**SYNONYMS**

Binary Space Partitioning Tree

**DEFINITION**

Binary space partitioning (BSP) is a technique for recursively subdividing a space into convex sets by hyperplanes. The subdivision can be represented by means of a tree data structure known as a BSP Tree. A BSP Tree is thus a Point Access Method. Each node in a BSP Tree represents a hyperplane which divides the space into two halves. It also contains references to two nodes which represent half-spaces.

![Figure 1: Example of a BSP Tree](image)

**HISTORICAL BACKGROUND**

The BSP Tree came out of the collaborative work of Henry Fuchs, Zvi Kedem and Bruce Naylor. The work on BSP Trees was also part of Naylor’s doctorate thesis. In 1979 published a paper [6] in the SIGGRAPH’79 conference that let to BSP Trees. The BSP Tree was described for the first time in a paper [7] by the same three people in the next SIGGRAPH conference in 1980.

**SCIENTIFIC FUNDAMENTALS**

In order to illustrate how BSP Trees work consider the example of partitioning a cube shown in Figure 2 which illustrates how a BSP Tree can be constructed for a three dimensional object. For simplification only the lines parallel to the axes will be considered and the space will be divided into two at equal subspaces at each split. Consider a cube in the XYZ plane such that its sides are parallel to the X, Y and Z axes.
For the first split, which is also the root of the axis, the tree is split in half at the center along the XZ-axis. The two children of the root node correspond to the two subspaces. The two sides are then further split into two by choosing a line which is opposite in orientation to the one which was chosen previously. Thus the new split is along the XY-axis. This results in addition of two nodes along the left subtree for node Y. The process of splitting the polygon, cube in this case, is applied recursively until a stopping criteria is reached.

Figure 2: Construction of a BSP Tree

The construction of a BSP Tree can be explained as follows: Consider a set of polygons in three dimensional space and the goal is to build a BSP Tree which contains all the polygons. First select a partition plane and then partition the set of polygons with the plane. Recursively apply the first two steps on the new partitions of the set. In three dimensional space planes are used to partition the space, while in two dimensional space lines are used to obtain the sub-spaces. There is no universal criteria for choosing the hyperplane for partition but rather the criteria varies from application to application. A front/back test is performed for the partitioning of polygons i.e., each point in the polygon is tested against the plane such that if all the points lie on the same side of the plane then the polygon is not partitioned.

Building an optimal BSP Tree is an NP Complete problem. The problem of building optimal trees can also be looked upon as the problem of splitting a tree versus balancing a tree. In some respects the two requirements for building optimal trees are opposed to one another. In most cases the intended use of the tree determines how these two factors should be balanced against one another.

The complexity of the BSP Tree also depends upon the application. Thus for hidden surface removal and ray tracing acceleration the complexity for the worst is \( O(n^2) \), and \( O(n) \) for the average case, where \( n \) is the number of polygons. BSP Trees are just one type of a whole class of space partitioning trees. Thus BSP Trees are related to Quad Trees and Octrees. Unlike the BSP Tree the Quad Tree recursively partition the space into four subspaces. Each node in a Quad Tree, other than the lead nodes, have four children. A special type of a quad tree - The point quadtree is an adaptation of a binary tree which is used to represent two dimensional point data. Another related datastructure is the Octtree which recursively partitions the space into eight subspaces. Octree are three dimensional analogues of Quadtrees.

A special case of BSP Trees are the KD Trees[3]. The main difference between KD Trees and other BSP
Trees is that while the former can use any splitting place, KD Trees use only splitting planes that are perpendicular to the coordinate axis. Additionally every node in a KD Tree stores a point as opposed to BSP trees where the nodes can be any type of geometric primitives. KD Trees are extensively used in indexing in multi-dimensional space.

KEY APPLICATIONS
BSP Trees has a wide variety of applications in fields dealing with spatial data. The most notable applications of BSP Trees are in Computer Graphics and in scientific domains where spatial data is used.

Computer Graphics
BSP Trees are a very efficient data structure for partitioning polygons and representing visual scenes and are consequently extensively used in Computer Graphics. Thus they are used in CAD Programs, rendering engines, animation etc.

Hidden Surface Removal
When a computer generated scene is rendered it usually consists of millions of polygons. Each of the drawn polygon can potentially be redrawn and thus the question of how to remove polygons in a scene arises. BSP Trees are extensively used for hidden surface removal[10] in computer generated graphics, especially in computer games. A example of hidden surface removal is given in figure 3

Ray Tracing
Ray tracing refers to modelling the path taken by light by following the path of rays of light as they interact with optical surfaces. The algorithm for ray tracing with BSP Trees is a simple forward tree walk[1]. It should be noted that ray tracing with BSP Trees is very similar to hidden surface removal with BSP Trees.

Solid Modeling
Geometric primitives like triangles, squares or other polygons can be used to model three dimensional space. BSP Trees can then be applied to these primitives so that the BSP trees can be used for solid modeling via Incremental construction of a BSP Tree and Tree merging. Thus BSP Trees along with other space partitioning data structures are frequently used for solid modeling [9].

Collision Detection
Two objects or moving points are said to collide with one another if they intersect at some point. Thus the
problem of collision detection can be stated in terms of a ray tracing problem. Since ray tracing can be done via BSP Trees the problem of collision detection be redescribed as a tree merging problem. Some gaming engines like Quake used BSP Trees for collision detection.

Shadow Generation
BSP Trees are also used in generating shadows of objects in computer generated scenes. [5] describes how a shadow is generated for a light source. A BSP tree is constructed for each light source such that each BSP represents the shadow volume of the polygons facing it. Once the contribution of each polygon to a light source’s shadow volume is determined, the polygon’s shadow is rendered. The lit fragments are rendered by filtering.

Rendering
BSP trees are also used in rendering in computer graphics. A commonly used algorithm for drawing scenes is the Painter’s algorithm which progressively renders objects from the background to the foreground. It is apparent that a large amount of resources are wasted in rendering since in many case objects in the foreground are rendered over objects in the background. The BSP Tree solves this problem by splitting objects so that the painter’s algorithm does not have to redraw objects and the objects can easily be accessed in the correct order by a tree traversal algorithm.

Scientific Applications
The scientific applications of BSP-Trees are in fields which deal with spatial data. Some of these applications are described below.

Geographic Information Systems
According to [4] BSP Trees have a number of applications in Spatial databases and references [12] for direct applications of BSP-Trees to Geographic Information Systems. [11] describes a system for extraction of buildings from high resolution satellite data which uses BSP Tree as data structures.

Robot Motion Planning
BSP Trees are frequently used in the cell decomposition approaches for robot motion planning. The main idea is that a path between the initial position of the robot and the destination of the robot can be determined by dividing the free space of the robot’s configuration into smaller regions called cells.[8] After the space has been decomposed into cells a connectivity graph is constructed based on the cells such that the nodes represent the cells in the free space and the edges represent the corresponding adjacent cells. A continuous path can then be constructed which charts the path of the robot from the initial point to the final point.

Consider the example of a robot which is navigating a train full of obstacles. Since the goal is path planning the ground plane can be decomposed into cells using a BSP Tree containing all of the blocks. A polygon representing the ground can then be passed into the tree. The splitting criteria for BSP Tree is this case should be modified so that connectivity information is maintained when splitting polygons. The tree can then be pruned to contain only those polygons which are the regions of the ground plane not contained inside any obstacles. A planning graph can then be built by selecting a point inside each polygon and connecting it to a point inside of each of its neighbors. Planning is done by identifying the cell which contains the initial position of the robot and also identifying the cell that contains the destination point. A search algorithm like $A^*$ can be employed on the planning graph to determine the set of polygons to be traversed in order to get from the initial position of the robot to the destination point.

Biomedical Image Registration
Image Registration is the process of establishing point by point correspondence between two images of the same scene. BSP and quad trees have been recently used in Image Registration for biomedical images [2]. In this case the image registration is based on the partitioning of the source images in binary space and quad tree structures.
FUTURE DIRECTIONS
BSP Trees are widely used in Computer Graphics and in related domains where spatial representation of a scene is required. In certain areas like hidden surface removal or ray tracing, BSP Trees seem to be the de-facto default data structure employed. Given its almost pervasive use in representing scenes in computer graphics it can reasonably be stated that the BSP Tree and its variants will continue to be used in this domain.

RECOMMENDED READING

Emergence of Geocomputation

Muhammad Aurangzeb Ahmad
Department of Computer Science and Engineering,
University of Minnesota

SYNONYMS
NONE

DEFINITION
Geocomputation is an emerging field that seeks to employ computational techniques for solving problems in geographical and spatial data. Many of these problems were considered intractable before the advent of the computers. The 'official' genesis of this field can be traced to the first international conference on 'GeoComputation' [3] at School of Geography in the University of Leeds in 1996. The main factors that led to the emergence of geocomputation was the availability of large amounts of quantitative geographical data at the disposal of geographers for the first time in history and the application of computers to solve corresponding problems.

MAIN TEXT
The pre-history of Geocomputation goes back to the late 1960s and early 1970s when computers became a widely available for geographical purposes. The period coincided with the "quantitative revolution in geography" [2]. The advent of wide availability of databases in the late 1970s gave rise to Geographical Information Systems (GIS). One of the main drawbacks in the earlier developments in the use of computers in solving geographical problems was that the type of data representation and the structure of generic databases restricted the type of problems that could be solved. Geocomputation thus emerged as a conscious effort to find a middle ground between computer science and geography such that the applications of computing in geography are not restricted by computation[1]. Although statistical techniques for analyzing spatial data were investigated even before the use of computers in geography, the complexity of many of the problems did not lend themselves to scalability until computers could be employed in solving these problems.

RECOMMENDED READING