アニメーションと衝突判定

Game Programming B #11

向井智彦

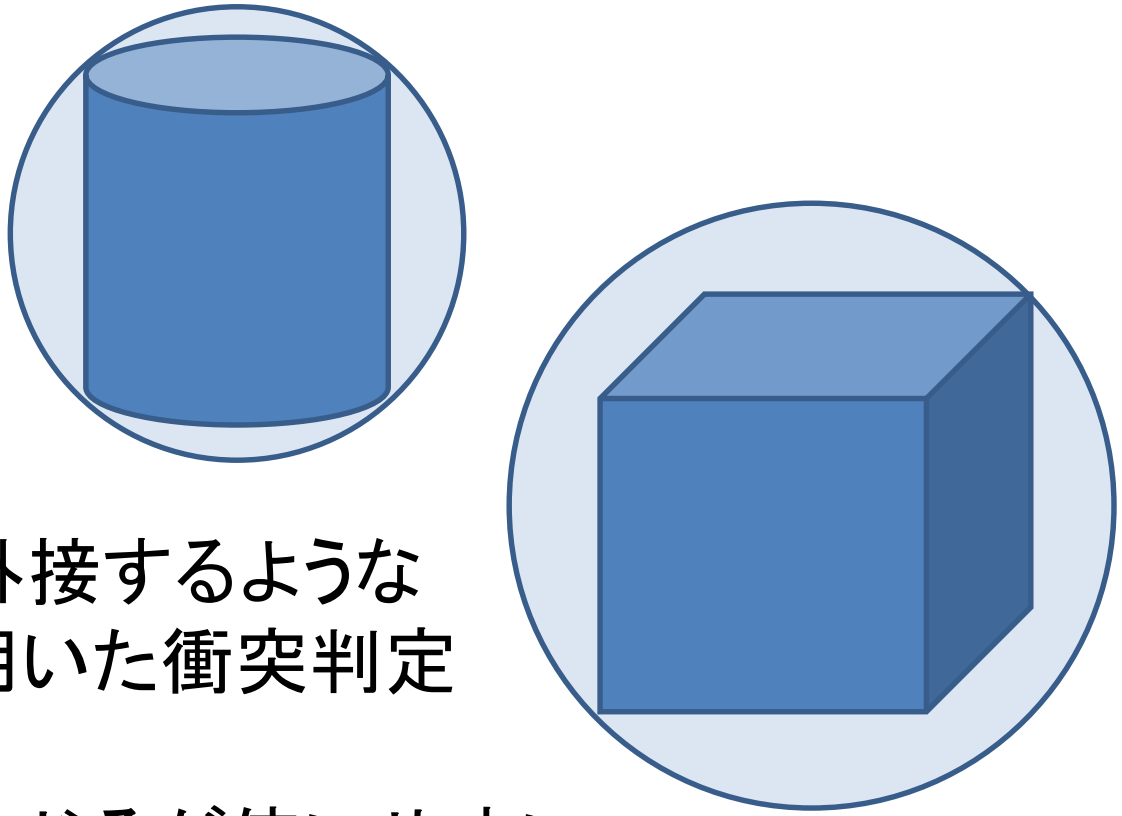
本日：衝突判定とアニメーション

1. 続・カメラエンティティの制作（演習）

2. 衝突判定
   - Bounding sphere

3. 相互作用のあるアニメーション
   - 衝突したら跳ね返る
   - 衝突したら消える
   - 衝突したら色が変わる

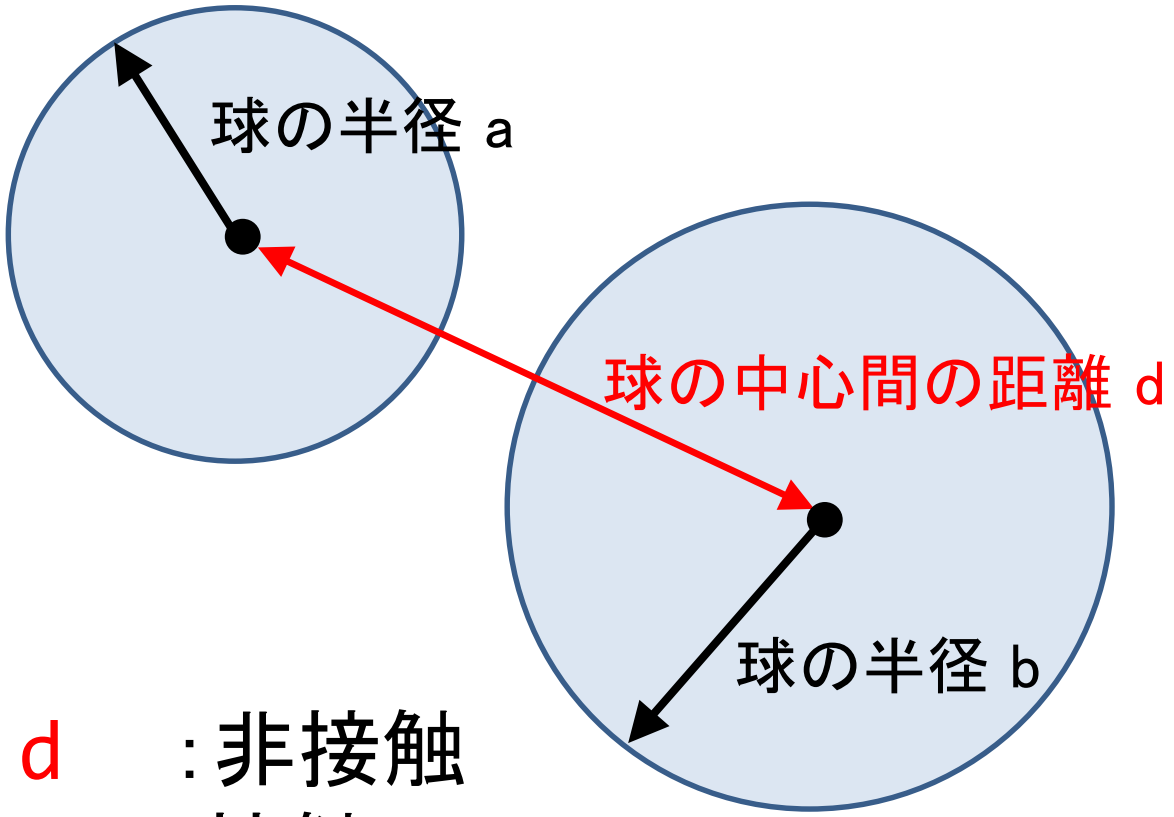
Bounding sphere



物体に外接するような
球体を用いた衝突判定

余白は生じるが使いやすい

単純な衝突判定：球同士



a + b < d　：非接触

a + b >= d　：接触

Bounding sphereの使い方

- 衝突判定を行いたいエンティティクラスに

  double BoundingRadius() const;

  glm::vec3 WorldPosition() const;

  のようなメンバ関数を追加（対応する変数も）
- SimpleGame::Update 内で半径および中心座標を取得し、衝突判定

- Game-Programming-B/bounding を参照

マウスカーソルとの衝突判定

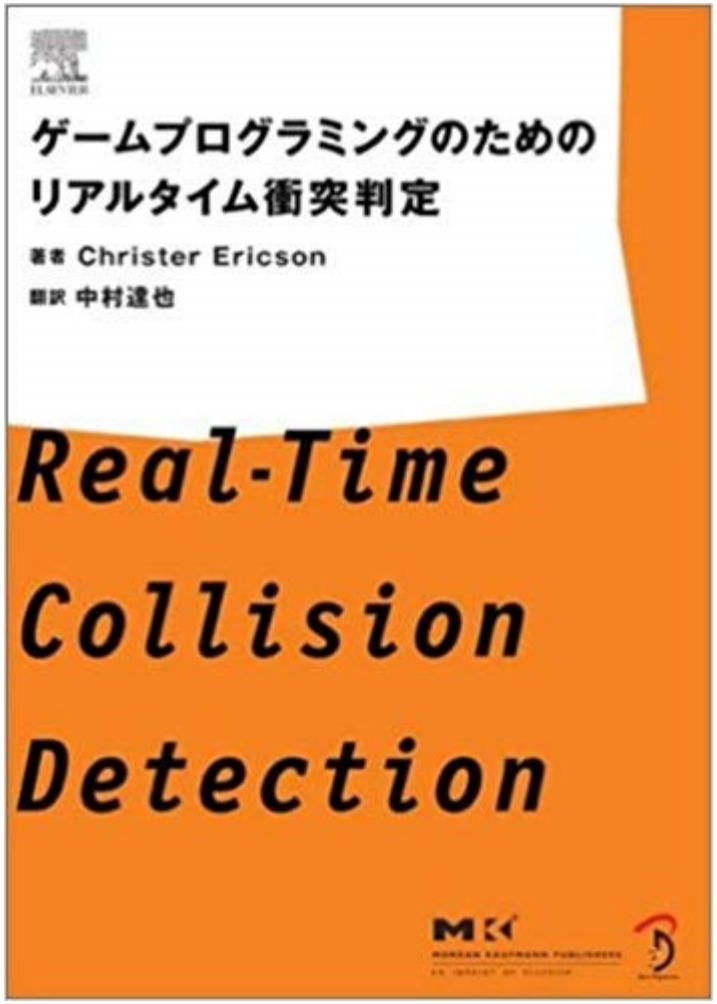
- Bounding sphere と Ray（レイ、光線）の衝突判定の応用

- Game-Programming-B/interactionブランチの下記ファイルを参照
  - SimpleGame.cpp
  - SphereEntity.h
  - SphereEntity.cpp

他の衝突判定

- 衝突判定の基本から
  プログラムコードも交えて
  網羅的に書かれた良書

- 3次元幾何学の最低限の
  知識は必要
  - ベクトル、内積、外積



細かいテクニック

- 描画される主エンティティに、見えない子エンティティをAddChildし、それぞれにBounding sphereを設定
  - Game-Programming-B/bounding2 を参照
- つまり、1つの主エンティティに、複数の当たり判定を追加
  - 無敵部分と弱点がある敵キャラクター
  - 複数のパーツが動くようなキャラクター

演習

- 縦横3×3、計9個並べられた球体を、マウスクリック順に消すプログラムを作成

1. カメラを真っ正面（+Z軸上）に移動
2. XY平面上に球体エンティティを9個配置
3. 球体エンティティの可視・不可視を操作するメンバ関数を追加（ヒント: Visibility）
4. マウスクリック時に衝突判定、可視性を操作するようなコードをSimpleGame::Updateに記述

最終作品に向けて

- 3DCGを活用したインタラクティブゲーム
  - 3D迷路ゲーム
  - 3Dモグラ叩きゲーム
  - 3Dシューティングゲーム
  - などなど
- 3D描画を用いた2次元ゲームも可
  - 「ゲームプログラミングA」のブロック崩しやシューティングゲームの素材を3D化するなど