Voxelek

Marching Tetrahedron módszerek

Gimbal lock problem

Koscsák Róbert

Tartalomjegyzék

Voxelek

Voxelek meghatározása (magyar)

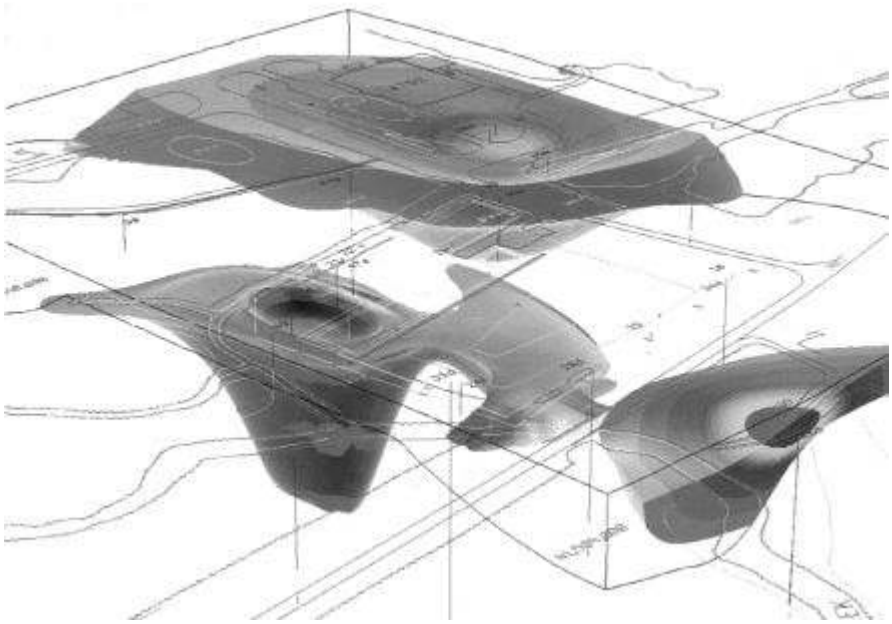
Elemi kockák. A három dimenziós képek elemi, mind a három tengely mentén kiterjedéssel bíró pontegysége. (volumetric pixel element) 3 dimenziós pixel.

1. forrás: Mi a Voxel? (magyar)

Három dimenziós raszter

Forrás: http://geo.efe.hu/hun/onlinejegyzet/geoinfo/geoinfo4.htm

A szabályos adatmodelleknek számos további változatuk van. Ezek közül az eddig tárgyalt két dimenziós raszterhez a térbeli raszter áll a legközelebb. A három dimenziós raszteres adatmodellt röviden voxelnek nevezik. A voxel előállítása is egy térbeli interpolációval indul. A teret felépítő elemi téglatestek csúcspontjaiban számítjuk a vizsgált térbeli adathalmaz értékeit. A voxeleken végzett elemzések, a voxel analízis, alkalmas a meteorológiai-, légköri folyamatok, a szennyező anyagok terjedésének, a talajvíz mozgásának térbeli tanulmányozására. A következő ábra egy voxel analízist mutat (szennyeződés terjedése a talajban):



2. forrás: Mi a Voxel? (angol)

Forrás: http://en.wikipedia.org/wiki/Voxel

A **voxel** (a portmanteau of the words *volumetric* and *pixel*) is a volume element, representing a value in three dimensional space. This is analogous to a pixel, which represents 2D image data. Voxels are frequently used in the visualisation and analysis of medical and scientific data. Some true 3D displays use voxels to describe their resolution. For example, a display might be able to show 512×512×512 voxels.

As with pixels, voxels themselves typically do not contain their position in space (their coordinates) - but rather, it is inferred based on their position relative to other voxels (i.e. their position in the data structure that makes up a single volume image).

Voxel data

There are two interpretations for a voxel value, depending on usage:

- A tiny cube with particular properties in a larger volume.
- A point sample in a regularly spaced 3D grid.

The value of a voxel may represent various properties. In CT scans, the values are Hounsfield units, giving the opacity of material to X-rays. Different types of value are acquired from MRI or ultrasound.

Voxels can contain multiple scalar values; in the case of ultrasound scans with B-mode and Doppler data, density, and volumetric flow rate are captured as separate channels of data relating to the same voxel positions.

Other values may be useful for immediate 3D rendering, such as a surface normal vector and color.

Uses

Visualization

A volume containing voxels can be visualised either by direct volume rendering or by the extraction of polygon iso-surfaces which follow the contours of given threshold values. The marching cubes algorithm is often used for iso-surface extraction, however other methods exist as well.

Computer gaming

Many NovaLogic games have used voxel-based rendering technology, including the Delta Force series.

3. forrás: Mi a Voxel? (angol)

Forrás: http://www.webopedia.com/TERM/V/voxel.html
Short for ***vo**lume **pixel***, the smallest distinguishable box-shaped part of a three-dimensional image.

Voxelization is the process of adding depth to an image using a set of cross-sectional images known as a *volumetric dataset*. These cross-sectional images (or slices) are made up of pixels. The space between any two pixels in one slice is referred to as interpixel distance, which represents a real-world distance. And, the distance between any two slices is referred to as interslice distance, which represents a real-world depth.

The dataset is processed when slices are stacked in computer memory based on interpixel and interslice distances to accurately reflect the real-world sampled volume.

Next, additional slices are created and inserted between the dataset's actual slices so that the entire volume is represented as one solid block of data.

Now that the dataset exists as a solid block of data, the pixels in each slice have taken on volume and are now voxels.

For a true 3D image, voxels must undergo opacity transformation. Opacity transformation gives voxels different opacity values. This is important when it is crucial to expose interior details of an image that would otherwise be hidden by darker more opaque outside-layer voxels.

Voxel images are primarily used in the field of medicine and are applied to X-Rays, CAT (Computed Axial Tomography) Scans, and MRIs (Magnetic Resonance Imaging) so professionals can obtain accurate 3D models of the human body.

Voxel imaging is also being used to create computer games, so 3D acceleration is not necessary.

4. forrás: Mi a Voxel? (angol)

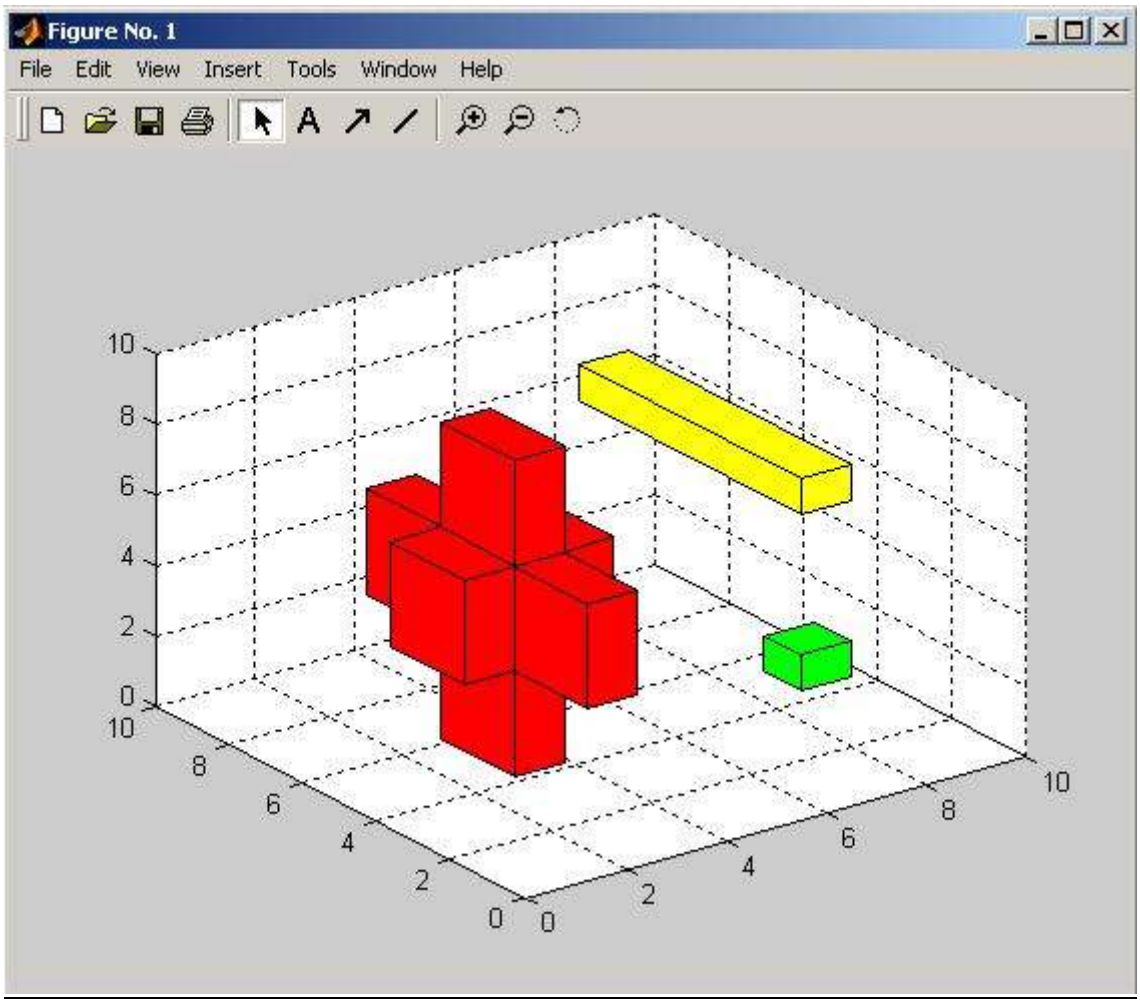
Forrás: http://www.pvrdev.com/pub/PC/eg/h/Voxel.htm
With the advent of Pixel Shader 3_0, graphics hardware has become capable of rendering hardware-accelerated voxels.

Voxel objects are stored as a three dimensional map of matter, with each voxel (or texel in a volume map) indicating something about that “lump” of matter – its colour, translucency, or “power” in the case of metaballs. In the “power” case, a threshold value is used; voxel values that are above (or below) this value are considered to be solid matter, with the rest considered to be empty space.

Typically, voxel objects are converted to polygons before rendering using the “marching cubes” algorithm, or similar. The method presented here submits a single eight-vertex cube and extracts the surface in the pixel shader; a ray is traced step by step through the volume, sampling the texture at each step, searching for matter.

5. forrás: Mi a Voxel? (angol)

Forrás:
http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=3280&objectType=file



Description: Simple function to draw a voxel (cube, cuboid) in a specific position of specific dimensions in a 3-D plot. Transparency of the voxel can also be specified.
Many voxels can be aggregated to form a different shapes (a simple 3-D "+" is shown in the screenshot).

6. forrás: Voxelek használata a grafikában példa (magyar)

Forrás: http://szteetr.uw.hu/
Részlet az Animáció anyagából

Landscape effekt

Mi az a Landscape effekt?

Egy 3D-s táj valós idejű (realtime) mozgatása. A 3D-s felületet mátrixba rendezett voxelekkel közelítjük.

Mi az a voxel?

A voxel olyan, mint a pixel, csak 3D esetén. Tehát míg egy területet egy pixellel közelítünk, aminek szélessége, és magassága van, addig a voxellel egy térfogatot, és mélysége is van. A voxel elnevezés egyébként a Volumized pixel-ből származik.

Landscape effekt:

Tehát a felület közelítése voxelek segítségével történik, mely voxelek szélessége és hossza megegyezik, csak a magasságuk változik. Ezért a mátrixban a magasságokat tároljuk. Ez a mátrix a Voxel Map. Matematikailag ez a Map a felületfüggvény diszkrét pontokban adott értékeit tárolja:
Map[x,y] = magassag(x,y).

E technika hátrány, hogy nagy felületek finom modelezéséhez rengeteg memória kell, mert ekkor nagy lesz a Voxel Map, sok magasságértéket kell benne tárolni.
Jelen példában a mátrix 256X256-os lesz, és a magasságértékek byte-ok formájában kerülnek tárolásra. Ez egyben a legegyszerűbb megoldás is, mert így real módban pont egy szegmens kell az eltárolásához, és a felület végtelenítése is megoldott.
Mivel a magasságon kívül a felület képét, azaz magát a textúrát is el kell tárolni, ezért szükség van még egy 256X256-os Map-re.
A felület kirajzolásánál most sem scanline-okat használunk, hanem oszlopról oszlopra építjük fel a képet, a képernyő aljáról indulva felfelé. Minden oszlop kirajzolásánál tudnunk kell, hogy hol járunk éppen a Map-ekben, és hogy milyen lépésközzel jutunk el a következő Voxelhez.
Tudjuk:
A kamera koordinátái: (X,Y)
A kamera elforgatási szöge: a
Az oszlop legalsó pixele a Voxel Map-en: (U,V)
A következő pixel által takart Voxel megkapásához szükséges lépésköz: (deltaU,deltaV)
Az egyszerűség kedvéért a kamera koordinátáit is a Voxel Map-re adjuk meg, ekkor: U=X és V=Y.
A lépésközök meghatározásához raycasting szükséges, ami a nézőpontból (kamera pozícióból) kiinduló sugarakon alapszik. A látóterünkben valahány fokonként sugarakat bocsátunk ki a nézőpontból, és csak a sugarak által érintett voxelek kerülnek ki a képernyőre.
A látótér szöge legyen b, ez a kamerával együtt a szöggel el van forgatva.
Ekkor az első kibocsátott sugár szöge: a + 270 - b/2 Az utolsóé: a + 270 + b/2

Marching tetrahedron módszerek

1. forrás: Leírás 1998 -ból (angol)

Forrás: *http://svr-www.eng.cam.ac.uk/reports/svr-ftp/treece_tr333.pdf*

Regularised marching tetrahedra:
improved iso-surface extraction
G.M. Treece, R.W. Prager and A.H. Gee
CUED/F-INFENG/TR 333
September 1998

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

Abstract

Marching cubes is a simple and popular method for extracting iso-surfaces from implicit functions or discrete three-dimensional (3-D) data. However, it does not guarantee the surface to be topologically consistent with the data, and it creates triangulations which contain many triangles of poor aspect ratio. *Marching tetrahedra* is a variation of marching cubes, which overcomes this topological problem. Improvement in triangle aspect ratio has generally been achieved by *mesh simplification*, a group of algorithms designed to reduce the large number of triangles. *Vertex clustering* is one of the simplest of these algorithms, but does not in general maintain the topology of the original mesh. We present a new algorithm, *regularised marching tetrahedra*, which combines marching tetrahedra and vertex clustering to generate iso-surfaces which are topologically consistent with the data and contain a number of triangles appropriate to the sampling resolution (typically 70% fewer than marching tetrahedra) with significantly improved aspect ratios. This improvement in aspect ratio greatly enhances the display of the surface, particularly when it is rendered using simple interpolated shading. Surface triangulations are shown for implicit surfaces, thresholded medical data, and surfaces created from object cross-sections. The application to data sampled on non-parallel planes is also considered.

Contents

2. forrás: Leírás 2001 –ből (francia)

Forrás: *http://www-evasion.imag.fr/Membres/Fabien.Vivodtzev/doc/rapportMag2.pdf*

Rapport de stage de Maîtrise d'Informatique

Fabien Vivodtzev

Visualisation d'un volume de données généré à partir de simulations d'un fluide dynamique.

Stage effectué au Laboratoire :
CIPIC de l'Université de Californie à Davis

Sommaire

Glimbal lock problem

1. forrás: Könyv (magyar)

Nyisztor Károly: Grafika és Játékprogramozás DiretX -szel

A könyv két fő részre tagolódik.
Az első fele elméleti részre és a könyv második fele szinte kizárólag gyakorlati know-how, kifejezetten a DirectX-szel történő grafikai és játékfejlesztést mutatja be.

A könyv tartalomjegyzéke és leírása megtalálható az író weblapján:
http://nkari.uw.hu/dxkonyv.php

2. forrás: Gimbal lock (angol)

Forrás: http://en.wikipedia.org/wiki/Gimbal_lock

In gyroscopic devices controlled by Euler mechanics or Euler angles, gimbal lock is caused by the alignment of two of the three gimbals together so that one of the rotation references (pitch/yaw/roll, often yaw) is cancelled. This would require a reset of the gimbals using an outside reference. It may also be described as the situation when all three gyros hit the limits of their ability to move within the sensing mechanism - they hit hard stops and stop moving around.

For example, an airplane uses three references, pitch (angle up/down), yaw (angle left/right on a vertical axis) and roll (angle left/right on the horizontal axis). If an airplane heads straight up or down (change of pitch), one other reference (the yaw) is cancelled, one loses a dimension of rotation, because there is always a value for one angle of rotation that yields infinite values of the other two angles (in this case, the yaw). A solution to this problem is the implementation of an extra gimbal in the INS-platform. This reduces the statistical chance of gimbal lock to almost zero.

Compare a similar problem with tangents on triangles -- say one has a right triangle ABC, with angle ACB=90°. Consider angle BAC. tan(BAC) = BC/AC, but BAC is less than 90°. We can make angle BAC closer and closer to 90 degrees by increasing the length of BC, but as we keep doing this, AC stays the same so the ratio BC/AC gets infinitely large. So tan(90°) has no geometric meaning. Two legs of the triangle become infinitely long and never meet one another.
Another real world comparison is latitude and longitude. At the poles (latitude 90° north or south), the definition of longitude becomes meaningless (as all longitude lines meet at a point or singularity).

3. forrás: Gimbal lock (angol)

Forrás: http://www.anticz.com/eularqua.htm

What is Gimbal Lock and why does it occur?

What is gimbal lock? Gimbal lock is the phenomenon of two rotational axis of an object pointing in the same direction. Simply put, it means your object won't rotate how you think it ought to rotate. Gimbal lock is frustrating problem that every CG artist will face sometime in their career and it always happens at the worst possible time. Gimbal lock occurs when animating an object with a rotational matrix known as Eular (pronounced Oiler) angles. It's a general limitation of that type of rotational matrix.

Any system that uses Eular angles (Maya, Max, Lightwave, Softimage) will have problems with gimbal lock. The reason for this is that Eular angles evaluate each axis independently in a set order. In the case of 3DS Max that order is generally X,Y,Z (you can change the order to whatever you'd like however) meaning ... first the object travels down the X axis. When that operation is complete it then travels down the Y axis, and finally the Z axis. The problem with gimbal lock occurs when you rotate the object down the Y axis, say 90 degrees. Since the X component has already been evaluated it doesn't get carried along with the other two axis. What winds up happening is the X and Z axis get pointed down the same axis.

Here's a sample of that phenomenon in 3DS Max. If you download and open the Gimbal lock.max file, select the dummy and rotate it down it's local Y axis 90 degrees you will actually see it's matrix gimbal lock. If your going to use Eular angles (and every 3D package has them somewhere) your going to have problems with gimbal lock. The only advantage to using Eular angles is that they are much easier for an artist to read and conceptualize than a **Quaternion Rotation.**

Quaternion Rotations are far more powerful and robust. In Max, TCB controllers use Quaternion math to find their orientation. Quat rotations evaluate all three axis at the same time to find a direction to travel and a fourth value (the w component or up vector) to tell the matrix how far to travel. The advantage to using Quat rotation is that you never have to worry about Gimbal Lock. It simply can not happen because all three axis get updated at the same time. The down side to using Quats is they are far more complicated to read and conceptualize than an Eular angle. RotationScript controllers in Max are Quat rotations which makes them far more powerful than Eular angles and expressions but they are also much more difficult to use and understand.

I generally chose to use Eular angles on my puppets because they are easy to understand and animate with. You just have to keep in mind that an Eular angle, no matter what you call it (Local Eular, World Eular, XYZ Bezier) is still an Eular angle and as such, has severe limitations.

A few things to keep in mind when working with Eular angles to keep you from going Postal:

1. If your object only needs to rotate along one axis, change the order of axis to match i.e.. if your rotating down the Y axis make the order of axis Y,Z,X.
2. If your rotating down two axis, change the order of axis to correspond with the first and last axis i.e... if your rotating down the X and Y axis, change the order of axis to X,Z,Y
3. If your rotating down all three axis be aware that there will be areas where gimbal lock will occur. Try to place those areas far enough out of the way that they won't be a problem or separate the axis with an additional node (even this isn't enough sometimes)

4. In certain specific cases, the parent of an object can actually gimbal lock it's child (I won't go into the math involved here) Try placing an extra Quat node in the chain between these two nodes. You can see a good example of this in 3DS Max if you link anything using Eular angles to a Biped. Instant Gimbal Lock. One workaround is to link a dummy with TCB controllers (Quat) to the biped first, then link your chain to the dummy.
5. The first axis evaluated will always happen in the objects local coordinate orientation
6. The last axis evaluated will generally happen in a world coordinate orientation

With careful planning and a little fore thought you'll be able to keep gimbal lock to a minimum. Happy animating.