В компютърните науки двоичното разделяне на пространството (BSP = Binary Space Partitioning) e метод за разделяне на пространството на изпъкнали региони чрез хиперравнини. Това разделяне поражда представяне на обектите в пространството чрез дървовидна структура известна като BSP-дърво.

Двоичното разделяне на пространството е разработено в контекста на 3D компютърната графика, където структурата на BSP-дървото дава възможност за бързо изчертаване на обектите в сцената представяйки полезна информация, като например подредбата на обектите в дълбочина по отношение на позицията на наблюдателя. Други приложения включват операции с геометрични форми в CAD-системите, откриване на колизии в роботиката и 3D компютърните игри, трасирането на лъчи и други приложения свързани с обработка на сложни сцени в пространството.

Преглед

Двоичното разделяне на пространството е процесът на рекурсивно разделяне на една сцена на две части докато разделянето удовлетвори едно или повече условия. На този процес може да се гледа като на обобщение на други дървовидни структури за представяне на пространствена информация като k-d-дървета и quad-дървета. Обобщение в което хиперравнините които разделят пространството, но тук разделящите хиперравнини могат да имат произволна ориентация, а не да са винаги успоредни на координатните равнини (както е при k-d-дърветата и quad-дърветата). Когато се използва в компютърната графика за изобразяване на сцени, съставени от равнинни многоъгълници, разделящите равнини често пъти (но не винаги) се избират да съвпадат с равнините дефинирани от някои от многоъгълниците в сцената.

Конкретния избор на разделяща разнина и критерий за спиране на процеса зависят от предназначението на BSP-дървото. Например, в компютърната графика, сцената се разделя докато многоъгълниците във всеки възел от дървото могат да бъдат начертани в произволен ред. Когато се използва отстраняване на обърнатите повърхности (back-face culling), всеки възел съдържа изпъкнало множество от многоъгълници, докато ако такова отсраняване не се използва, възлите съдържат само многоъгълници лежащи в една равнина. При откриването на колизии или трасирането на лъчи (ray tracing), сцената може да бъде разделена на примитивни обекти за кото тестовете за колизия или пресичане с лъч са достатъчно прости.

Двоичното разделяне на пространството възниква от нуждата на компютърната графика за бързо изчертаване на триизмерни сцени съставени от многоъгълници. Един прост метод за това е т.нар. алгоритъм на художника. Той обхожда полигоните на сцената според разстоянието до наблюдателя (по-далечните първи) и ги рисува върху фона и преди това нарисуваните многоъгълници. Този подход има два недостатъка: времето необходимо за подреждане на многоъгълниците и възможността за грешки при застъпващи се многоъгълници. Показано е, че BSP-дървото решава и двата проблема като предоставя възможност за бързо сортиране на многоъгълниците (линейно по броя многоъгълници) и избягва грешките чрез подразделяне на застъпващите се многоъгълници. Надостатък на BSP-дърветата е, че конструирането им може да е сравнително бавно. По тази причина обикновено дървото се конструира еднократно върху статичната геометрия, преди изчертаването и останалите опеарации в реално време. Бавното конструиране на BSP-дървото прави трудно и неефективно директното вмъкване на движещи се обекти в дървното.

BSP-дърветата често се използват в 3D игрите, особено в шутърите от първо лице (FPS) и тези в затворени пространства. Ядра на игри които използват BSP-дървета са например Doom (една от първите използващи BSP) и Quake. В компютърните игри BSP-дърветата съдържащи статичната геометрия често се комбинират със Z-буфер за правилно изчертаване на движещите се обекти. Макар че двоичното разделяне на пространството е удобен начин за съхраняване и извличане на пространствена информация за многоъгълниците в сцената, то не решава проблема за определяне на видимите повърхности.

Произвеждане

BSP дървeтата се използват за направата на полигони с алгоритъма на художника. Всеки полигон е направен с предна и задна страна които могат да бъдат избрани произволно и касаят само структурата на дървото, но не и нужният резултат. Такова дърво е конструирано от несортиран списък на всички полигони в дадена сцена. Рекурсивният алгоритъм за конструкцията на BSP дърво от списъка на полигони е:

1. Избор на полигон P от списъка .
2. Правене на възел N в BSP дървото и добавяне на P в списъкът с полигони при този възел.
3. За всеки друг полигон в листа:
   1. Ако полигонът е изцяло пред равнината съдържаща P, преместване на полигона в списъка с възли пред P.
   2. Ако полигонът е изцяло зад равнината съдържаща P, преместване в списъка с възли зад P.
   3. Ако полигонът е пресечен от равнината съдържаща P, разделяне на два полигона и преместване до съответните списъци с полигони пред и зад P.
   4. Ако полигонът лежи в същата равнина, в която е P, добавяне в списъка с полигони при възел N.
4. Прилагане на този алгоритъм към листа с полигони пред P.
5. Прилагане на този алгоритъм към листа за полигони зад P.

Следната диаграма илюстрира прилагането на този алгоритъм в конвертиране на лист от линии или полигони в едно BSP дърво. На всяка от 8-те стъпки, алгоритъмът по-горе е приложен към лист от линии и се добавя един нов възел към дървото.

|  |  |  |
| --- | --- | --- |
|  | Започваме със списък от линии (или полигони), представящи текущата сцена. Списъците са означени със заоблени правоъгълници, а възлите на BSP дървото – с кръгове. Във пространствената диаграма от линии, избранате посока е „предната част“ на линия, означена със стрелка. | Example of BSP tree construction - step 1.svg |
| I. | Следваме стъпките от горепосочения алгоритъм:   * 1. Избираме линия, А, от списъка...   2. и я добавяме към възел.   3. Разделяме останалите линии във списъка на тези, стоящи пред А (В2, С2, D2), и тези, стоящи зад нея (В1, С1, D1).   4. Обработваме всички линии пред А (в стъпки II-V)…   5. последвани от тези зад нея (стъпки VI-VIII). | Example of BSP tree construction - step 2.svg |
| II. | Сега прилагаме алгоритъма върху списъка с линии пред А (В2, С2, D2). Избираме линия, В2, добавяме я към възел и разделяме остатъка от списъка на линиите, стоящи пред В2 (D2), и тези зад нея (С2, D3). | Example of BSP tree construction - step 3.svg |
| III. | Избираме линия, D2, от списъка с линии пред В2. Тя е единствена, затова, след добавянето й към възел, не трябва да се прави нищо повече. | Example of BSP tree construction - step 4.svg |
| IV. | Приключихме работата с линиите пред В2, затова продължаваме с тези зад нея (С2 и D3). Избираме една от тях (С2), добавяме я към възел, и поставяме другата линия, D3, и я поставяме в списъка с линии, стоящи пред С2. | Example of BSP tree construction - step 5.svg |
| V. | Разглеждаме списъка с линии пред С2. Тъй като има само една линия, я добавяме към възел и продължаваме. | Example of BSP tree construction - step 6.svg |
| VI. | След като сме добавили всички линии, стоящи пред А във BSP дървото, продължаваме с тязи, стоящи зад нея. Избираме линия (В1) от този списък, добавяме я към възел и разделяме остатъка от списъка на линии, стоящи пред В1 (D1) и тези, стоящи зад нея (С1). | Example of BSP tree construction - step 7.svg |
| VII. | Първо обхождаме списъка от линии пред В1. D1 е единствената линия в списъка, затова я добавяме към възел и продължаваме. | Example of BSP tree construction - step 8.svg |
| VIII. | Обхождаме списъка с линии зад В1. Единствената линия е С1, затова я добавяме към възел, и BSP-дървото ни е готово. | Example of BSP tree construction - step 9.svg |

Крайният брой полигони или линии често е по-голям от оригиналният списък, тъй като линиите (полигоните), пресичащи делящата равнина трябва да бъдат разделени на две. Желателно е да се намали това увеличаване, но също така, да се забази балансът на крайното дърво. Изборът кой полигон или линия да бъде деляща равнина (стъпка 1 на алгоритъма) играе важна роля в създаването на качествено BSP-дърво.