Nelson: *Modula-3 Report (revised)*, Systems Research Center, Research Report 52, 1989.
- [7] Ecma International: *Standard ECMA-363: Universal 3D File Format*, Rue du Rhône 114, CH-1204 Geneva, 4th Edition, June 2007.
- [8] Herbert Edelsbrunner and Ernst Peter Mücke: *Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms*, ACM Transactions on Graphics, 9(1), 1990, Pages 66-104.
- [9] Herbert Edelsbrunner and Nimish R. Shah: *Incremental Topological Flipping Works for Regular Triangulations*, Algorithmica 15, 1996, Pages 223-241, First published in: Proceedings of the 8th Annual ACM Symposium on Computational Geometry, 1992, Pages 43-52.
- [10] Ana Maria Erosa: *A Goto-Elimination Method and its Implementation for the McCAT C Compiler*, Master Thesis, School of Computer Science, McGill University, Montreal, May 1995.
- [11] Paul Fishwick: *Exploring Multiple Visualization Perspectives with Aesthetic Computing*, International Conference for Visual Languages and Computing, 2003.

- [12] The Khronos 3D Formats Working Group: *glTF 2.0 Specification*, Khronos Group.
- [13] Leonidas Guibas and Jorge Stolfi: *Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams*, ACM Transactions on Graphics, Vol. 4, No. 2, Pages 74-123, April 1985.
- [14] Samuel P. Harbison: *Modula-3*, Prentice Hall, 1992.
- [15] Tao Ju, Frank Losasso, Scott Schaefer and Joe Warren: *Dual contouring of hermite data*, ACM Transactions on Graphics, Volume 21, Number 3, Pages 339–346, Proceedings of ACM SIGGRAPH July 2002.
- [16] Michael Kazhdan, Matthew Bolitho and Hugues Hoppe: *Poisson Surface Reconstruction*, Eurographics Symposium on Geometry Processing, 2006.
- [17] John Kessenich (Editor): *The OpenGL® Shading Language (Language Version 4.0)*, The Khronos Group Inc., 2010.
- [18] Donald E. Knuth: *The TeXbook*, Addison-Wesley, Reading, Massachusetts, 1986.
- [19] Leslie Lamport: *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, 2nd edition, 1994.
- [20] William E. Lorensen and Harvey E. Cline: *Marching cubes: A high resolution 3D surface reconstruction algorithm*, SIGGRAPH Computer Graphics, Volume 21, Number 4, Pages 163–169, July 1987.
- [21] Dani Lischinski: *Incremental Delaunay Triangulation*, Graphics Gems IV, Editor: Paul S. Heckbert, AP Professional (Academic Press), Boston, 1994.
- [22] Greg Nelson (Editor): *Systems Programming with Modula-3*, Prentice Hall Series in Innovative Technology, 1991.
- [23] Joseph O'Rourke: *Computational Geometry in C*, 2nd edition, Cambridge University Press, 1998.
- [24] Andreas Rossberg (Editor): *WebAssembly Specification*, Release 2.0, WebAssembly Community Group, 2023.
- [25] Mark Segal and Kurt Akeley: *The OpenGL® Graphics System: A Specification*, The Khronos Group Inc., 2010.
- [26] Greg Turk and Marc Levoy: *Zippered polygon meshes from range images*, In Proceedings of SIGGRAPH '94 (Orlando, FL, July 24-29, 1994), pages 311–318. ACM Press, July 1994.

- [27] Dean Jackson and Jeff Gilbert: *WebGL Specification 1.0 Editor's Draft*, Khronos Group.
- [28] Dean Jackson and Jeff Gilbert: *WebGL Specification 2.0 Editor's Draft*, Khronos Group.
- [29] National Imagery and Mapping Agency (NIMA): *Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems*, NIMA Technical Report TR8350.2, 3rd edition, Amendment 1, January 2000.
- [30] Niklaus Wirth: *Programmieren in Modula-2*, Springer Verlag (Übers. d. 3rd corrected edition), 1985.