Angkor - Realtime Visualisation with a game engine

Jens Schöbel

Visualization and Numerical Geometry Group
University Heidelberg

9th of September 2005

Introduction 1/2

Nowadays most 3D games are using highly optimized engines, in which engine describes the underlying program, that is responsible for calculating the graphic, the physic, gamehandling, data input etc.

With the focus on high framerates, the newest engines are highly optimized to guarantee high performance even on old computers. So, using game engines in a scientific way could open new possibilities for visualization.

3D games are mainly developed for showing virtual worlds in realtime. For our project this means that such engines can be used to show people the virtual world of Angkor. People can move around and have a look to different temples and places in an interactive way.

Besides all the positive things there are some disadvantages.

Introduction 2/2

Game engines are designed for realtime interaction, i.e. these engines can not simulate a correct world. Using heuristics creates fakes close to it. Usual games don’t need a perfect simulation and mostly it is not possible.

Calculating a jetstream takes several hours - sometimes days - on a cluster of supercomputers. Game engines are designed to run on home computers. This makes it impossible to use a game engine for a model of a temple with millions of polygons, or calculating the statical problem of a temple.

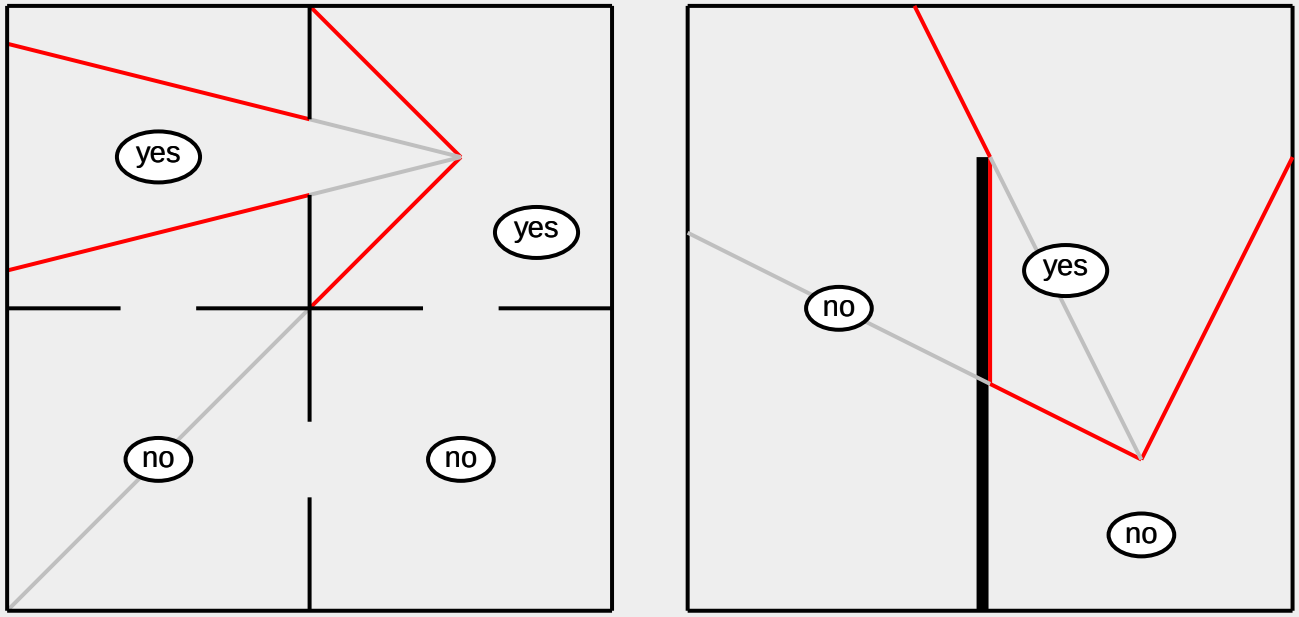
Game engines provide the possibility of realtime visualization, but with (very) few details. In the future with the development of new hardware and software, it could be possible to realize much more than today.

Optimization Techniques 1/2

Normally the computer doesn’t know anything. He is always doing what we tell him to do. In computer graphics this means that he would draw every object. Different techniques ensure that only the visible objects will be drawn.

- *Portals*:
  The virtual space is devided into smaller parts. The engine determines in which zone the camera is and calculates all zones visible from the current one. A visible zone will be marked and checked if another zone is visible from it. Portals are separating different zones from each other.
- *Antiportals*:
  are important for occlusion culling. Objects behind another object don’t need to be drawn. Imagine some trees, that you can not see, behind a big wall.
- *Spatial trees*:
  is a tree structure for hierachical rendering of a scene. The virtual space is devided into smaller parts which are arranged in a tree.

Optimization Techniques 2/2



The pictures represent portals (left) and antiportals (right). In some cases these techniques are named differenty, but they mean the same.

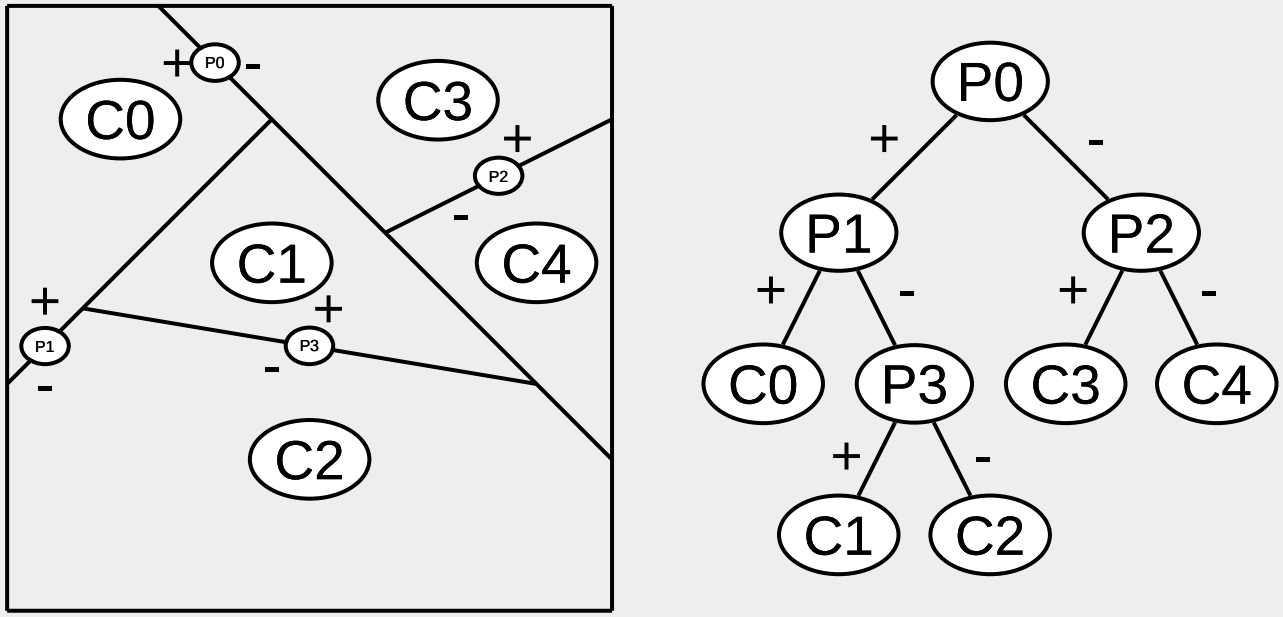
The nodes show wich parts need to be drawn and wich not.

Spatial Trees - BSP 1/5

Spatial trees can be used to partition the world into cells. Famous in terrain rendering are quadtrees and octtrees. The The structure introduced here will be the BSP - Binary Space Partitioning - Tree.

A **Binary Space Partitioning (BSP) tree** is a standard binary tree used to sort and search for polytopes in a n-dimensional space. The tree taken as a whole represents the entire space, and each node in the tree represents a convex subspace. Each node stores a ”hyperplane”which divides the space it represents into two halves, and references to two nodes which represent each half. In addition, each node may store one or more polytopes.

Spatial Trees - BSP 2/5



BSP tree partition in $\mathbb{R}^2$

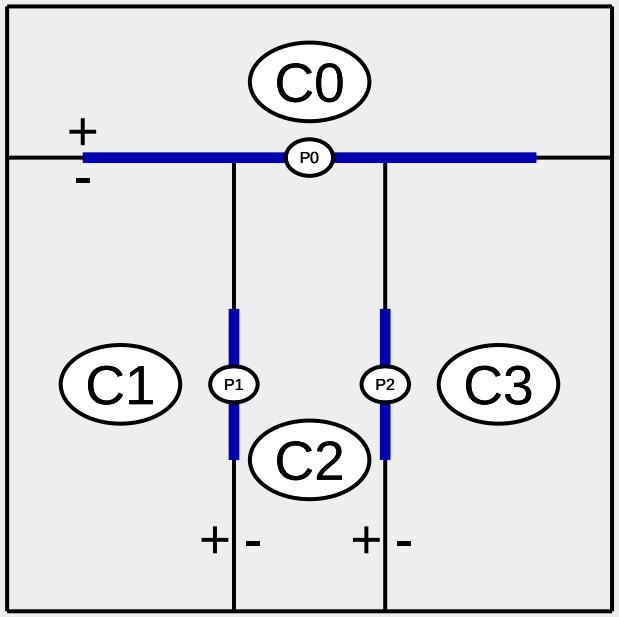
Spatial Trees - BSP 3/5

BSP trees provide an efficient method for sorting polygons by depth-first traversal of the tree. The price for sorting is that polygons have to be split during the process.

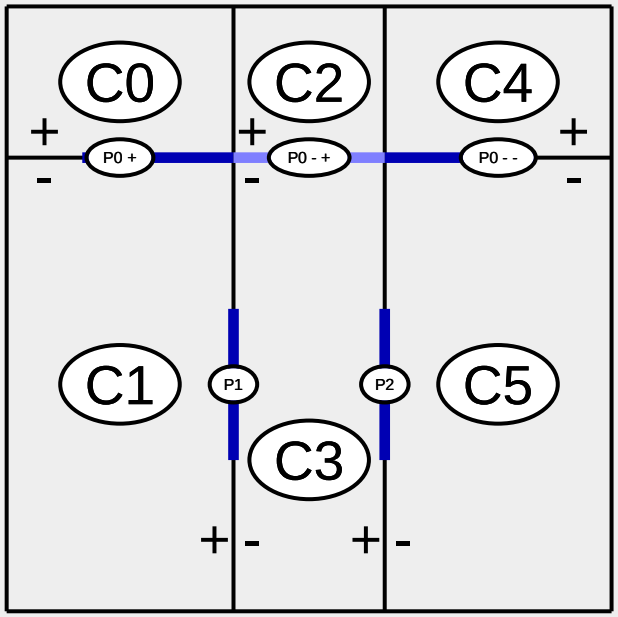
With a given eye point it can be determined which parts of the space are visible from the current view point.

Spatial Trees - BSP 4/5

order: P0, P1, P2



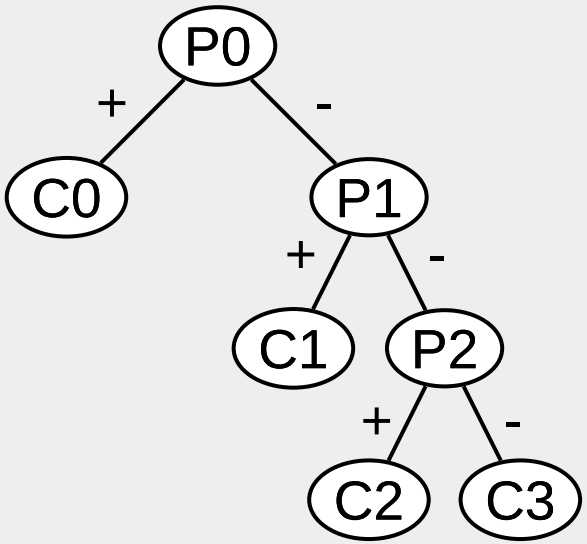
order: P1, P2, P0



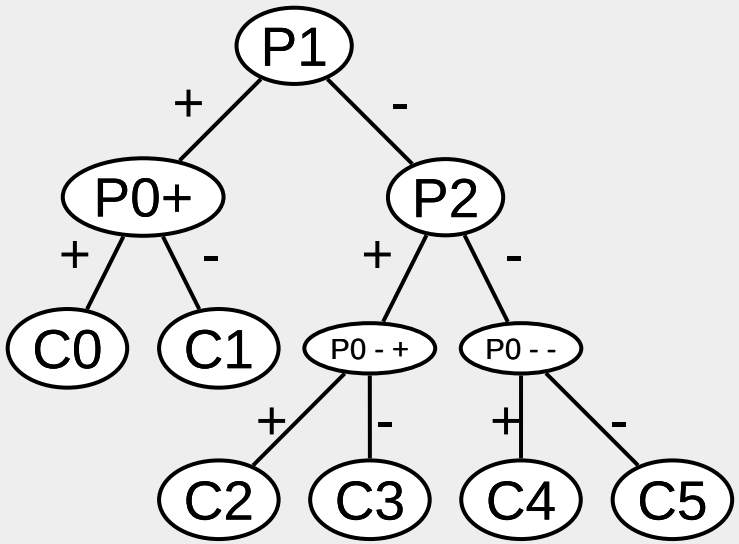
The pictures show different orders of the cutting planes. The blue lines represent the geometry from wich the cutting planes are determined.

Spatial Trees - BSP 5/5

order: P0, P1, P2



order: P1, P2, P0



The picture shows different trees for the same static geometry. The efficiency of a BSP-tree depends on the order of the cutting planes.

Literatur

[1] Fuchs, H., Z. Kedem, and B. Naylor. 1979. Predetermining visibility priority in 3-D scenes. *Proceedings of SIGGRAPH 1979, pp. 175-181*

[2] Fuchs, H., Z. Kedem, and B. Naylor. 1980. On visible surface generation by *a priori* tree structures. *Proceedings of SIGGRAPH 1980, pp. 124-133*

[3] BSP FAQ, *http://www.faqs.org/faqs/graphics/bsptree-faq/*

[4] David H. Eberly, *3D Game Engine Design*

[5] Epic Games, Digital Extremes, Atari - Unreal Tournament 2004 *http://udn.epicgames.com*