Angkor - Geländevisualisierung

Jens Schöbel

Visualization and Numerical Geometry Group
Universität Heidelberg

24. Juni 2005

Einführung 1/2

Eine einfache und weit verbreitete Art der Visualisierung von großflächigem Gelände besteht in der Visualisierung mit einer Heightmap.

Heightmap: Eine Heightmap ist ein beliebiges Datenformat, welches in einem rechteckigen Gebiet G zu gegebenen endlichen Koordinaten

$$(x,y) \in G$$

die Höhe

$$h(x,y)$$

in diesem Punkt liefert.

Einführung 2/2

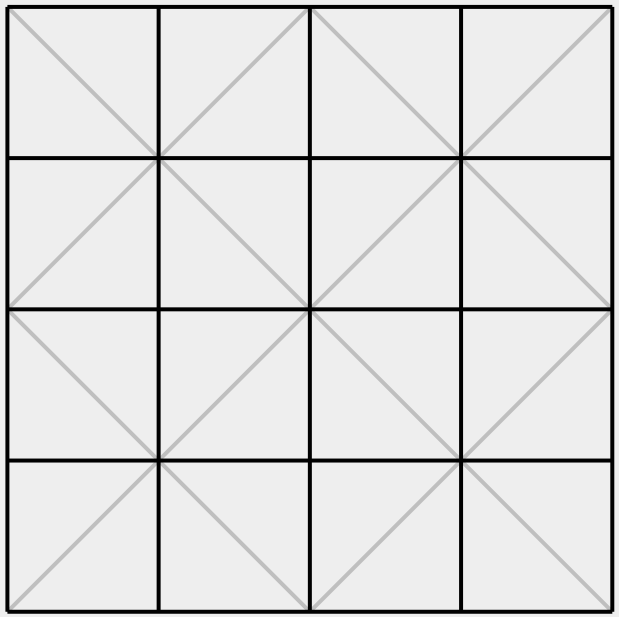
Der in diesem Vortrag vorgestellte Algorithmus verwendet ein quadratisches Gebiet der Länge $2^i + 1$ mit $i \in \mathbb{N}$. Als Datenformat wird ein Bitmap verwendet. An den Koordinaten (x, y) des Bitmaps gibt die Farbe des Pixels die Höhe an.

In der Praxis hat es sich bewährt, nicht die komplette Farbpalette zu verwenden. Meist werden zum Sparen von Speicherplatz nur Graustufen im Grafikformat verwendet. Man spricht in solchem Fall auch von **grayscale maps**.

Ich habe mich für ein Bitmap entschieden, da es für jeden Pixel die Daten ohne Verlust an Informationen speichert. Dankbar wären allerdings auch andere Formate wie *PNG*, *TIF*. Beide speichern ebenfalls verlustfrei. Möglich wäre es auch, ein eigenes Datenformat zu verwenden.

Quadtree 1/4

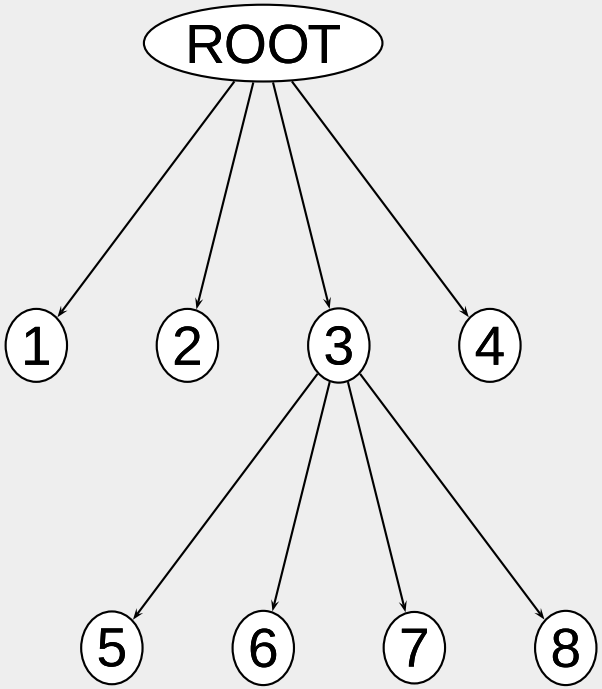
Wollte man alle Dreiecke zeichnen, die das Gelände darstellen, so würde man sehr schnell eine viel zu hohe Datenlast haben. Bei einer Seitenlänge von $2^i + 1$ müßte man 2^{2i+1} Dreiecke zeichnen.

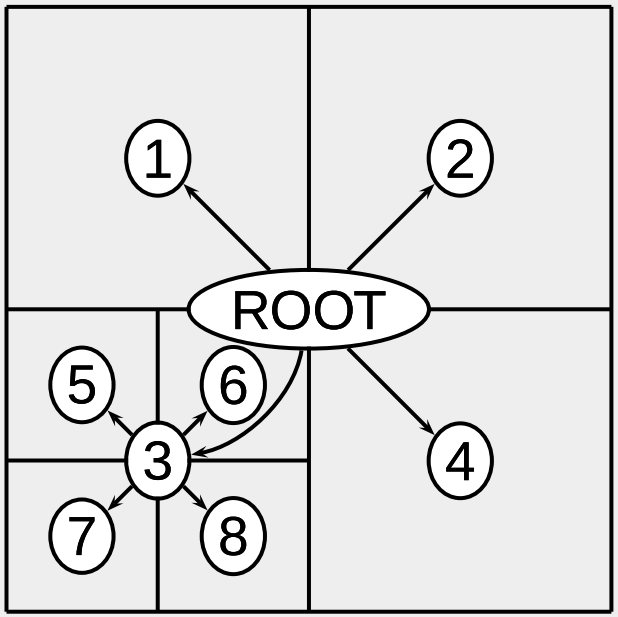


Im Beispiel ist $i = 2$, somit müssen also 32 Dreiecke gezeichnet werden.

Quadtree 2/4

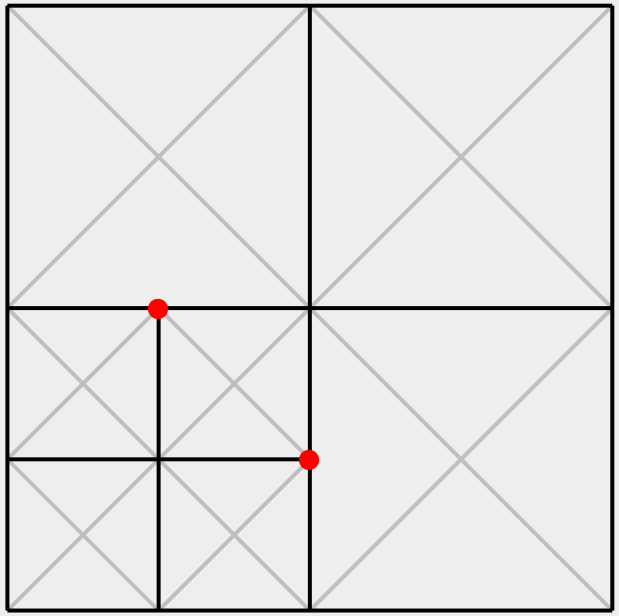
Quadtree: Ein Quadtree ist eine Baumstruktur, in der jeder Knoten außer den Blättern (*leafs*) genau vier Kindknoten besitzt.





Quadtree 3/4 - Cracks

Teilt man das Terrain in einen Quadtree wie oben, so kann es zu so genannten **cracks** kommen, das sind „Löcher” zwischen zwei angrenzenden Quads (rot markiert). Im Bild ergänzen die grauen Linien das zugrunde liegende Drahtgittermodel (*mesh*).



Quadtree 4/4

Der Vorteil eines Quadtrees besteht darin, die kleineren Gebiete (*quads*) nur so weit teilen zu müssen, bis sie gewissen Bedingungen genügen.

Im folgenden Algorithmus wird diese Bedingung ein sog. *Pixelfehler* sein. Da durch die Struktur eines Quadtrees einige Höhendaten nicht beachtet werden, wird es unter Umständen zu Abweichungen vom Original kommen. Damit diese nicht zu sehr auffallen, wird der Pixelfehler als Maß für die Abweichung dienen.

Beidem (dem Vermeiden von Cracks und einer möglichst guten Annäherung an das originale Mesh) wird der SOAR - Algorithmus Rechnung tragen.

SOAR 1/6

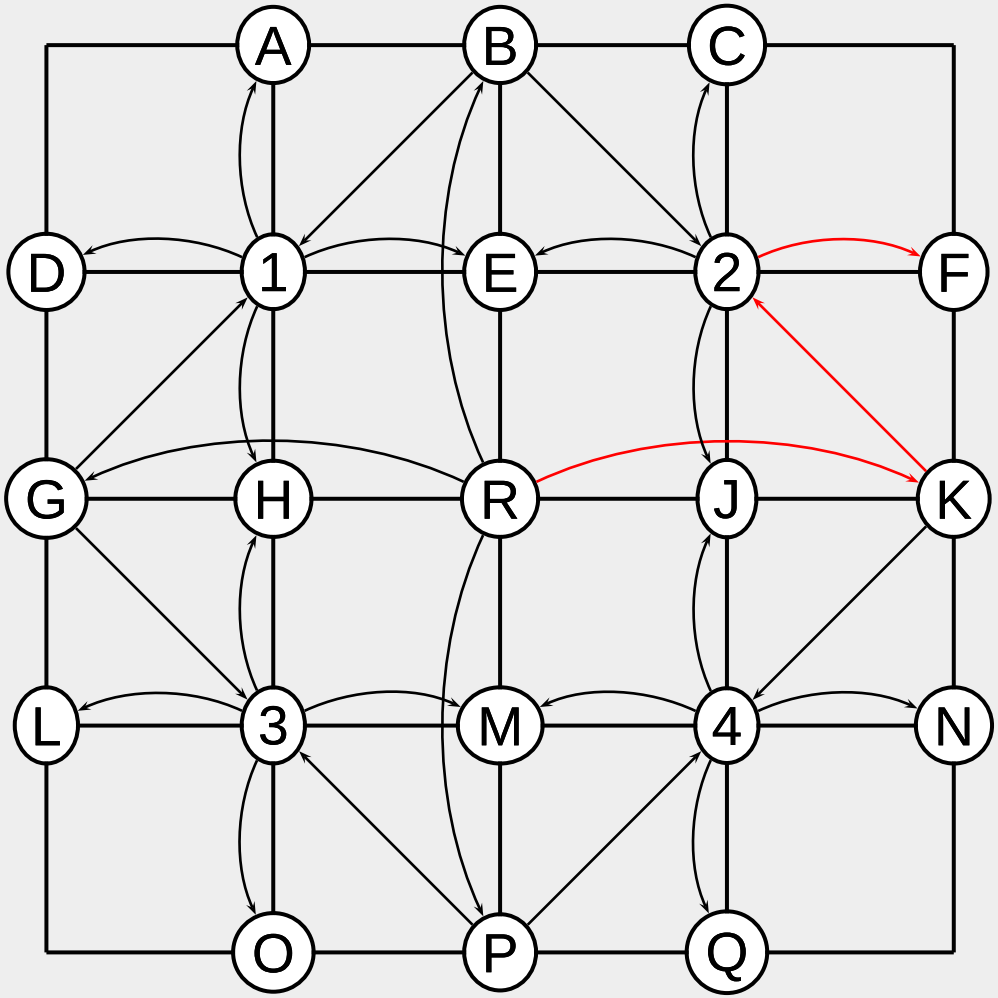
Zur Vermeidung der oben angesprochenen Cracks, wird die Struktur des Quadtrees leicht geändert. Als zusätzliche Bedingung soll nun gelten:

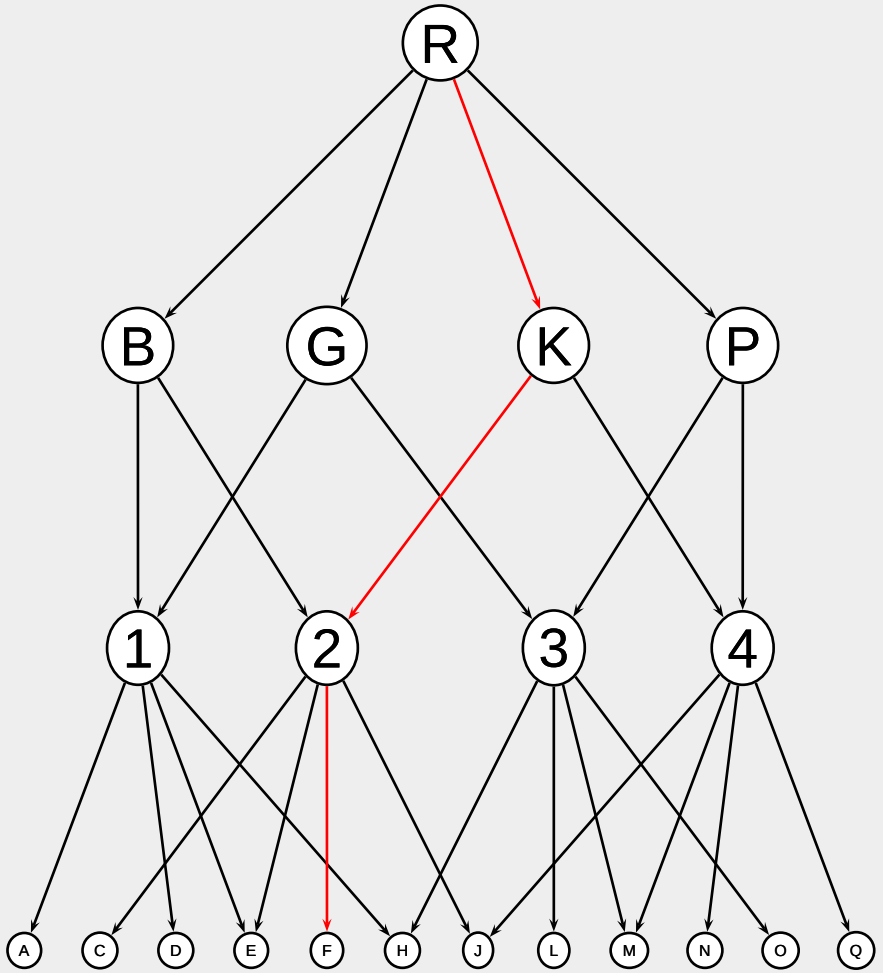
Wird in ein Quad geteilt, so soll auch der Quad des zugehörigen Elternknotens geteilt werden.

Dies allein würde allerdings noch nicht die Problematik der Cracks beheben. Es werden zusätzliche Knoten in unserem Baum eingefügt. Die eingefügten Knoten befinden sich zwischen den Ebenen der Quads und repräsentieren die Kante von jeweils zwei adjazenten Quads.

Etwas anschaulicher wird dies sicherlich anhand folgender Abbildung.

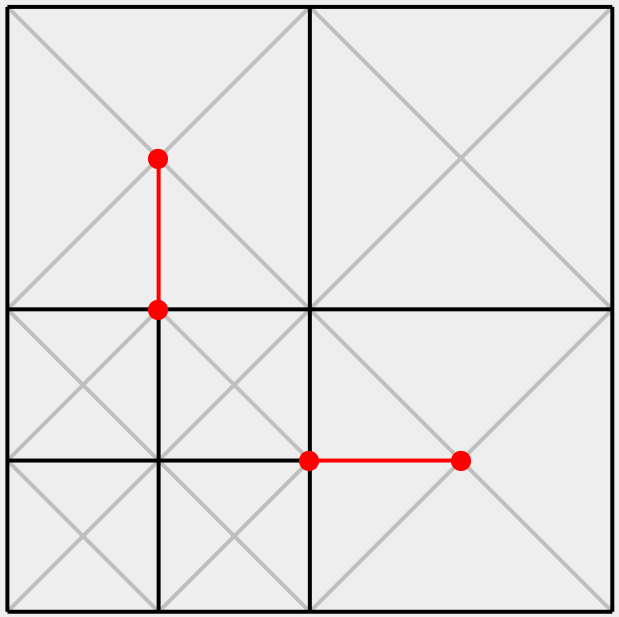
SOAR 2/6





SOAR 3/6

An obigen Beispiel verdeutlicht müßten also die rot markierten Knoten geteilt werden. Dadurch ergeben sich neue Dreiecke, deren Kanten durch die roten Linien markiert sind.



Auf diese Weise werden Cracks wirkungsvoll verhindert.

SOAR 4/6

Diese Vorgehensweise entspricht einem *bottom-up* Prinzip. Allerdings ist diese Vorgehensweise nicht erwünscht, da dies hieße, alle Quads zu betrachten.

Idealer wäre es einen *top-down* Ansatz zu verfolgen. Hier hieße dies, vom Wurzel-Knoten ausgehend die Quads so lange zu unterteilen, bis eine Unterteilung nicht mehr nötig ist. Dies läßt sich erreichen durch die Einführung des oben angesprochenen Pixelfehlers. Dazu stellt man als Bedingung an die Knoten, daß der Fehler eines Elternknoten größer/gleich aller Fehler der Kindknoten ist.
Solange dieser Fehler größer als ein fest vorgegebener Wert ist, teilt man das entsprechende Quad.

Die Berechnung des Fehlers setzt sich dabei aus zwei Teilen zusammen.

SOAR 5/6

- *Abweichung*:
  Die Abweichung gibt den Fehler zwischen einem Quad und dem original Mesh an. Sie wird für **jeden** Quad in einem *preprocessing*-Schritt berechnet. Beginnend bei den leafs errechnet man den Fehler und stellt sicher, daß dieser monoton wachsend zum nächsten Elternknoten ist.
  Dieser Schritt ist sehr rechenintensiv.
- *Entfernung*:
  Damit ist die Entfernung zwischen dem aktuellen Kamerapunkt und den Koordinaten des aktuellen Knoten gemeint.
  Wegen des *top-down* Schrittes müssen nicht sehr viele dieser Berechnungen durchgeführt werden.
- *Pixelfehler*:
  Er ist die Kombination beider Fehler.

SOAR 6/6

Auf die genaue Berechnung dieser Fehler soll hier nicht eingegangen werden.

In meiner Implementierung habe ich ausgehend von den leafs die Abweichung als das Volumen zwischen Original und Quad betrachtet.

Denkbar wäre auch die maximale oder die mittlere Abweichung zu verwenden. Prinzipiell ist die Wahl der Berechnung der Abweichung zweitrangig, muß aber obigen Bedingungen genügen. Eine „gute” Berechnung aller Fehler kann allerdings den Algorithmus enorm beschleunigen, da u.U. ein großer Teilbaum aus der Berechnung genommen werden kann.

Höhendaten 1/2

Wie anfangs erklärt, basiert dieser Algorithmus auf einer Heightmap. Interessant in diesem Zusammenhang ist die Frage, wo man Höhendaten erhalten kann.

- `http://srtm.usgs.gov/data/obtainingdata.html`
  Hier kann man sich zu fast allen Orten der Welt eine mehr oder weniger detaillierte Karte mit Höheninformationen beschaffen. Kambodscha mit dem Angkor Gelände ist allerdings nur mit einer Auflösung von ca. 1800 Metern erhältlich.
- `http://www.vterrain.org`
  Diese Seit ist eine sehr gute Anlaufstelle, wenn es um Visualisierung von Terrain geht. Hier gibt es Links zu Papern wie dem SOAR Algorithmus, Berechnung von Wasser, Wolken und anderer Umgebung.
  Über diese Seite gelangt man auch zu anderen Homepages, in denen es Höhendaten gibt.

Höhendaten 2/2

- ArcView
  ist ein Programm zum Betrachten und Bearbeiten von Geographischen Daten. Darunter fallen auch Höhendaten.
  Aus Kambodscha haben wir Vektordaten, die mit ArcView gelesen werden können, mitgebracht. Diese Daten decken einen Großteil von Kambodscha inklusive dem Angkor-Gebiet ab.

Texturierung

Zusätzlich ist es möglich, Texturen auf des Gelände zu projizieren. Dadurch wird eine bessere Orientierung im 3D-Raum erreicht.

Aus Kambodscha haben wir 167 Satellitenbilder mitgebracht. Zusammengefügt und für den Algorithmus aufbereitet wird ein Gelände von mehr als 400 km^2 abgedeckt.

Bibliotheken

Das Programm `SR_main` benötigt folgende Bibliotheken:

- SDL
  Die **S**imple **D**irectmedia **L**ayer Bibliothek ist unter `http://www.libsdl.org/index.php` erhältlich. Dies ist eine Open-Source Bibliothek unter der GNU LGPL Lizenz. Sie erleichtert die Arbeit mit Input-Events wie Maus-, Tastatur-, Joystick- usw. Eingaben.
- OpenGL
  OpenGL ist eine 3D Grafik Bibliothek, die es schon seit Jahren für alle gängigen Betriebssysteme gibt. Sehr reichliche Informationen gibt es unter `http://www.opengl.org/`.

Literatur

[1] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust and G. Turner. Real-time continous level of detail rendering of height fields. *Proceedings of SIGGRAPH' 95*

[2] P. Lindstrom and V. Pascucci. Visualization of large terrains made easy. *IEEE Visualization 2001 Proceedings*