**The Transvoxel™ Algorithm**

Der Transvoxel Algorithmus ist eine Methode für die nahtlose Integration von benachbarten Dreieck-Netzen, die von Voxel-Daten bei unterschiedlicher Genauigkeit generiert wurden, sodass ein „level of detail (LOD)“ mit überaus großen Voxel-Datenmengen wie volumetrischem Gelände in neuartigen Computerspielen angewandt werden kann.

Der Algorithmus wurde erfunden von Eric Lengyel in 2009 und in der C4 Game-Engine implementiert. Diese Website bietet eine qualitativ hochwertige Übersicht von dem Transvoxel Algorithmus, Links zu weiteren Informationen und Tabellendaten um eine Implementierung zu erleichtern.

**Patent status**

Der Transvoxel Algorithm steht nicht unter Patentansprüchen.

**The problem it solves**

Die Probleme welcher der Transvoxel Algrotihmus löst

Voxel basierte Gelände-System steigen in ihrer Anwendung, denn diese schränken die Topographie nicht wie bisherige konventionellen evaluations-basierenden System ein und bieten so die Möglichkeit noch komplexere Strukturen wie Höhlen, Überhänge und Bögen darzustellen. Dreiecks-Netze werden typischerweise von Voxel-Daten durch den Marching Cubes Algorithmus generiert. Die große Anzahl der Dreiecke in diesem Netz erzwingt eine Reduzierung dieser für die Verbesserung der Darstellungsgeschwindigkeit oder eher auch ein Detaillierungsgrad. Ein einfacher Weg wäre das Gelände mit unterschiedlichen Auflösungen generieren zu lassen, jedoch führt dies meist zu Löchern im Netz wie auch bei Verschmelzen unterschiedlicher Detailgrade zu unschönen, inkorrekte Verbundsdreiecke (siehe Abbildung 1).

Während elevation-basierende Gelände einfach „nachberarbeitet“ werden können, um diese Fehler auszubügel, ist dies nicht direkt effizient möglich mit Voxel-basierenden Geländen, denn das Entfernen der Fehler zwischen den Ecken und der Grundfläche ist meist ein deutlich komplizierterer Prozess.

Der Transvoxel Algorithmus präsentiert das Konzept der „Übergangs-Zellen“, um eine weiche und nahtlose Verbindung zwischen den unterschiedlichen Gelände-Netzen unabhängig der Auflösung zu ermöglichen. Der Algorithmus ist derartig konzepiert, dass lediglich lokale Voxel-Daten benötigt werden für die Umsetzung dieser „Übergangs-Zellen“ und erlaubt dadurch eine schnelle Neuberechnung für Echtzeit Applikationen mit sich oft ändernden Voxel-Daten (veränderliche).

|  |  |  |  |
| --- | --- | --- | --- |
| Multiresolution voxel terrain boundary artifacts | Multiresolution voxel terrain boundary artifacts wireframe | Multiresolution transvoxel stitching on voxel terrain | Multiresolution transvoxel stitching on voxel terrain wireframe |
| (a) | (b) | (c) | (d) |
| **Figure 1.** Triangle meshes for adjacent voxel terrain blocks are rendered at two different resolutions near a cave entrance. (a) Cracks are visible along the boundary between the blocks, allowing the sky in the background to show through, and a large hole appears where the terrain surface is nearly tangent to the boundary plane between the blocks. (b) A wireframe is overlaid to show the triangles generated by Marching Cubes. (c) The surfaces of differing resolutions are seamlessly joined by triangles generated for transition cells using the Transvoxel Algorithm, which follows in the footsteps of Marching Cubes but operates on an alternate arrangement of voxels at the boundary between cells of different resolutions. (d) The wireframe overlay reveals the triangles corresponding to the transition cells. | | | |

**How it works**



**Figure 2.** These are the 73 equivalence classes into which the 512 possible transition cell cases fall.

The Transvoxel Algorithm works by inserting special transition cells in between regular cells along the boundary between voxel data sampled at one resolution and voxel data sampled at exactly half that resolution. Instead of considering all possible combinations of voxel state for both the full-resolution and half-resolution data at a transition boundary, which would require that approximately 1.2 million cases be handled, we consider only nine samples of the high-resolution data. This gives us a much more manageable 512 cases to handle, and these cases happen to fall into the 73 equivalence classes shown in Figure 2. Every transition cell is filled with one of these triangle patterns in order to perfectly fill the seams, cracks, and holes that appear between meshes of different resolutions, as was done for the cave in Figure 1.

In Figure 2, black dots represent voxels that lie inside solid space, and corners without a dot represent voxels that lie outside in empty space. Green triangles are front-facing polygons, and red triangles are back-facing polygons.

**Look-up tables**

Like the Marching Cubes algorithm, the Transvoxel Algorithm is efficiently implemented using a set of look-up tables that provide information about each of the 512 possible cases that can arise. These tables are not easy to generate, so I am providing the tables that I made in the following C++ data file. There are comments in the file that describe what the numbers mean.

[Download look-up tables](http://transvoxel.org/Transvoxel.cpp) (Transvoxel.cpp, 60 kB)

**For more information**

* **Eric Lengyel’s Dissertation**, University of California at Davis, 2010. This dissertation goes into great detail about the Transvoxel Algorithm and multiresolution voxel terrain in general. It also describes how the algorithm can be efficiently implemented.

[Download PDF document](http://transvoxel.org/Lengyel-VoxelTerrain.pdf) (50.7 MB)

* Lengyel, Eric. “Transition Cells for Dynamic Multiresolution Marching Cubes”. *Journal of Graphics, GPU, and Game Tools*. Vol. 15, No. 2 (2010), A K Peters.  
  DOI: 10.1080/2151237X.2011.563682
* A complete implementation of the Transvoxel Algorithm is included with the [Tombstone Engine](http://tombstoneengine.com/).

**How to cite the Transvoxel Algorithm**

Please cite the dissertation in which this algorithm was originally described:

Lengyel, Eric. “Voxel-Based Terrain for Real-Time Virtual Simulations”. PhD diss., University of California at Davis, 2010.

Copyright © 2010–2016, Terathon Software LLC