XVI. Partycjonowanie przestrzeni: czyszczenie pokoju

W tym momencie powinieneś być w stanie wykryć kolizje i widoczność zestawu obiektów. Powinieneś teraz mieć trywialne zadanie, aby zweryfikować przecięcie statyczne. W poprzedniej części podano dość skuteczne techniki, aby osiągnąć to zadanie, ale tylko one nie pomogą Ci zbudować szybkiej gry o jakości komercyjnej. Testy są szybkie, ale musisz je zastosować do każdego obiektu na świecie. Byłoby miło, gdybyś mógł pogrupować obiekty razem, podobnie jak to było zrobione przy użyciu obwiedni, ale uogólnione na świat, a nie tylko prosty obiekt. Wyobraź sobie, jak powolna byłaby Twoja gra, gdybyś musiał wziąć udział w grze takiej jak GTA3 i zweryfikować każdy możliwy obiekt. Bez wątpienia uczyniłoby to Pentium jak X86. Ta część pokazuje, jak podzielić świat i zapisać te informacje w strukturze danych. Ta struktura danych może zająć kilka dni (nie obchodzi cię to, ponieważ może być wstępnie wygenerowana dla statycznego świata), ale musi mieć szybkość działania zabójcy i musi wydajnie dać ci mały podzbiór obiektów do przetestowania . W szczególności dowiesz się o następujących kwestiach:

■ Podział siatki

■ Czworoboki i ośmiokąty

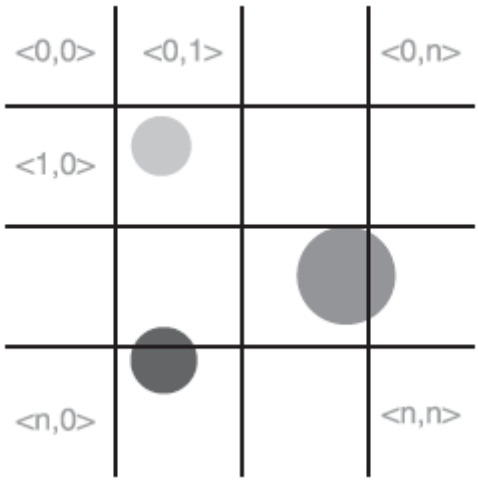
■ drzewa k-D

■ Drzewa z partycjami binarnymi (BSP)

■ Podział na portal

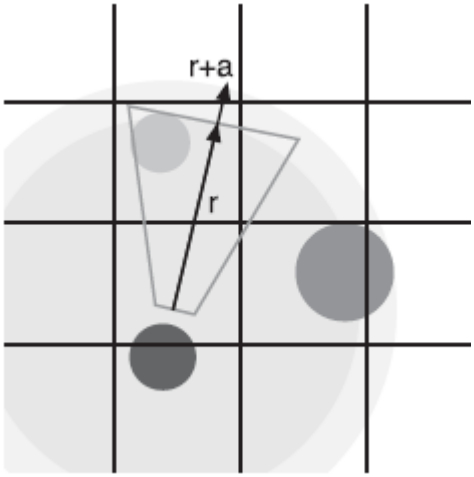
Podział Siatki

Jednym z najprostszych sposobów dzielenia świata jest podzielenie go na kwadraty o równej wielkości, jak pokazano na rysunku 16.1.



Ta prosta, ale wartościowa metoda może zrobić wielką różnicę. Struktura danych dla takiego podziału jest dość prosta. Trzeba tylko utrzymywać tablicę 2D, w której indeks <x, y> jest blokiem x^{tym} w x i blokiem y^{tym} w osi y. Tablica 2D powinna zatem zawierać wskaźnik do połączonej listy (lub tablicy statycznej), podając listę obiektów zawartych w przedziale <x, y>. Gdyby twój świat jest bardziej 3D niż 2D (na przykład, jeśli jest to kosmiczna gra, w przeciwieństwie do gry takiej jak GTA3), nadal możesz zastosować tę strukturę. Możesz podzielić świat na tablicę 3D reprezentującą poddział <x, y, z>. Zasadniczo wersja 2D jest częściej używana niż wersja 3D. Ten podział światowy może być całkiem pomocny w grach takich jak GTA3, gdzie każda dywizja ma w sobie sporo elementów. Testy na

przecięciu są uproszczone przez wymuszenie, że siatka musi być wyrównana względem osi ze światem. Jak widzisz, taki podział daje znacznie mniej skomplikowanych problemów niż brak wyrównania osi. Jedynym problemem związanym z tego typu strukturą jest to, że biorąc pod uwagę duży świat, wciąż musisz zweryfikować przeciwko dość dużej liczbie podziałów. To prawda, że test jest bardzo skuteczny ze względu na wyrównanie podziałów, ale nie jest to warte wielu testów, które możesz wykonać dla małej bryły ściętej (w porównaniu ze światem). To staje się trochę bardziej atrakcyjne, gdy myślisz poza polem. Na przykład można znaleźć podział, w którym znajduje się kamera. Jeśli długość dzielenia wynosi na przykład 10, wystarczy podzielić położenie kamery o 10, aby uzyskać indeks. Podział ten jest prawie gwarantowany, ponieważ jest to pierwszy punkt w kształcie ściętego owocu. Co jest miłe, to masz gwarancję, że widoczne podziały sąsiadują z tym samym podziałem. W ten sposób można zweryfikować każdy podział, który znajduje się w danym promieniu od środka kamery, jak pokazano na rysunku 16.2.

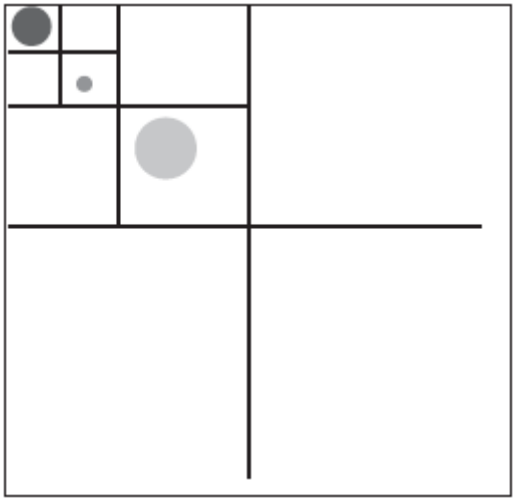


Możesz uzyskać dobre przybliżenie promienia, obliczając różnicę między bliską a daleką płaszczyzną. Jest to tylko przybliżenie, ponieważ zasięg ściętego stożka (który idzie w bok) generuje większy promień, ale można nieco zwiększyć promień, aby to zrekompensować. W końcu promień stożka ściętego powinien być w większości stały. Po uzyskaniu tej stałej możesz po prostu pozostawić ją jako stały promień. Oczywiście, znowu, będziesz musiał przekształcić swój promień przestrzeni kosmicznej w przestrzeń dzielącą, która jest tylko dzieleniem przez 10 dla twojego przykładu. Jeśli chciałbyś być jeszcze szczuplejszy i wredniejszy z podziałem 2D, możesz rzutować stożek ścięty na płaszczyźnie 2D tworzących ziemię (zakładając, że twój podział jest wykonywany na tej płaszczyźnie) i weryfikować każdy podział, który znajduje się wewnątrz lub dotyka granicy bryły ściętej. W tej metodzie nie ma widoczności ani testu kolizji między ścięciem a podziałem. Zamiast tego otrzymasz wielobok 2D zawierający zestaw podziałów, które są widoczne. Jest to niezwykle skuteczna technika, ponieważ zapewnia dokładne podziały, które są potencjalnie widoczne. Możesz to osiągnąć, obliczając przecięcie każdej płaszczyzny ściętego stożka. (To da ci osiem punktów w 3D.) Rzutuj wierzchołek ortogonalnie do płaszczyzny podłoża, odrzucając składnik y współrzędnych, uzyskując <x, z>. Po uzyskaniu tych dwuwymiarowych wierzchołków musisz obliczyć wypukły wielokąt, który łączy ten zestaw wierzchołków. Możesz to zrobić, znajdując jeden punkt w zestawie, który wiesz, że jest częścią wypukłego ciała (minimum lub maksimum), i możesz znaleźć następny punkt, znajdując wierzchołek, który generuje linię ze wszystkimi wierzchołkami po jednej stronie tego. Jeśli nie chcesz zbytnio komplikować problemu, możesz również zastosować technikę, która sprawdza tylko podział w kwadratowy sposób, gdzie długość kwadratu jest zbliżona do długości promienia. Z drugiej strony, ta metoda nie wygląda na to, że bardzo pomaga, ponieważ wciąż ma wiele działów do sprawdzenia. Ale wygląda całkiem nieźle, jeśli spojrzeć na okoliczne podziały z podziałem kamery. W rzeczywistości, jeśli

rozmiar podziału jest dobrze dobrany, może to być jedna z najlepszych dostępnych metod podziału. Ta metoda jest naprawdę przydatna w przypadkach, gdy świat jest dość zagracony (przychodzi mi na myśl GTA3). W przeciwnym razie skończysz z podziałami, które nie zawierają żadnych obiektów, i wtedy zaczynają być dość drogie pod względem pamięci. Pamięć jest głównym problemem w tej metodzie. Wybór małego podziału sprawi, że rzeczy będą wyjątkowo szybkie, ale pozostawi wiele pustych podziałów (a więc utraconą pamięć). Z drugiej strony, jeśli wybierzesz duże dywizje, możesz skończyć z jedną dywizją niosącą zbyt wiele elementów. Byłby to podział zbyt duży dla świata. Równowaga jest kluczowa i zależy od konkretnej gry. Jest to prawdopodobnie miejsce, w którym chcesz przeprowadzić analizę statystyczną i sprawdzić, który rozmiar najlepiej pasuje do konkretnego problemu. Piękno dzielenia świata polega na tym, że dotyczy to nie tylko determinacji widoczności, ale także wykrywania kolizji. Zmniejsza to problem do poziomu, który pomaga całej twojej grze radzić sobie z mniejszą liczbą obiektów, co jest ogólnie rzeczą wspaniałą. W szczególności siatki są bardziej przydatne do wykrywania kolizji, ponieważ liczba podzadań, które mają być testowane, jest ogólnie mniejsza. Nie potrzebowałbyś bardzo drobnoziarnistej siatki, gdybyś używał jej do wykrywania kolizji, podczas gdy widoczność wymagałaby bardziej drobnoziarnistej siatki. Jest to prawdą w ogólności, ale zawsze naprawdę zależy bardziej od gry, którą tworzysz, niż od czegokolwiek innego.

Drzewo czwórkowe i Drzewo Ósemkowe

Poprzednia metoda jest świetna pod względem szybkości, ale jest głodna. Nie wiele możesz zrobić, aby to zmienić bez drastycznej zmiany sposobu działania tej metody. Ta drastyczna zmiana metody jest tym, co proponują próby czworoboku. W szczególności problem z poprzednią funkcją polegał na tym, że jedna strona świata mogła mieć tylko jeden obiekt, a mimo to podzieliła przestrzeń na równe interwały, co marnowało pamięć. Jeśli podzielisz tylko drugą połowę i zachowasz połowę nietkniętą, zaoszczędzisz znaczną ilość miejsca. W tym miejscu pojawia się czworobok. Zdefiniowany abstrakcyjnie czworobok jest siatką rekurencyjną. Aby uprościć problem i złagodzić problemy z pamięcią, potrzeba siatki, która dzieli n spacji na dwie części na wymiar. W ten sposób otrzymujesz w sumie 2n podziałów dla jednego etapu rekursji, jak pokazano na rysunku 16.3. Struktura danych dla takiego podziału oczywiście różni się od podziału siatki. Ze względu na swoją rekurencyjną naturę drzewa czwórkowego bardzo dobrze wiąże się z drzewem. Korzeniem drzewa powinno być pudełko, które trzyma cały świat. To pole powinno być podzielone na pół dla xi na pół dla y, dając w ten sposób cztery nowe podziały i czwórkę dzieci drzewa root. Jeśli podziały są puste, bezużyteczne jest zastosowanie procesu rekursywnie w poddziale. W takim przypadku można ustawić wskaźniki dla dzieci na NULL. Z drugiej strony, jeśli w dywizjonie jest więcej niż jeden obiekt, będziesz chciał go jeszcze raz podzielić. Praktyczną zasadą przy budowie kwadratu jest to, że obiekt musi być całkowicie zawarty w obrębie podziału. Innymi słowy, obiekt nie może obejmować więcej niż jednego podziału. W przypadku poprzedniej metody, jeśli obiekt obejmował więcej niż jeden podział, wystarczy powtórzyć obiekt w obu działach i upewnić się, że nie zostanie narysowany dwukrotnie (wstawiając obiekt do listy możliwej do renderowania). Tutaj nie możesz tego zrobić, ponieważ nie wiesz, kiedy przerwać rekursję. Oczywiście, jeśli ściśle przestrzegasz powyższej reguły, wszystkie kształty wpadną do węzła głównego. Właściwa zasada jest taka, że obiekt powinien być przypisany do najmniejszego podziału, w którym jest całkowicie zawarty. Tak więc, gdy wstawiasz obiekt do drzewa czwórkowego, musisz podzielić rekursywnie drzewo czwórkowe, aż uzyska punkt przecięcia z obiektem. W tym miejscu musisz przypisać go do ostatniego liścia (to znaczy do tego węzła końcowego drzewa, który jest podziałem, który nie jest w tym przypadku podzielony). Można to zobaczyć na rysunku 16.3.

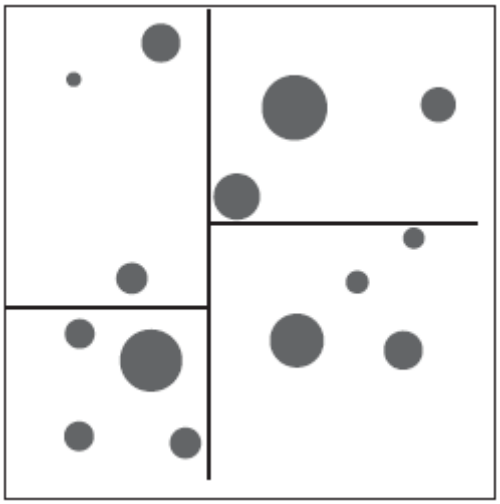


Największy krąg nie mógł sobie pozwolić na podzielenie swojej przestrzeni na pół bez powodowania przecięcia, podczas gdy dwa inne okręgi mogłyby. Kiedy przychodzi czas na sprawdzenie skrzyżowania lub kolizji, należy rozpocząć od sprawdzenia, z której z czterech połówek przecinają się, czyli za pomocą prostego testu kolizji obiekt-obiekt. Jeśli jedna z ćwiartek się nie przecina, możesz pominąć tę część przestrzeni. Po ustaleniu, które kwartały przecinają się, musisz sprawdzać w podziale na każdy kwartał, aż osiągniesz liść, w którym to momencie będziesz chciał przetestować zestaw obiektów, które znajdują się w tym liściu. Oczywiście struktura danych dla każdego liścia kwadratu jest połączoną listą obiektów, które są zawarte w tym podziale (ponownie, tak jak działała metoda siatki). Pod względem szybkości nie oszczędzasz tak bardzo, jak w poprzedniej metodzie, ponieważ może to dać całkiem dobre przybliżenie zbioru przecinających się regionów w jednym przebiegu. Ta metoda wymaga kilku powtórzeń rekursywnych, ale oszczędzasz dużo pamięci, nie dzieląc pustych regionów. Quadrree jest strukturą 2D, a jego rodzeństwo jest wersją 3D. W przypadku ósemki nie tylko dzielisz przestrzeń na x i y, ale także na z. Oznacza to, że każdy węzeł ma maksymalnie ośmioro dzieci. Sam proces pozostaje taki sam jak w przypadku poczwórnego, z wyjątkiem tego, że musisz uwzględnić większą liczbę dzieci.

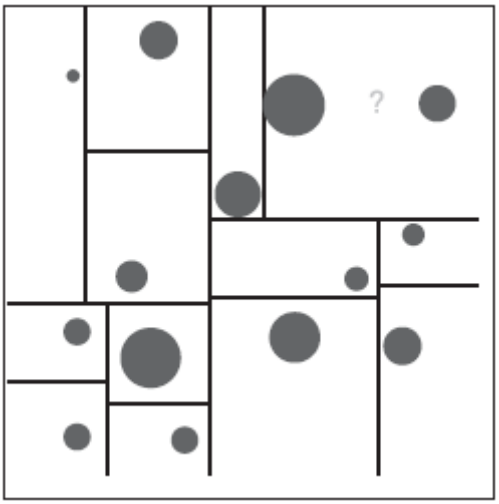
Drzewa k-D

Jeśli spojrzeć na strukturę drzewa czwórkowego / drzewa ósemkowego z punktu widzenia danych statystycznych, drzewo nie jest ogólnie zrównoważone. Idealne drzewo zawsze podzieliłoby świat na pół dla każdej osi. Jeśli połowa obiektów jest podzielona na x, połowa z nich w y, a druga połowa w z, podzieliłeś podział, który miał n obiektów na kolejne cztery (lub osiem) podziały, które mają obiekty n / 4 (lub n 8 na ośmiu) każdy. Załóżmy na przykład, że masz świat, w którym 99% obiektów jest zagraconych w tym samym segmencie oktawy. Drzewo ósemkowe jest mało pomocne, ponieważ zawsze musisz zweryfikować skrzyżowania dla tego podziału, w którym znajduje się większość obiektów. I oczywiście musisz zweryfikować każdy z tych obiektów. Z drugiej strony, jeśli masz zbalansowane drzewo, połowa tych obiektów znajduje się w oddzielnym obszarze niż druga połowa, a ponadto dla wszystkich innych osi. Drzewo jest efektywnie zrównoważone, ponieważ każdy obiekt jest dzielony równomiernie. Drzewo k-D jest k-wymiarowym drzewem, które rozwiązuje ten problem, dzieląc przestrzeń nierównomiernie, geometrycznie mówiąc. Przestrzeń jest podzielona równomiernie pod względem gęstości obiektów. Oznacza to, że jest tyle przedmiotów w jednej połowie, ile jest w drugiej połowie. W pewnym sensie można zdefiniować drzewo k-D jako zrównoważoną oś ósemkową. Drzewo k-D jest strukturą danych, która dzieli świat na pół dla każdego podziału. Nie dzieli świata na połowę geometrycznie, lecz raczej na gęstość (liczbę obiektów). Na przykład pierwsze przejście drzewa k-D zlicza liczbę elementów i dzieli świat na pół według współrzędnej x. Innymi słowy, znajdziesz wartość x, która jest taka, że połowa wartości x obiektów jest mniejsza od niej, a druga połowa jest

większa od niej lub równa. Jest to znacznie bardziej wydajna metoda niż ósemka. Dla każdego przejścia drzewa k-D masz gwarancję, że spojrzysz na połowę liczby obiektów, które wcześniej oglądałeś. Tak więc, jeśli masz n obiektów, powinieneś oczekiwać log2 (n) przechodzi (plus podatek, jak wkrótce zobaczysz). Drzewo k-D działa w jednym kierunku na raz. Porządek kierunku zależy od ciebie, ale załóżmy teraz, że podzielimy się kolejnością podziału {x, y, z}. Oznacza to, że powinieneś najpierw podzielić na x. Po podzieleniu obiektów 50/50 na x, spójrz na podział dwóch podregionów na y. Gdy to zrobisz, spójrz na podział subregionów w z. Następnie proces ten jest stosowany rekurencyjnie. Jedną z rzeczy, które mogłeś zauważyć, jest to, że nie daje to miłego symetrycznego modelu. Pierwszy podział na x da ci dwie połówki. Ponieważ gęstość obiektów jest prawdopodobnie różna dla każdej strony, wartość y, która dzieli prawą połowę prawdopodobnie nie będzie taka sama jak ta, która dzieli lewą połowę, jak pokazano na rysunku 16.4.



To nie jest prawdziwy problem jako taki, ale oznacza to, że będziesz musiał śledzić wartość, w której wykonujesz podział. Innymi słowy, będziesz musiał śledzić wartość, w której podzieliłeś komponent x na pierwsze przejście. Podobnie, będziesz musiał śledzić wartość y, w której podzieliłeś świat w każdym podregionie i tak dalej. Jeśli dalej dzielisz drzewo, jak pokazano na rysunku 16.5,

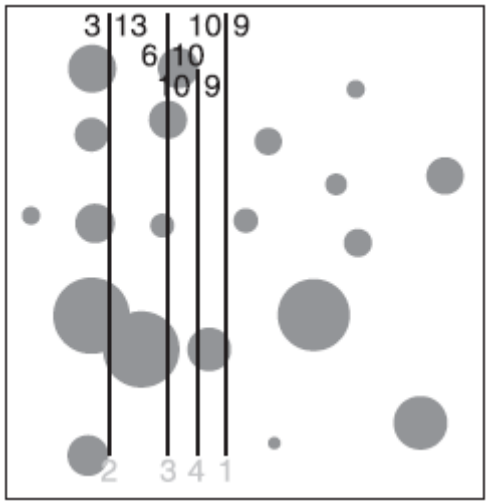


w końcu uderzysz w ścianę. Oznacza to, że dojdziesz do punktu, w którym nie ma czegoś takiego jak linia dzieląca, która może dzielić dwa obiekty (lub dwa zestawy obiektów) w równy sposób. Jeśli zatrzymasz tutaj proces, podobnie jak w przypadku ósemka, nie możesz podzielić zestawu obiektów

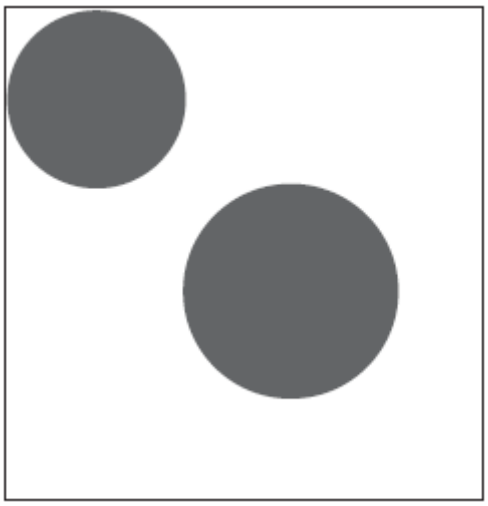
na dwie części. Oznacza to, że drzewo k-D w zasadzie staje się połączoną listą. Oczywiście to nie jest dobre rozwiązanie. Zamiast tego możesz odłożyć na bok regułę dotyczącą posiadania dokładnie takiej samej liczby obiektów z każdej strony (lub z jedną różnicą, w przypadku, gdy całkowita liczba obiektów jest nieparzysta), i po prostu znajdź najlepsze dopasowanie. "Najlepsze dopasowanie" definiuje się jako podział minimalizujący różnicę między dwoma stronami. W przypadku rysunku 16.5, musisz dodać pozorną linię poziomą, aby zakończyć wszystko pionową linią, która poprawnie podzieliłaby oba obiekty

Znalezienie najlepszego podziału na osi

Technika ta prawie zawsze działa, gdy dwa obiekty się nie przecinają. Jeśli nie przecinają się, musi istnieć oś oddzielająca je (od twierdzenia o osi rozdzielającej widziane w poprzedniej części). Trudną częścią jest to, że ta struktura pozwala tylko na wycięcia wzdłuż osi, więc możesz mieć dwie sfery wystarczająco blisko, aby linia wyrównana do osi mogła przeciąć przestrzeń na pół. W tym zgniłym przypadku dwa obiekty muszą zostać zarejestrowane w jednym dziale. Idealnie byłoby wygenerować drzewo, w którym jest maksymalnie jeden obiekt na podział. Z powyższych powodów nie zawsze jest to możliwe, więc musisz także poradzić sobie z podziałem zawierającym wiele elementów. Teraz musisz zwrócić się do nowo powstałego problemu, który polega na znalezieniu najlepszej linii, która dzieli zestaw obiektów na dwa równe (lub jak najbliżej jak to możliwe) zestawy. Ponieważ nie istnieje wymóg generowania takiej struktury w czasie rzeczywistym (nie chcesz dynamicznie generować tej struktury, jest to jednorazowa operacja dla statycznego świata), możesz użyć metody, która zajmuje trochę więcej czasu, ale jest łatwa do wdrożenia i rozumiem. Technika opiera się na metodzie dziel i rządź. Jeśli podzielisz przestrzeń na połowę geometrycznie, możesz policzyć, ile obiektów znajduje się całkowicie po jednej stronie linii, a ile jest całkowicie po drugiej stronie. Upewnij się, że uwzględniasz efektywny promień każdego obiektu. To sprawia, że jest to trudniejsze zadanie niż powinno być. Jeśli dostaniesz tam równość, możesz nawet nie zostać zrobioną, ponieważ może istnieć przecięcie między obiektem a tą linią. Jeśli jest skrzyżowanie, być może będziesz musiał kontynuować algorytm. Jeśli nie znajdziesz przecięcia, a spacja jest podzielona na pół z wyjątkiem nieparzystej liczby, powinieneś przerwać. Inną sytuacją, w której powinieneś się zatrzymać, jest dzielenie na pół (do nieparzystej liczby obiektów)) i przecinają go obiekty po obu stronach linii. Jeśli znalazłeś linię, która równoważy liczbę obiektów po obu stronach i dla których obie strony się przecinają, nie będziesz w stanie znaleźć linii, która podzieli tę przestrzeń na połowę, która się nie przecina. Z drugiej strony, jeśli okaże się, że obiekty z jednej strony przecinają linię, nadal może istnieć linia, w której w ogóle nie występuje przecięcie, a która dzieli tę przestrzeń w równym stopniu. Dlatego musisz kontynuować algorytm. Aby zastosować rekursję, musisz zdecydować, która połowa linii będzie rekursywna. Strona z największą liczbą obiektów powinna zostać zmniejszona w obszarze, aby niektóre obiekty mogły zostać przesunięte na drugą stronę. W związku z tym musisz dodać lub odjąć połowę całkowitej odległości w x od bieżącego x i ponownie zastosować obliczenia. To, czy dodasz, czy odejmiesz tę wartość, zależy od tego, czy największa liczba obiektów znajduje się po lewej stronie (gdzie chcesz odjąć), czy po prawej (gdzie chcesz dodać). Następnie należy zastosować ten proces rekursywnie, jak pokazuje rysunek 16.6.



W przypadku, gdy obiekty są podzielone równo, ale przecięcie występuje tylko po jednej stronie linii, należy przesunąć linię w kierunku strony, gdzie nie ma przecięcia. Dlatego musisz mądrze wybrać stronę dodawania lub odejmowania. Jedyny problem to wiedzieć, kiedy zakończyć ten proces. Co się stanie, jeśli spróbujesz uruchomić ten algorytm na przykładzie z rysunku 16.7?



Algorytm będzie działał jak Forrest Gump, nigdy nie wróci na ten świat. Aby to naprawić, możesz zastosować próg. Działa na zawsze, ponieważ dwa okręgi naprawdę nie przecinają się ze względu na pionową lub poziomą linię. Biorąc pod uwagę linię wyrównaną do osi, zawsze będzie jeden obiekt po drugiej stronie i jeden przecinający linię, ale dwa obiekty nigdy nie przecinają linii w tym samym czasie. Zasadniczo są nieskończenie blisko siebie. Możesz powiedzieć, że gdy wartość dodana (lub odejmowana) staje się mniejsza lub równa 0,0001, zatrzymujesz i wybierasz bieżącą linię jako linię oddzielającą. Jest to najlepsze przybliżenie linii podziału, która dzieli przestrzeń na pół bez faktycznego rozdzielenia jej na pół (z powodu niemożliwości). Teraz, aby podsumować najlepszy algorytm przeszukiwania linii w osi:

```
Function (x = split line, T = Total Length of the axis, d = depth)
{
for (every objects) {
if (object center + effective radius < x)
Left++;
```

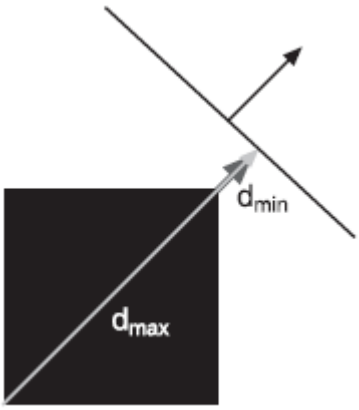
```
else if (object center – effective radius > x)
Right++;
}
if (Left + OddNumberOfObjects < Right)
Recurse(x – T/2, T/2, d++)
else if (Left > Right + OddNumberOfObjects)
Recurse(x + T / 2, T/2, d++);
if (intersections on only one side)
Recurse(x _ T/2, T/2, d++);
// No Intersection, so return the split line
return x;
}
```

Teraz właśnie skończyłeś dzielić przestrzeń na pół na x. Powinieneś mieć jedno drzewo z jednym korzeniem zawierającym wartość x, która jest wartością podziału x = a, którą znalazłeś przy poprzednim algorytmie. Węzeł ten powinien mieć dwoje dzieci, jeden po lewej, a drugi po prawej. Następnym krokiem jest zrobienie lewego zestawu i zastosowanie tego samego podziału w y. Poniżej znajduje się właściwa strona, w której również można zastosować podział na y. Po tym można kontynuować, stosując podział w z dla każdego z czterech poddziałów (dając ośmiu dzieci). Na końcu drzewa (dla każdego liścia) powinieneś mieć listę połączoną (lub tablicę statyczną) zawierającą listę obiektów znajdujących się w tym obszarze. Ta połączona lista powinna być dość mała, ponieważ szanse na to, że wiele obiektów zostanie obciętych o jedną linię (lub samolot w 3D) są dość małe. Pojawia się także jeden pozostały problem, który jeszcze nie został rozwiązany: Jak określić, kiedy podział, taki jak ten na rysunku 16.7, nie jest możliwy i czy można uzyskać dofinansowanie? Jeśli zastosowałeś to rekurencyjnie, to przestrzeń byłaby ciągle dzielona przez x, y, z, x, y, z. . . Musisz znaleźć warunek, który powstrzyma to szaleństwo. W rzeczywistości jest to dość łatwe do wykrycia. Jeśli wszystkie osie są takie, że całkowita liczba obiektów, które nie przecinają się z linią, wynosi co najwyżej 1 (w przypadku grupy nieparzystej), powinieneś wyskoczyć z dywizji. Jak zwykle, ponieważ niektóre obiekty mogą pojawiać się więcej niż raz, należy uważać, aby dwukrotnie nie zastosować testu dla tych obiektów. Prosta flaga załatwiłaby sprawę tutaj.

Spacerując po drzewie

W dwóch ostatnich metodach tę sekcję pominięto, głównie dlatego, że dość łatwo jest określić, które obszary przecinają się ze stożkiem ściętym (do badania widoczności) lub jak znaleźć obszar określonej lokalizacji (do testów zderzeniowych). W pierwszej metodzie wystarczyło skalować współrzędne, natomiast w ósemkowym trzeba było powtórzyć i wykonać test przecięcia skrzyni z frustum. Powinieneś już widzieć te techniki już teraz. Na szczęście ta struktura nie jest dużo trudniejsza niż poprzednie dwa. Obszary są nadal definiowane jako pola, chociaż nie są wyrównane względem osi. Znalezienie obszaru, biorąc pod uwagę pozycję, jest dość łatwe. Musisz po prostu sprawdzić, czy obiekt znajduje się po dodatniej lub ujemnej stronie linii / płaszczyzny. Zastosuj ten proces rekursywnie i ostatecznie osiągniesz punkt, w którym nie występują podziały. Oznacza to, że znalazłeś obszar zawierający współrzędną, której szukasz. W przypadku frustum jest to jednak o wiele bardziej bolesny proces. Będziesz musiał spojrzeć wstecz na punkt przecięcia się stożka ściętego z płaszczyzną. Jedyny problem polega na tym, że w twoim przypadku wiedza, czy cielesny przecina się z samolotem, nie daje ci pełnej informacji. Jeśli wiesz, że płaszczyzna dzieli stożek ścięty na dwa obszary, wiesz, że obie strony płaszczyzny są potencjalnie w ciele. Zatem musisz wejść głębiej w drzewo po każdej stronie samolotu. Jeśli stożek ścięty jest całkowicie zawarty na jednej stronie, powinieneś wejść głębiej w drzewo tylko po tej stronie. Ze względu na konstrukcję tego drzewa żadne przedmioty nie powinny leżeć po drugiej stronie (tej, która nie zawierała stożka ściętego). Ale musisz wiedzieć, po której stronie to jest, zanim

będziesz mógł wybrać lewą lub prawą gałąź swojego drzewa. Jeśli zaczniesz od przekształcenia płaszczyzny w przestrzeń ściętego stożka, wygenerujesz stożek ścięty, którego rogi będą miały format <±1 , ±1 , ±1>. W związku z tym, jeśli obliczysz odległość między najbliższym wierzchołkiem a płaszczyzną, a także najdalszy wierzchołek z płaszczyzny, uzyskasz minimalną i maksymalną odległość między pudłem a płaszczyzną. Zostało to zilustrowane na rysunku 16.8



i jest podane za pomocą następujących wzorów dla płaszczyzny <n, D>:

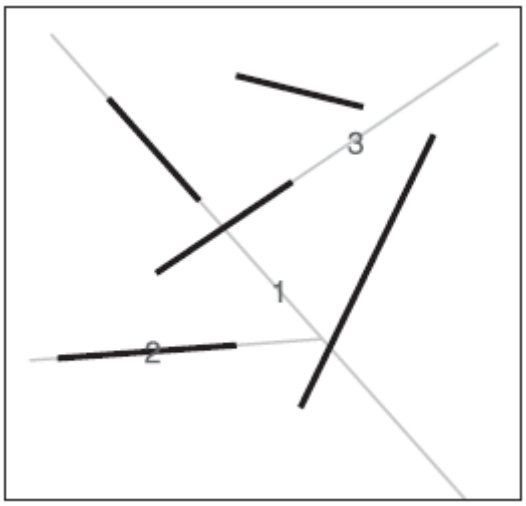
$$d_{\max} = |\mathbf{n}_x| + |\mathbf{n}_y| + |\mathbf{n}_z| + D$$
$$d_{\min} = -|\mathbf{n}_x| - |\mathbf{n}_y| - |\mathbf{n}_z| + D$$

Jeśli te dwie wartości mają różne znaki, stożek przecina się z płaszczyzną i dlatego obie strony powinny być odwiedzane. Jeśli obie wartości są ujemne, stożek ścięty leży po ujemnej stronie płaszczyzny. Odwrotnie, jeśli obie wartości są dodatnie, stożek ścięty leży po dodatniej stronie płaszczyzny (która jest w połowie wskazywana przez normalny wektor). Dzięki tej metodzie nie trzeba nawet obliczać odległości między pudłem a płaszczyzną, ponieważ ta metoda informuje również, czy było przecięcie, czy nie.

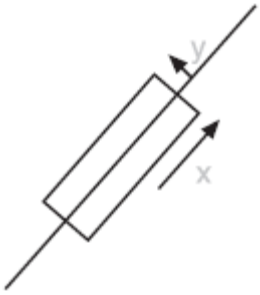
Drzewa Binary Space Partitioned (BSP)

Przestrzenie binarne Partycjonowane drzewa były rozpalone kilka lat temu. Zostały one upowszechnione przez takie gry, jak Doom, a później przez Quake'a, co rozszerzyło koncepcję do prawdziwego świata 3D. BSP są nadal dość popularne w grach indoor. Jeśli przejdziesz do gier na świeżym powietrzu, zwykle poprzednie metody są bardziej wydajne. BSP jest naturalnym następnym krokiem po drzewie k-D. BSP jest drzewem k-D, którego płaszczyzny nie muszą w rzeczywistości być wyrównane względem osi. Innymi słowy, drzewo BSP umożliwia cięcie przestrzeni za pomocą linii (lub płaszczyzny w 3D) bez względu na jej wyrównanie. To oczywiście wprowadza większą elastyczność (a tym samym większą, dokładniejszą metodę usuwania), co faktycznie rozwiązuje problem, który pojawił się na rysunku 16.7. W rzeczywistości, dopóki dwa obiekty nie przecinają się w świecie (coś, co można wymusić), zgodnie z twierdzeniem osi rozdzielającej, zawsze będzie istniała linia / płaszczyzna, która oddziela dwa obiekty. Używanie drzewa BSP do wykrywania uboju lub kolizji jest łatwą częścią. Jeśli rozumiesz drzewo k-D, w zasadzie rozumiesz drzewo BSP. Tam, gdzie staje się ono znacznie trudniejsze, znajduje się w tworzeniu drzewa. Idealnie, drzewo zawsze podzieliłoby bieżący podział na równe połowy, tworząc logarytmiczne drzewo dostępu. Metoda, której używasz do stworzenia drzewa BSP, naprawdę zależy od tego, jakie masz zastosowania, a także od świata, z którym masz do czynienia.

Podana tutaj konstrukcja dobrze pasuje do świata zbudowanego z wielu ograniczonych obiektów (jak w poprzednich przypadkach). W przypadku gier wewnętrznych, takich jak Quake i Unreal, ściany i sufit oraz podłoga będą faktycznie odpowiednikami obiektów w tej metodzie. Oczywiście, ponieważ są to ściany, można zastosować bardziej precyzyjne strojenie, ale ogólny pomysł jest taki sam. Jak zatem określić dobry kandydat na samolot? Gdybyś miał orientację, minimalnie zablokowałby jedną zmienną, a pozycja byłaby jedyną pozostałą. Prawdziwym problemem jest tutaj duża liczba samolotów, które możesz wybrać, aby oddzielić świat. Możesz łatwo marnować dni, próbując znaleźć najlepiej pasujący samolot. Aby zaoszczędzić pamięć i uprościć rzeczy, stosuje się podejście Quake / Unreal / Doom. Płaszczyzna cięcia jest wybrana tak, aby pasowała do ściany (lub podobnej geometrii dla w pełni trójwymiarowych światów). Rysunek 16.9 ilustruje wygląd drzewa BSP w Doomie.



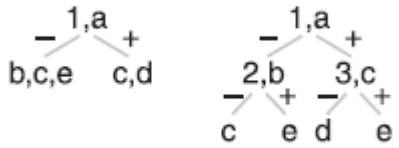
Linie na tym rysunku reprezentują ściany tak, jak widziałbyś je z widoku z góry. Jak widać, jeden segment linii oddziela przestrzeń na pół. Dla lewej i prawej połówki, przestrzeń jest podzielona odpowiednio przez segmenty linii 2 i 3. Jak pewnie zauważyliście, niektóre podziały przecinają kolejny segment na dwa. W takich przypadkach gry takie jak Doom mają tendencję do dzielenia tej ściany / wielokąta na dwie części, tak aby obie z nich wzajemnie się wykluczały. Pomaga to, gdy nadejdzie czas renderowania, ponieważ oznacza to, że nie trzeba sprawdzać, czy obiekt został już wyrenderowany. Podobnie jest w przypadku kolizji. W przypadku takich ścian jest to dość łatwe. Wybierasz ścianę i dzielisz przestrzeń przy tej ścianie. Wszystko z jednej strony idzie po jednej stronie drzewa, a reszta po drugiej stronie drzewa. Tę procedurę stosuje się rekurencyjnie, dopóki obszar nie zawiera co najwyżej jednego elementu. Różnica między BSP a ścianą w tym miejscu jest taka, że BSP zawiera również informacje o obiektach w węzłach. Odsuńmy trochę od scenariusza Zagłady i załóżmy, że chcesz podzielić zestaw obiektów (który może być ograniczony objętością ograniczającą). Ściana byłaby po prostu postrzegana jako nieskończenie cienki ograniczony prostokąt. Biorąc pod uwagę obiekt, należy wybrać płaszczyznę, która przebiega w tym samym kierunku, w którym znajduje się obiekt. Ta zasada "ogólnego kierunku" oznacza jedynie, że powinieneś wybrać najdłuższy zakres objętości ograniczającej jako kierunek dla samolotu. Aby być nieco bardziej szczegółowym, płaszczyzna oddzielająca świat powinna być płaszczyzną, dla której wektor normalny jest najmniejszy, aby zminimalizować odległość między objętością ograniczającą obiektu a płaszczyzną, jak pokazano na rysunku 16.10.



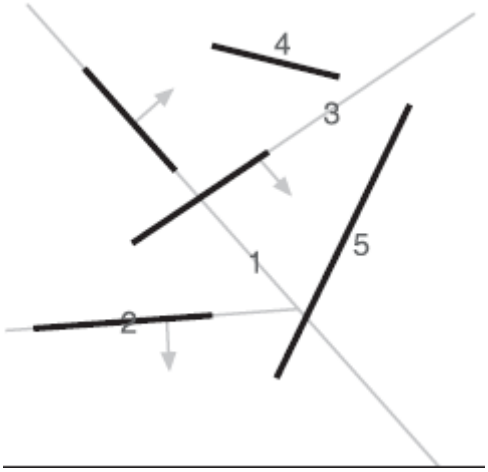
Ten plan jest dość łatwy do obliczenia. Jeśli masz wektor najmniejszego zakresu v (może to być obszar elipsoidy, skrzyni, cylindra ...), możesz obliczyć równanie dla płaszczyzny podziału, takie jak

$$s = \langle \mathbf{n}, -\mathbf{n} \bullet \mathbf{c} \rangle$$

Tutaj c jest środkiem obiektu. Wraz z płaszczyzną podziału należy również przechowywać obiekt w węźle drzewa (więcej o tym później). Po napisaniu pierwszego węzła obiekty znajdujące się całkowicie po dodatniej stronie płaszczyzny idą na lewą gałąź, podczas gdy pozostałe elementy idą na prawą gałąź. Oczywiście, nie zawsze można podzielić obiekt na dwie części. Jeśli możesz sobie na to pozwolić, zaoszczędzi ci to trochę czasu. Jeśli nie jest to opcja lub nie chcesz przeszkadzać w dodatkowym zwiększeniu prędkości, będziesz musiał przechowywać obiekt zarówno po stronie dodatniej, jak i ujemnej płaszczyzny, tak jak w poprzednich metodach. Na koniec przyjrzyjmy się strukturze danych nieco bardziej szczegółowo. Struktura danych jest drzewem binarnym (ma dwoje dzieci). Jedno dziecko jest dla obiektu siedzącego na dodatniej połówce (połowa wskazywana przez normalny wektor), a pozostałe dziecko jest w drugiej połowie. Początkowo świat jest podzielony na pół przez pierwszą płaszczyznę (lub linię). Ta płaszczyzna jest powiązana z obiektem, który ma oś w najmniejszym stopniu w tym kierunku. W przypadku kuli, gdzie tak naprawdę nie ma największego lub najmniejszego zakresu, można użyć metody statystycznej przedstawionej w części 11, aby określić, która oś jest najmniejsza. Obiekt ten jest przechowywany w drzewie wzdłuż płaszczyzny podziału. Rekursywnie zestaw obiektów dla dzieci jest podzielony na partycje. Na przykład spójrz na rysunek 16.9. Ponadto zdefiniuj pozostałe dwie linie, które nie są numerowane jako 4 i 5 od góry do dołu. Przypuśćmy, że te linie reprezentują obiekty posortowane alfabetycznie (1 = a, 2 = b, itd.). Etapy budowy drzewa będą takie, jak pokazano na rysunku 16.11.



Oczywiście nie ma sensu budować drzewa, jeśli nie wskazano pozytywnych i negatywnych kierunków podziału. W związku z tym drzewo rzeczywiście odzwierciedla rysunek 16.12,



który wskazuje kierunek płaszczyzn (lub linii dla 2D). Ostatnią rzeczą, którą należy się zająć, jest wybór samolotu. Wiele analiz zostało wykonanych na drzewach BSP, a generalnie losowy wybór generuje całkiem rozsądne drzewo BSP. Jeśli chcesz zmarnować dodatkowe cykle, możesz zaimplementować metodę prób i błędów, która spróbuje znaleźć obiekt, który podzieli miejsce na najbardziej równe części. Taka metoda nie jest zła, ale nie ma gwarancji, że wygeneruje najbardziej zrównoważone drzewo.

Renderowanie z przodu do tyłu

W dniach Dooma, gdy karty przyspieszenia sprzętowego były dostępne tylko na niezwykle kosztowne profesjonalne stacje robocze, techniki widoczności i renderowania różniły się od tego, czym są dzisiaj. Ponieważ bufor z-z jest niezwykle kosztownym oprogramowaniem, gry takie jak Doom nie mogły sobie pozwolić na użycie czegoś podobnego. W związku z tym nadal wymagały sposobu renderowania sceny tak, aby widoczność była zachowana. BSP może dać ci zestaw widocznych powierzchni, ale co z porządkiem wielokątów? Jeśli nie masz bufora Z (lub czegoś podobnego), kolejność renderowania staje się ważna. Jeśli wykonasz bliską powierzchnię przed dalszą powierzchnią, nie będziesz szanować faktu, że najbliższa powierzchnia zamyka drugą. Dlatego najczęściej stosowaną techniką był algorytm malarza. Ta technika jest tak prosta, jak się wydaje. Chodzi o to, aby najpierw poligony, które są najdalej od aparatu. W ten sposób widoczność zostaje zachowana. Przy dzisiejszym sprzęcie możesz to osiągnąć, jeśli planujesz wyłączyć bufor Z, ale dlaczego chcesz to zrobić? Zamiast tego należy zoptymalizować renderowanie, aby skorzystać z bufora z. Jeśli umiesz pisać wieloboki w algorytmie odwrotnego malowania - innymi słowy, renderuj wielokąty z bliska - możesz zaoszczędzić znaczną ilość GPU. Dokładniej, można zaoszczędzić dużo w rurociągu rasteryzacji, ponieważ każdy piksel, który został znaleziony w buforze z, nie będzie wyświetlany na ekranie. Z drugiej strony, gdybyś rysował używając czegoś podobnego do algorytmu malarskiego (renderowania z dalekiej do najbliższej), to byś miał dużo przekroczeń, co nie jest dobre. Nowoczesny sprzęt wykonuje wczesną próbę, aby uniknąć kosztownych operacji na fragmentach, takich jak teksturowanie fragmentów, które i tak nie są widoczne. Dlaczego tak jest w przypadku tego rozdziału? Cóż, drzewa BSP zostały użyte w Doomie do renderowania przy użyciu algorytmu malarskiego. Jeśli możesz renderować z powrotem na wierzch, z pewnością możesz renderować od początku do końca, po prostu odwracając proces. BSP lub dowolna połowa płaszczyzny (podział rekurencyjnie różnicujący stronę dodatnią i ujemną płaszczyzny) pozwala na taką technikę. Sztuczka polega na tym, jak przemierzasz drzewo. Musisz odwiedzić zestaw obiektów, które są bliżej kamery niż druga połowa. To niezwykle proste. Zakładając, że masz listę obiektów, które są widoczne, im bliżej kamery znajduje się bliższa płaszczyzna, tym bliżej zestawu obiektów opisanych w niej znajduje się kamera. Zakładając, że obie strony płaszczyzny są widoczne, wystarczy określić, która z dwóch podziałów jest bliżej kamery. Bardzo łatwo jest stwierdzić, która połowa jest bliżej

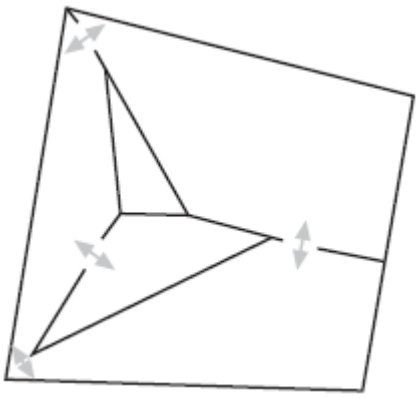
aparatu. Wystarczy sprawdzić, czy sam aparat znajduje się na dodatniej lub ujemnej stronie samolotu. Obiekty po tej samej stronie płaszczyzny są bliżej kamery niż obiekty po drugiej stronie płaszczyzny. Ten test nie różni się od testu, który można zastosować do dowolnego obiektu. Ponieważ już wiesz, jak ustalić, czy obiekt znajduje się na dodatniej lub ujemnej stronie samolotu, wybierz negatywne lub pozytywne dziecko drzewa, w zależności od następujących. Ponieważ już wiesz, jak określić, czy obiekt znajduje się na dodatniej czy ujemnej stronie płaszczyzny, wybierz ujemne lub dodatnie potomstwo drzewa, w zależności od poniższej formuły dla płaszczyzny <n, D> i kamery 4D <c, 1>

$$Side = \langle \mathbf{n}, d \rangle \bullet \langle \mathbf{c}, 1 \rangle$$

Side < 0 oznacza ujemną pół-płaszczyznę, natomiast Side > 0 oznacza dodatnią pół-płaszczyznę. Ponieważ standaryzowałeś drzewo tak, że jedno dziecko trzyma obiekty na dodatniej połowie, a reszta na negatywu, to tylko kwestia tego, co jest najważniejsze. Gdy Side = 0, obiekt znajduje się na prawdziwej płaszczyźnie. W takim przypadku musisz arbitralnie określić, czy obiekty znajdujące się na płaszczyźnie mają się znajdować na dodatniej lub ujemnej stronie płaszczyzny. To naprawdę nie robi różnicy, ale musisz uczynić ją stałą w całym algorytmie.

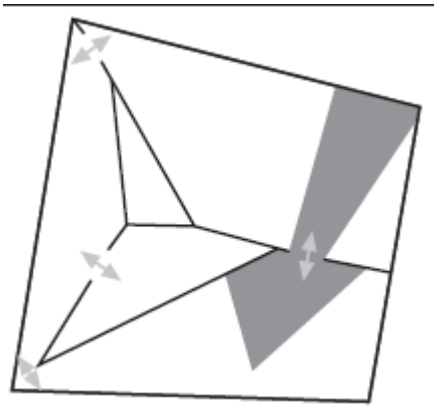
Podział na portal

Drzewa BSP są często używane w pomieszczeniach zamkniętych, ale mają również zastosowanie do światów zewnętrznych, jeśli ustali się prawidłową płaszczyznę separacji, która optymalizuje drzewo. Z drugiej strony, niektóre gry wolą trzymać wszystkie swoje szkielety w swoich szafach i pozostać w domu. Doom to czysta gra, a Quake 1 i 2 również pasują do tej kategorii. Quake 3 jest nieco pośrodku, ponieważ ma zarówno mapy zewnętrzne, jak i wewnętrzne, ale to tylko kombinacja kilku technik, które są tutaj podane. Podział na portal oferuje całkowicie nowy sposób myślenia. W poprzednich metodach świat był traktowany jako świat surowy (to znaczy bez jakiejkolwiek wcześniejszej hierarchii), a ty musiałeś podzielić ten świat za pomocą samolotów, aby stworzyć wydajną strukturę dla porównania. Idea portali jest prosta. Każdy pokój jest podzielony na rozłączny zestaw stref, z których wszystkie są połączone z innymi strefami za pośrednictwem podmiotu, który określa się jako portal. Możesz myśleć o portalu jako ciemnym lustrze, dzięki któremu możesz zobaczyć inny fragment świata, jak pokazano na rysunku 16.13.



To jak patrzenie przez okno, gdzie strefa definiuje każdą stronę okna. Ten pomysł najlepiej widać w Shadow Warrior. Ta gra nie była tak głośna jak gra, na której opierał się jej silnik (Duke Nukem 3D), ale dodatek do silnika był całkiem sprytny. Portale były już używane w grach takich jak Doom, ale tym, co wyróżniało Shadow Warrior było to, że mógłbyś mieć portal w suficie. Dzięki temu programiści mogli uzyskać efekt wielu poziomów dla tego samego <x, y>, po prostu przypisując portal do sufitu. To był

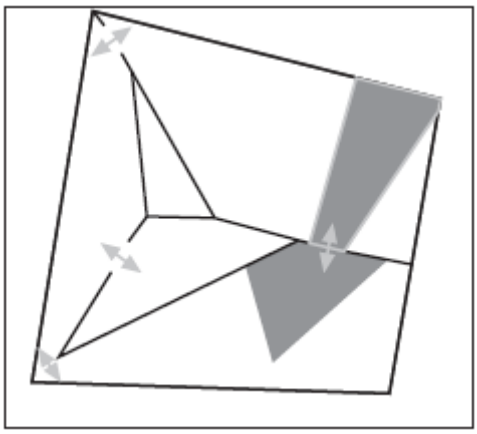
całkiem genialny pomysł. Jeśli pamiętasz Doom i Doom II, prawdopodobnie pamiętasz, że nie możesz być na dwóch różnych poziomach jednocześnie. Innymi słowy, mapa była naprawdę Mapa 2D z informacjami o wysokości w każdym miejscu i nie można znaleźć miejsca na mapie, które niosłyby więcej niż jedną wysokość (jeden sufit i jedno piętro). Technika 3D Realms pozwoliła na to za pomocą portali, zachowując przy tym szybkie renderowanie Dooma i pozwalając na dwa poziomy wysokości. Kluczem do zrozumienia portali jest sam portal i frustum. Sposób działania tej metody polega na tym, że najpierw musisz wyodrębnić strefę, w której znajduje się kamera. Po ustaleniu tego, musisz sprawdzić, które strefy są podłączone do tej strefy i sprawdzić, czy ta strefa jest widoczna. Jeśli tak, musisz powtarzać, dopóki nie będzie widać żadnych stref. Klucz do określenia, czy strefa jest widoczna, czy nie, pochodzi od samego ściętego stożka, jak to zostało zrobione wcześniej, ale piękno tej metody polega na tym, że można zmniejszyć wielkość ściętego stożka dla następnej strefy, jak pokazano na rysunku 16.14.



Ta zmniejszona wielkość stożka może znacznie zmniejszyć liczbę obiektów, które wymagają weryfikacji. Jedyną trudną częścią jest to, że ten nowy stożek może nie być w takiej formie, w jakiej by się spodziewał. W rzeczywistości, łatwiej jest uznać ten stożek za zbiór niezależnych płaszczyzn.

Przycinanie bryły ściętej na portalu

Zanim spuścicie bryłę ściętą do portalu, powinniście najpierw przyjrzeć się bliżej portalowi. Aby uczynić prostym, portal powinien być zdefiniowany jako wypukły wielokąt. W części 15 zobaczyłeś, jak przyjemnie jest pracować z wypukłym stożkiem ściętym, ponieważ możesz łatwo sprawdzić, czy obiekt znajduje się wewnątrz czy na zewnątrz. W rzeczywistości, sekcja wypukłego zaciemnienia cieni z 11 będzie bardzo przydatna tutaj, ponieważ pomysł jest prawie taki sam. Sam portal wypukły powinien być przechowywany w ściśle określonej kolejności. Aby uprościć sprawę, po prostu wybierzmy ją w kolejności przeciwnej do ruchu wskazówek zegara. Ważne jest, aby wybrać zamówienie na portal wypukły, ponieważ chcesz wiedzieć, która strona portalu zawiera następną strefę do przetestowania. Jeśli przycinasz ściętą bryłę do portalu, prawdopodobnie chcesz usunąć objętość ściętego stożka, która znajdowała się w początkowej sekcji, i zachować tylko objętość ścięgna, która znajduje się w następnej strefie. Przypomnijmy, że sam ścięty kawałek składa się z ośmiu płaszczyzn: jednego w pobliżu, jednego daleko i czterech płaszczyzn ograniczających objętość z każdej strony. W systemie portalowym dalekie i bliskie płaszczyzny to banalne płaszczyzny do przycinania. Jeśli przenosisz się z jednej strefy do drugiej, dalsza płaszczyzna nie powinna zostać zmieniona. Z drugiej strony, płaszczyzna bliska zostanie przesunięta do przodu tak, aby pasowała do płaszczyzny opisanej przez wypukły wielokąt tworzący portal, jak pokazano na rysunku 16.15.

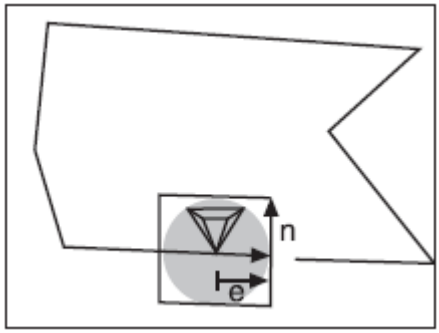


Tak więc te dwie płaszczyzny są dość łatwe w obsłudze. Najtrudniejszą częścią jest obsługa bocznych samolotów. Są to płaszczyzny, które najprawdopodobniej zostaną obcięte i zrewidowane przez portal. Początkowo jest to typowy zestaw czterech samolotów, które bardzo dobrze znasz. Ostatecznie stanie się to prawdopodobnie zbiorem płaszczyzn, więc powinieneś posiadać dynamiczną strukturę, która pozwala śledzić zestaw samolotów. Zdefiniuj zestaw wierzchołków, które tworzą portal jako vi. Pomysł polega na przycięciu wielokąta każdą płaszczyzną w zestawie ściętego stożka. Jeśli zastosujesz ten proces obcinania do każdej pojedynczej płaszczyzny, skutecznie przyciśniesz wypukły wielokąt na planie płaszczyzn opisujących ścięty stożek. Po przycięciu wypukłego wielokąta portalu ustaliłeś nowy wypukły wielokąt, który opisuje okno, przez które widać kolejną strefę. Okno samo w sobie nie jest zbyt użyteczne; musisz zbudować stożek ścięty. Ale już nauczyłeś się budować stożek ścięty metodą wypukłego cienia, więc ten proces jest bułka z masłem. Obcinanie odbywa się za pomocą metody obcinania trójkątów z części 14, a regenerację ściętego drzewa wykonuje się za pomocą wypukłej metody cienia okluzyjnego z części 15. Zapewni to kolejny zestaw płaszczyzn (mniejszy lub równy obszarowi do poprzedniego stożka ściętego) . W tym momencie możesz zastosować proces rekursywnie w strefach wchodzących w skład nowej odwiedzanej strefy. Oczywistym pytaniem jest, kiedy przestajesz? Cóż, to jest łatwa część. Zatrzymujesz się, gdy stożek ścięty jest całkowicie obcięty (nie ma wypukłych wierzchołków wielokąta) lub gdy nie są dostępne żadne inne strefy.

Potencjalnie widoczny zestaw (PVS)

Zazwyczaj, ponieważ twój świat jest w jakiś sposób połączony, możesz zweryfikować cały szereg portali, które są technicznie niemożliwe do wyświetlenia z danej strefy. Przy poprzednim algorytmie nadal trzeba przejść przez każdy portal w strefie i sprawdzić, czy jest on widoczny. Wymaga to wielu bezużytecznych obliczeń, jeśli już wiesz, że dany zestaw stref nigdy nie będzie widoczny z określonego pomieszczenia, niezależnie od kąta. W związku z tym ideą PVS jest zbudowanie zestawu stref potencjalnie widocznych dla danej strefy. Pomysł jest dość prosty. Powinieneś zachować listę stref, które są potencjalnie widoczne pod dowolnym kątem w pokoju, a kiedy przyjdzie czas na poszukiwanie sąsiednich stref, po prostu sprawdź na tej liście, aby określić, które strefy są widoczne. Brzmi prosto, prawda? Najtrudniejszą częścią jest uzyskanie tej listy stref. Najprostszym sposobem jest zrobienie tego ręcznie. Często jest jednak praktyczne, aby skonfigurować zautomatyzowany sposób osiągnięcia tego celu. Jest na to wiele opcji, ale najlepszym rezultatem jest zbudowanie stożka ściętego o 180 stopni (pole) i sprawdzenie widoczności z danej strefy poprzez umieszczenie kamery bardzo blisko wierzchołków portalu. Jeśli przesuniesz kamerę bardzo blisko pierwszego wierzchołka portalu i uzyskasz stopę ścięcia o 180 stopni, będziesz w stanie uchwycić każdą inną strefę, która została ci udostępniona, przechodząc przez algorytm i sprawdzając, czy możesz przejść do następny pokój (to znaczy, jeśli stożek ścięty nie jest pusty po przycięciu). Umieszczenie kamery na wierzchołku portali jest

tutaj kluczowe, ponieważ jest to najbliższe miejsce w pierwszej strefie, które można zobaczyć w drugiej strefie. Ponieważ jest najbliżej drugiej strefy, jest to najszerszy widok z tego miejsca. Strzała ścięta musi mieć kąt 180 stopni, głównie dlatego, że upraszcza to problem ze znalezieniem kąta stożka ścięgna. Jeśli stożek ścięty nie miał kąta 180 stopni, tworząc w ten sposób pole, musielibyśmy ustalić zestaw kątów do przetestowania. Ale z kątem 180 stopni pokrywasz wszystkie możliwe kąty widoczne w strefie. Wszystko, co musisz zrobić, to upewnić się, że najbliższa płaszczyzna stożka ściętego jest wyrównana z płaszczyzną portalu; reszta to tylko ortogonalny zasięg stożka ściętego. Idealnie, aby znaleźć skrzynkę taką, że jeśli obrócisz stożek ścięty pod dowolnym kątem, pudełko zawsze będzie zawierało stożek ścięty. Możesz znaleźć ten zakres, jeśli obliczasz przecięcie trzech prostopadłych płaszczyzn ze stożka ściętego. Na przykład, obliczając przecięcie dalekiej płaszczyzny z dowolną płaszczyzną boczną i płaszczyzną górną / dolną, otrzymamy pojedynczy punkt przecięcia. Jeśli obliczysz odległość między tym punktem a środkiem kamery, uzyskasz najdłuższy zasięg, jaki kiedykolwiek osiągniesz. W tym momencie jest prostą sprawą budowania pudełka wyrównanego z płaszczyzną portalu, która rozciąga się we wszystkich kierunkach. Można zbudować kulę, ale przycinanie do niej będzie większym bólem niż radzenie sobie z zestawem płaszczyzn. Aby uzyskać dwie brakujące płaszczyzny, można je po prostu zbudować za pomocą procesu ortogonalizacji Grama Schmidta. Rysunek 16.16 ilustruje, co zostało tutaj zrobione.



Portale jako dywizja wielkoskalowa

W większości wdrożeń portale są używane do dzielenia świata na dużą skalę. Na przykład, możesz użyć portali do strefy różnych pomieszczeń w grze takiej jak Quake. Zazwyczaj same pokoje są dość skomplikowane. Na przykład możesz dodać kilka obiektów do pomieszczenia, renderować bardziej eleganckie ściany i ogólnie nadać temu miejscu trochę stylu. To znacznie zmienia geometrię i nierozsądnie jest oddzielić wszystkie te małe szczegóły na portale, w których jedna duża krawędź jest portalem. W takich przypadkach pomieszczenia te powinny być traktowane jako strefy i powinny mieć powiązaną strukturę podziału (na przykład BSP na pokój). Piękno portali polega na tym, że pozwalają na dzielenie pokoi według własnego uznania. "Okna" przez inne pokoje są naprawdę tam, gdzie mogą zoptymalizować rzeczy dla ciebie. Ogólnie jest to świetna optymalizacja. Zazwyczaj pomieszczenie nie będzie miało tak wielu obiektów okluzyjnych, więc inna struktura podziału jest odpowiednia dla obiektów na tym poziomie. Tam, gdzie naprawdę można odstrzelić, wiele rzeczy znajduje się na korytarzach prowadzących do innych pomieszczeń, i tam właśnie portale są znakomite.