Instrukcja	Fizyka i animacja w grafice komputerowej
5	***Temat: Obsługa kolizji*** Przygotował: mgr inż. Tomasz Michno

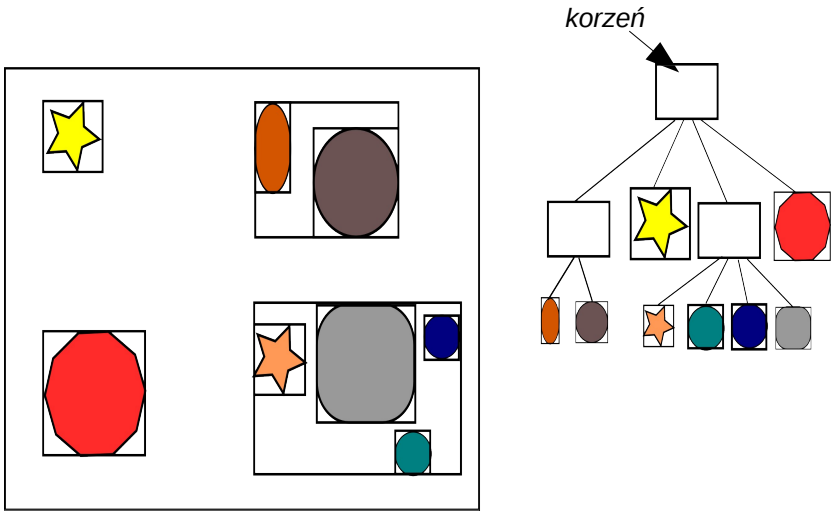
1 Wstęp

Obsługa kolizji jest jednym z najważniejszych elementów gier, czy symulacji. Zazwyczaj składa się z dwóch etapów: detekcji kolizji oraz odpowiedzi na zderzenie. Wykrywanie kolizji najczęściej realizowane jest z użyciem hierarchicznych brył ograniczających (bounding volume hierarchies). Ich podstawą jest bryła ograniczająca, która jest znacznie uproszczonym obiektem, w środku którego znajduje się ograniczany, właściwy obiekt. Podczas renderowania bryła taka nie jest wyświetlana.

Wyróżniamy dwa rodzaje brył ograniczających składających się z prostokątów/prostopadłościanów:

- axis aligned bounding boxes (AABBs) – boki bryły wyrównane do osi układu
- oriented bounding boxes (OBBs) – bryła wyrównana do obiektu, tak aby jak najbardziej do niego przylegała

Najczęściej stosuje się hierarchiczną strukturę brył – korzeń obejmuje całą scenę, następnie obiekty grupowane są w węzły pośrednie, które z kolei dzielą się na kolejne węzły pośrednie, aż do sytuacji, w której w bryle znajduje się tylko jeden obiekt.



Jak widać na rysunku, struktura BVH ma charakter drzewa o nieustalonej maksymalnej liczbie potomków. Rozwiązanie to jest bardzo wydajne w przypadku testów wykrywających kolizje, jednak wymaga dużego nakładu podczas tworzenia. Z tego względu zazwyczaj budowanie drzewa wykonywane jest na początku programu, natomiast w przypadku późniejszych zmian w scenie, przebudowywany jest jedynie fragment drzewa.

Drugim etapem obsługi kolizji jest odpowiedź na zderzenie. Najczęściej bazuje ona na zasadzie zachowania pędu, która mówi: „suma wektorowa pędów wszystkich elementów układu izolowanego pozostaje stała” (pęd początkowy ciał jest równy ich pędowi końcowemu). Pęd definiowany jest jako:

$$\vec{p}=m\cdot\vec{v}$$

gdzie:

p – pęd (wektor)
m – masa ciała
v – prędkość ciała (wektor)

Z ruchem obiektu związana jest również energia kinetyczna (Ek), która określa stan układu w ruchu:

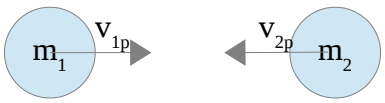
$$\vec{E}_k = \frac{m \cdot \vec{v}^2}{2}$$

Zderzenia dzielimy na:

- doskonale sprężyste
- doskonale niesprężyste

W przypadku ciał doskonale sprężystych nie występują żadne odkształcenia na powierzchni obiektów podczas zderzenia.

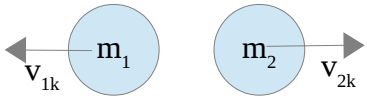
Przed zderzeniem



$$\vec{p}_p = m_1 \cdot \vec{v}_{1p} - m_2 \cdot \vec{v}_{2p}$$

$$\vec{E}_{kp} = \frac{m_1 \cdot \vec{v}_{1p}^{\,2}}{2} + \frac{m_2 \cdot \vec{v}_{2p}^{\,2}}{2}$$

Po zderzeniu



$$\vec{p}_k = m_1 \cdot \vec{v}_{1k} - m_2 \cdot \vec{v}_{2k}$$

$$\vec{E}_{kk} = \frac{m_1 \cdot \vec{v}_{1k}^{\,2}}{2} + \frac{m_2 \cdot \vec{v}_{2k}^{\,2}}{2}$$

$$\vec{p}_p = \vec{p}_k$$
$$\vec{E}_{kp} = \vec{E}_{kk}$$

Zarówno przed, jak i po zderzeniu pęd układu oraz energia pozostaje taka sama (zmianie ulegają wektory prędkości).

W przypadku zderzeń doskonale niesprężystych pęd układu pozostaje taki sam przed, jak i po zderzeniu. Natomiast różnicą jest stan energii kinetycznej, której część zamienia się w ciepło.

2 Zadanie.

Napisz aplikację, która będzie symulowała zderzenia kul, np. bilardowych (zderzenia sprężyste), które będą odbijały się od siebie. Użytkownik powinien posiadać możliwość sterowania jedną z kul, która będzie służyła do rozpoczęcia odbić. W programie zastosuj dodatkowo nieruchome elementy prostokątne, w które również będą mogły uderzać kule. Zastosuj bryły ograniczające. Program może zostać wykonany w Allegro, OpenGL lub HTML5, w 2D lub 3D. Fizyka zderzeń może zostać uproszczona, jednak odbicia powinny wyglądać naturalnie (np. wektory ruchów przed i po zderzeniu powinny być zbliżone do wynikających z zasady zachowania pędu).