情報メディア実験A

# 物理エンジンを使ったアプリケーション開発

筑波大学情報学群
情報メディア創成学類
藤澤誠

# 実験スケジュール

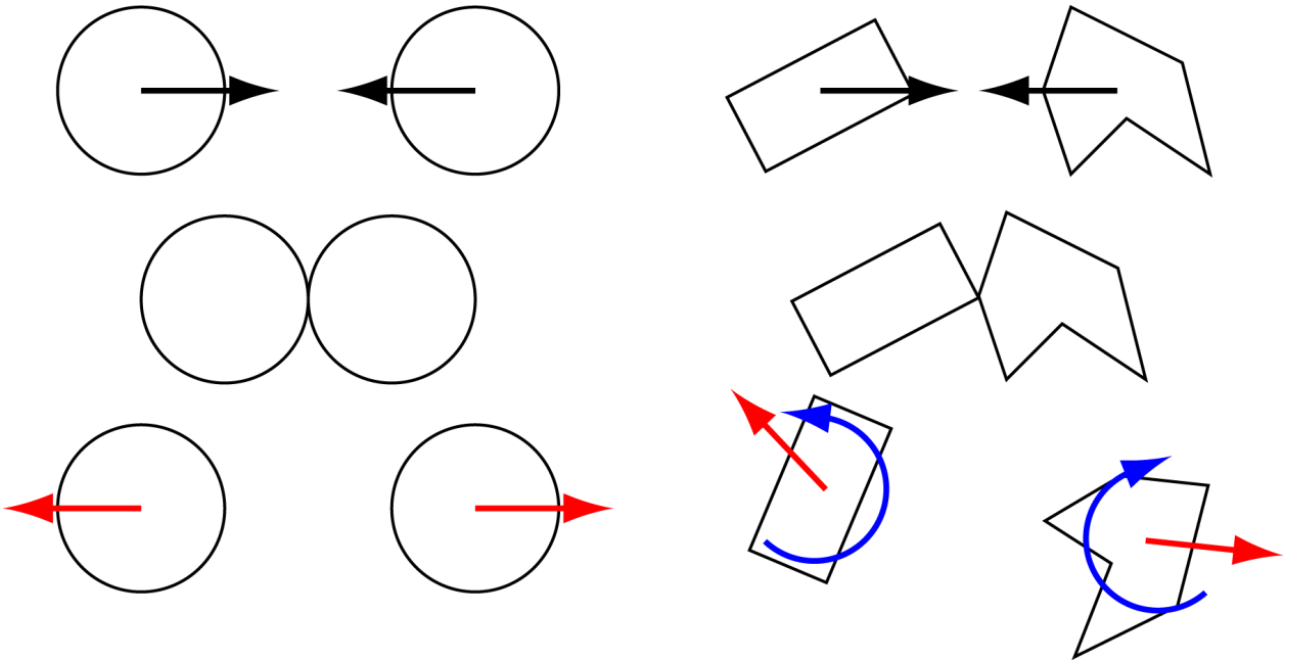
- 毎週 水3,4限＆金5,6限
- テーマ内スケジュール
  1. ガイダンス＆事前知識(C++)説明：4/16
  2. 物理シミュレーションとは?：4/21, 23, 28
  3. 物理エンジンとは？ ：4/30, 5/7, 12
  4. 剛体間の衝突判定,衝突応答：5/14, 21, 26, 28
  5. 剛体間リンク：6/2, 4, 9, 11
  6. 3Dモデル読み込みと弾性体：6/16, 18, 23, 25
  7. アプリケーション開発：6/30, 7/2, 7, 9, 14, 16, 21
  8. 成果発表会：7/28

赤太字は説明回，7/22は発表準備＆レポート作成回
5/7(金)は水曜授業日，5/19(水)は春A期末試験日

# 剛体シミュレーション

物体を**剛体(変形しない物体)**と仮定して，ニュートン力学に基づきその動きを計算

剛体同士が衝突したら？



# 剛体シミュレーションの例

[Weinstein et al. IEEE TVCG 2005]

たくさんの剛体の衝突や剛体同士の接続などを考えなければならない！

# 剛体シミュレーション

- **衝突検出**(Collision Detection : CD)
  - すべてのオブジェクト同士で衝突判定をする必要がある. →非効率的
  - 高速化のための様々な手法がある

- **衝突応答**(Collision response)

  剛体同士が衝突した後の運動計算
  - 衝突角度と物体の反発係数に従い, 速度ベクトルを変更する
  - 最も単純なケース:ボールが平面に垂直に衝突

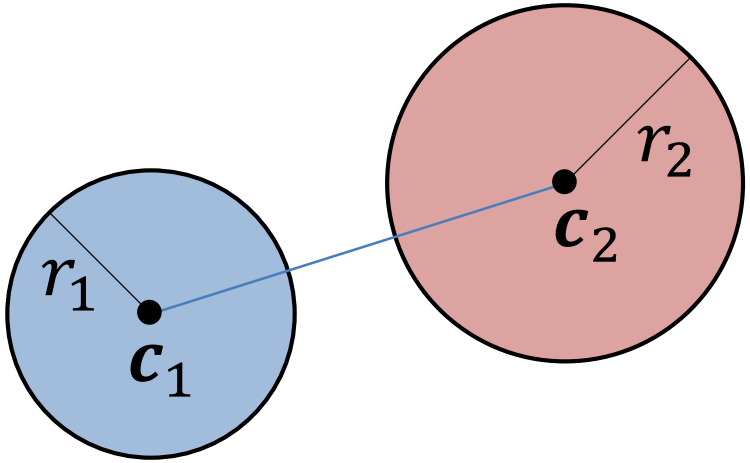
$$\boldsymbol{v}_r = -e\boldsymbol{v}$$

# 衝突検出

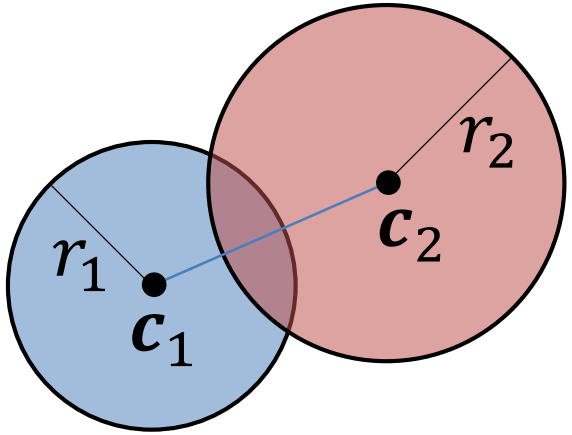
- 物体同士の衝突検出：最も単純な場合
  球同士の衝突

(実際には2乗距離を用いる)

$|\boldsymbol{c}_1 - \boldsymbol{c}_2| > r_1 + r_2$




$|\boldsymbol{c}_1 - \boldsymbol{c}_2| \leq r_1 + r_2$

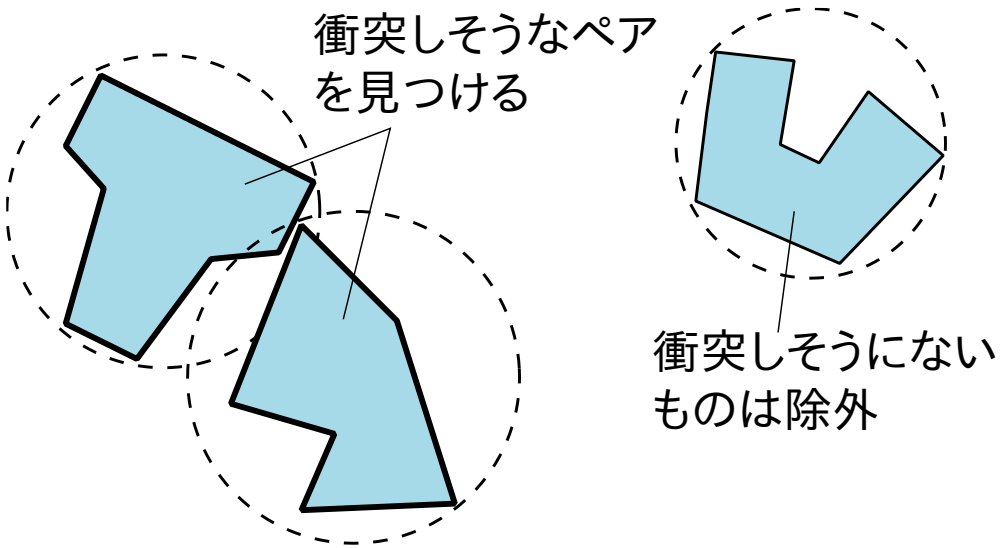



$N$オブジェクト同士の組み合わせ：$\frac{N^2-N}{2}$回の計算 $O(N^2)$

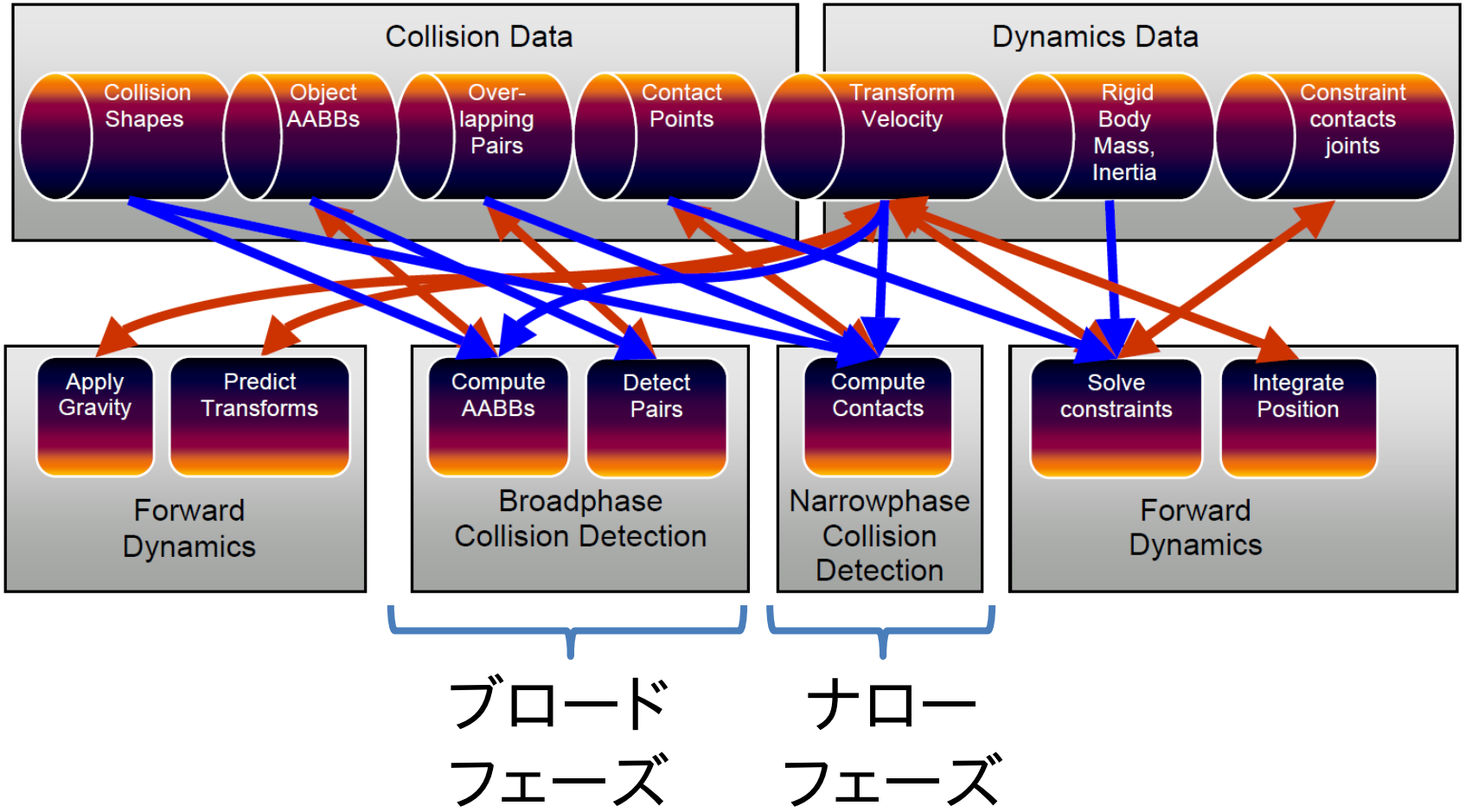
⟹ **もっと複雑な形状だと？**

# 衝突検出

- ブロードフェーズとナローフェーズ
  - ブロードフェーズ(Broad-phase)
    大まかな形状で衝突の可能性のあるペアを探索
  - ナローフェーズ(Narrow-phase)
    ブロードフェーズで見つかったペアに対して
    正確に衝突判定

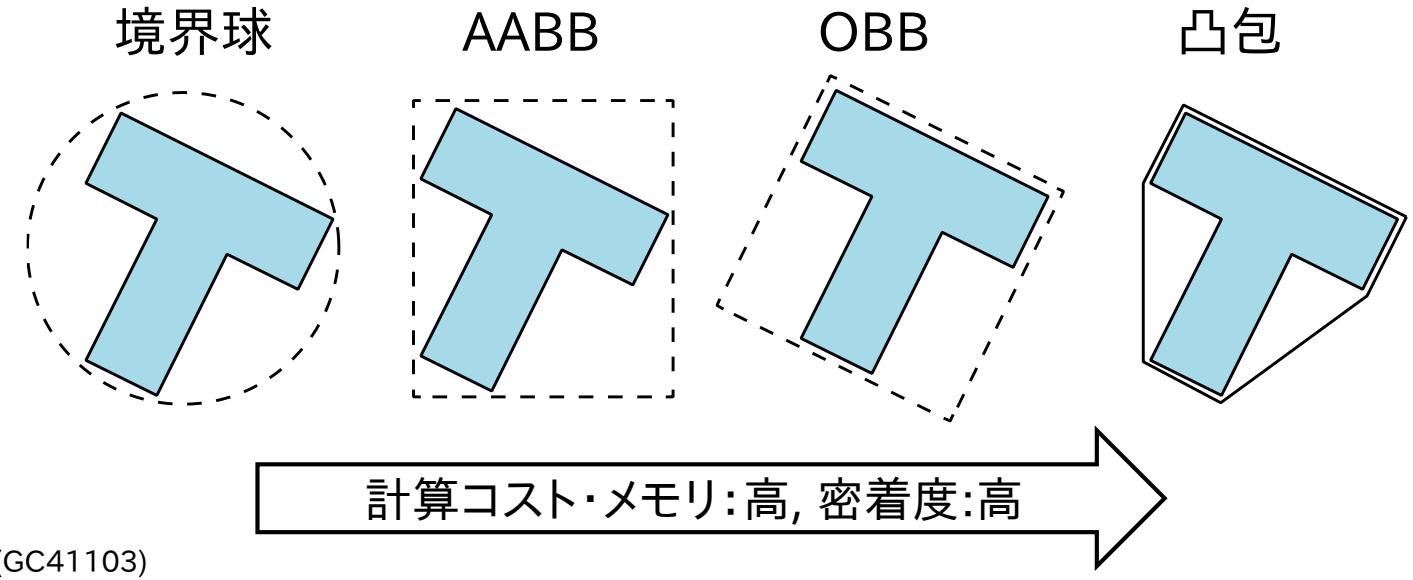


# Bulletの処理の流れ



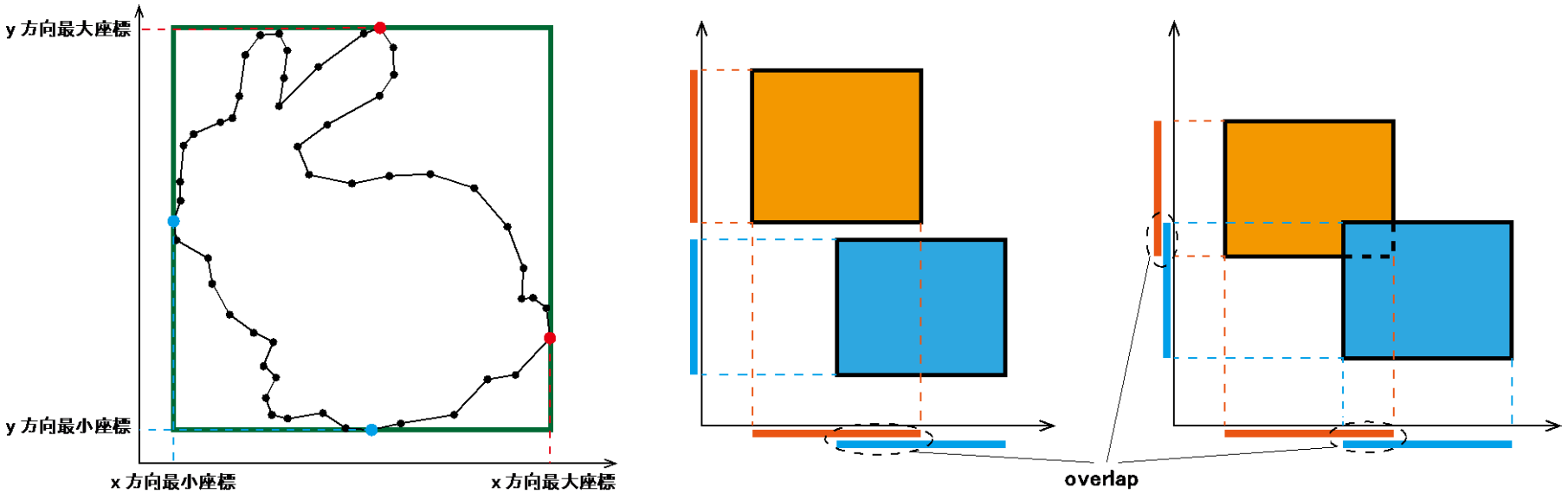
# 衝突検出:ブロードフェーズ

- 境界ボリューム(Boundary Volume)
  - 境界ボリュームに求められる条件
    - 交差判定コスト:低
    - ボリューム作成コスト:低
    - 必要メモリ量:低
    - 元形状に対する密着度:高
  - 代表的な境界ボリューム



# 衝突検出:ブロードフェーズ

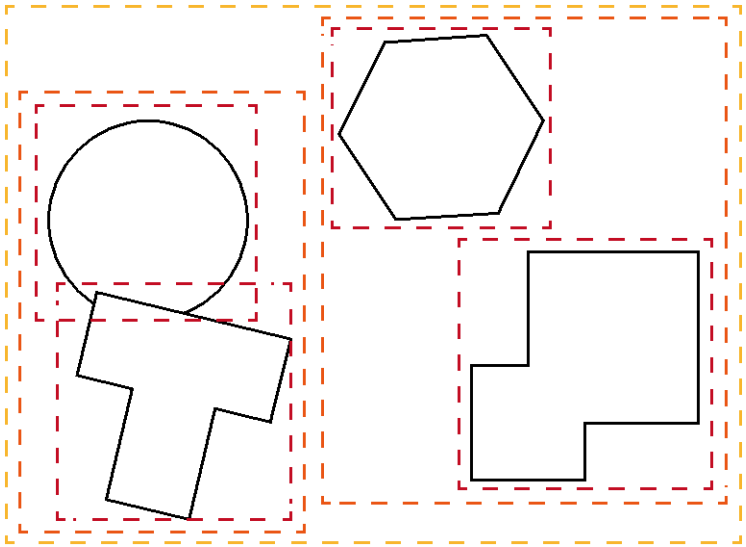
- 境界ボリュームによる衝突判定の例:AABB
  各辺が軸に平行な直方体 (Axis-Aligned Bounding Box)
  - 境界ボリュームの中で最も一般的
  - 重なり判定：最小, 最大座標の比較だけ

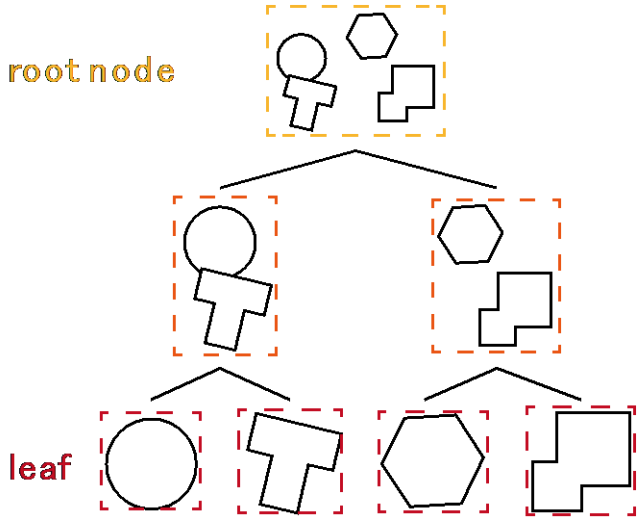



AABBの定義と重なり判定

# 衝突検出：ブロードフェーズ

- AABB木
  すべての物体を含むAABB ⇒ ルートノード
  空間を2分してAABBを求める ⇒ 子ノード
  ルートノードから順番に重なりを調べていく $O(N \log N)$

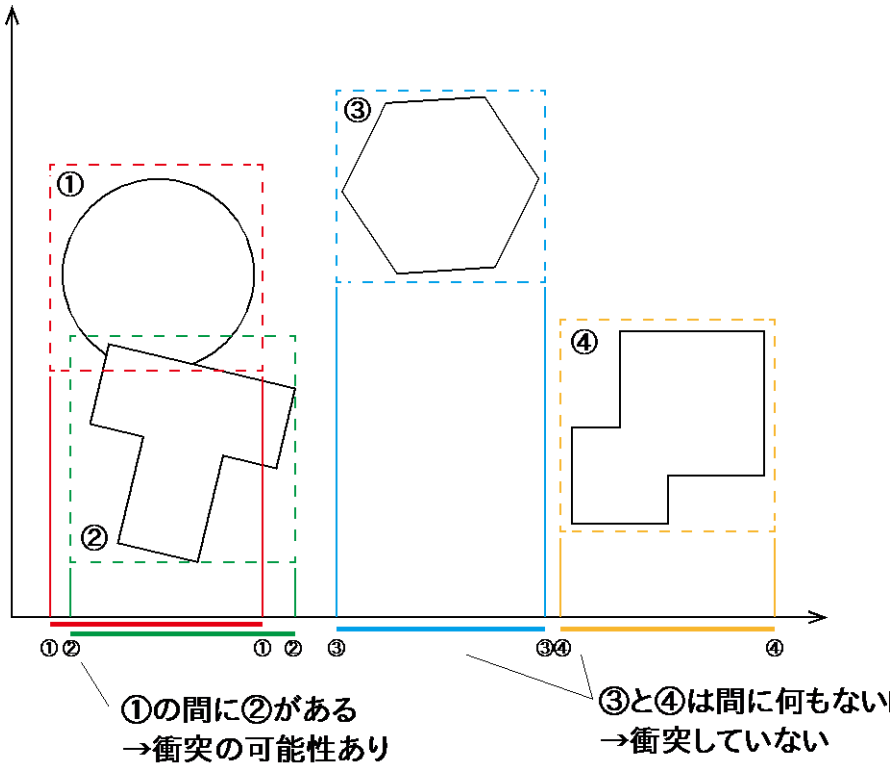




# 衝突検出:ブロードフェーズ

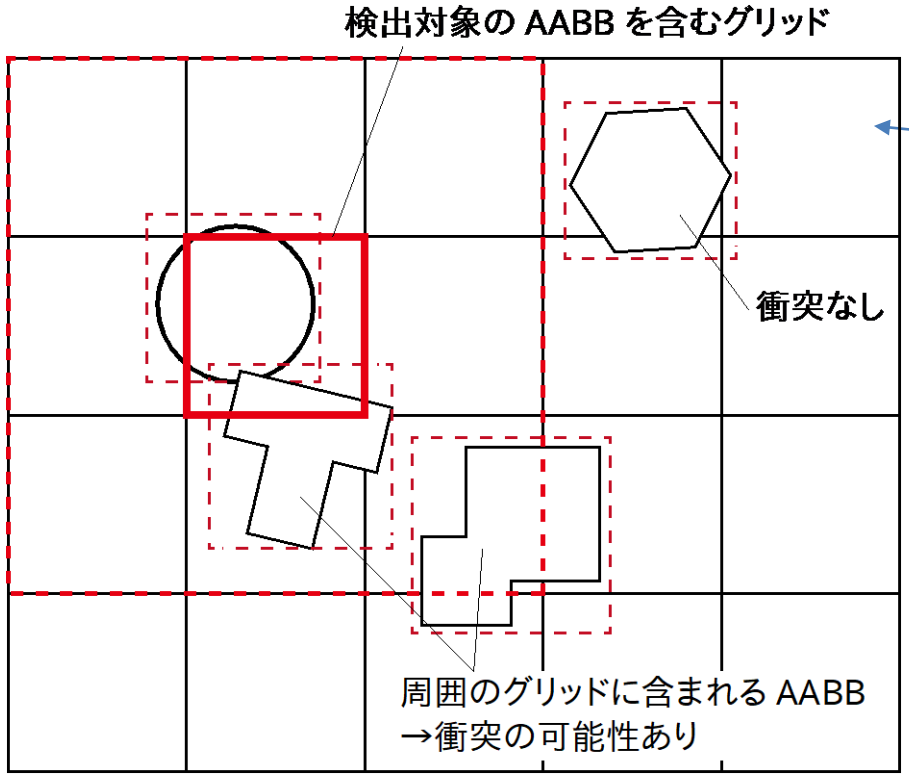
- 3Dスィープ&プルーン(sweep and prune)
  1. AABBを軸に投影して端点座標値を計算
  2. 端点座標値をソート(挿入ソート:$O(N)$～$O(N^2)$など)
  3. 端点座標を比較
     - 同じAABBの端点が並んでいる
       ⇒ 衝突なし
     - 別のAABB端点が並んでいる
       ⇒ 衝突の可能性あり

$x, y, z$軸すべてで1～3を行う



# 衝突検出:ブロードフェーズ

- グリッド構造
  空間を均一グリッドで分割し, オブジェクトが属するセルの周囲セルを探索




分割幅は最大オブジェクトサイズなどから決定

# 衝突検出:ブロードフェーズ

- Bulletでの設定方法
  Worldの設定時に指定

```
// ブロードフェーズ法の設定(Dynamic AABB tree method)
btDbvtBroadphase *broadphase = new btDbvtBroadphase();
// Bulletのワールド作成
g_pDynamicsWorld = new btDiscreteDynamicsWorld(dispatcher, broadphase,
                                               solver, config);
```

ここまで説明した3種類に対応

AABB木 : btDbvtBroadphase,
スイープ＆プルーン : btAxisSweep3 or bt32BitAxisSweep3
グリッド法 : btCudaBroadphase (GPUを使うので実習室PCでは使えない?)

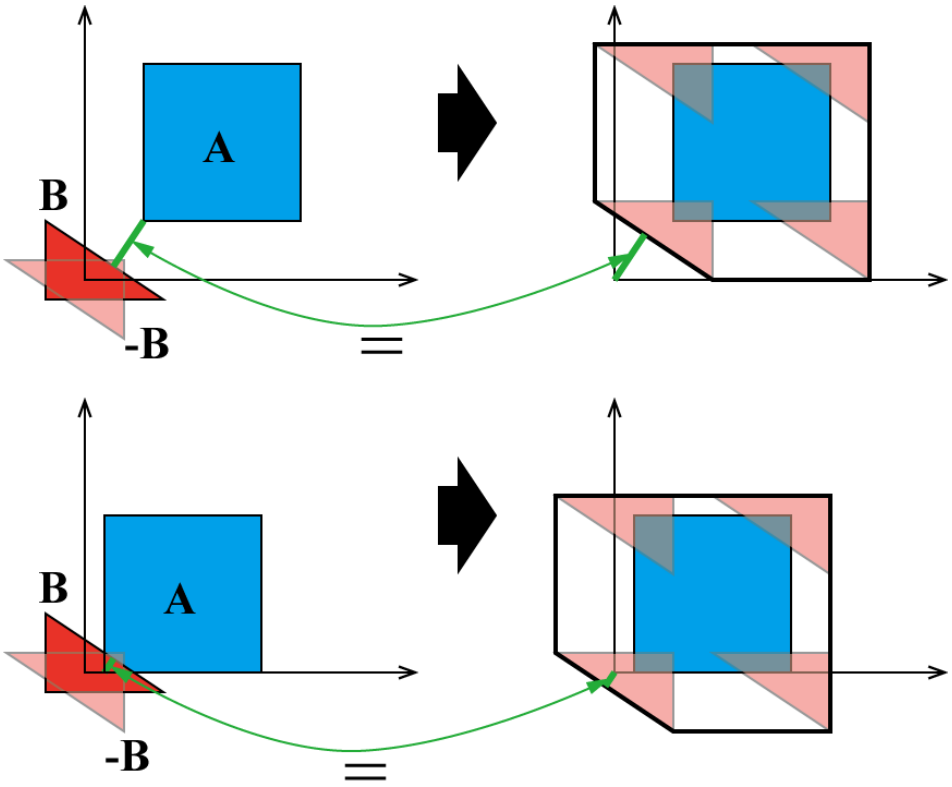
(全探索 : btSimpleBroadphase (デバッグ，テスト用))

# 衝突検出:ナローフェーズ

- ナローフェーズ
  ブロードフェーズで検出したペアの正確な衝突判定を行う (単純な例:球同士の衝突検出)
- GJKアルゴリズム

  「ミンコフスキー和
  $A \oplus (-B)$が原点を含むと重なっている」

  - GJKではミンコフスキー和を明示的に計算する代わりにサポート写像を用いる
  - 凸多角形/多面体のみ



# 衝突検出:ナローフェーズ

- Bulletが対応する形状

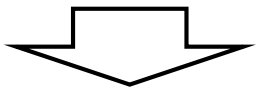
| | box | sphere | convex,cylinder cone,capsule | compound | triangle mesh |
|---|---|---|---|---|---|
| box | boxbox | spherebox | gjk | compound | concaveconvex |
| sphere | spherebox | spheresphere | gjk | compound | concaveconvex |
| convex, cylinder, cone, capsule | gjk | gjk | gjk or SAT | compound | concaveconvex |
| compound | compound | compound | compound | compound | compound |
| triangle mesh | concaveconvex | concaveconvex | concaveconvex | compound | gimpact |

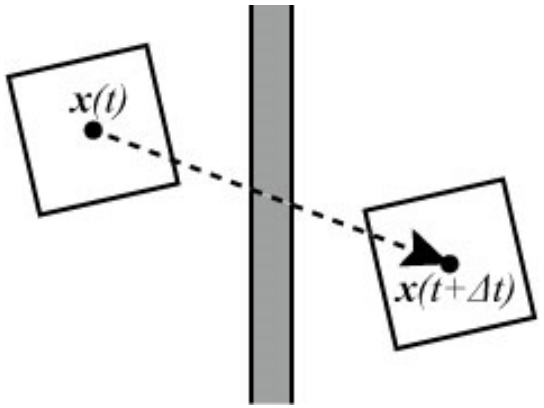
直方体
球
凸形状
円筒
円錐
カプセル
複合形状
三角形ポリゴン

ナローフェーズに使われるアルゴリズム

# 衝突検出:CCD

- Continuous Collision Detection (CCD)
  $\Delta t$ごとに衝突判定をすると，動きの
  速い物体がすり抜けることがある

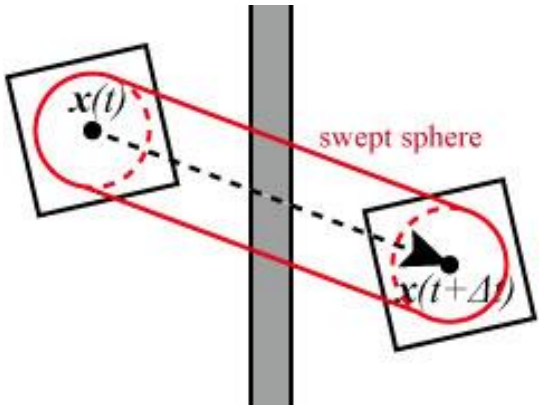





スイープ形状を作成，衝突判定する

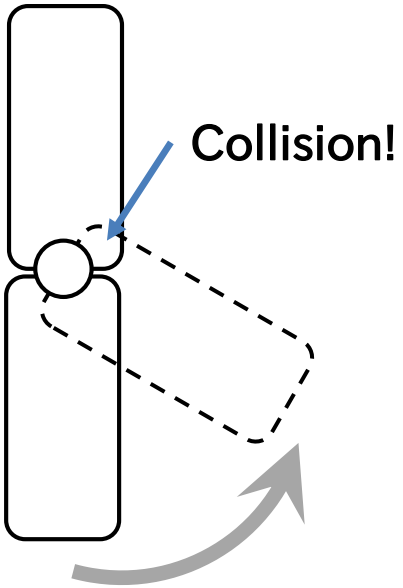
- Bulletでの設定方法
  btRigidBodyに対して設定

```
btRigidBody *body = ···
body->setCcdSweptSphereRadius(0.5);
body->setCcdMotionThreshold(0.05);
```



# 衝突検出:Collision Filtering

- Collision Filtering
  衝突判定する対象を設定できる機能
  ⇒ 物体同士をすり抜けさせたい場合に利用



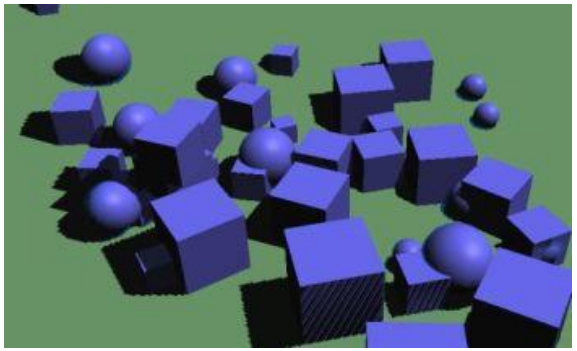

- Bulletでの設定方法
  WorldにbtRigidBodyを設定するときに
  自身のIDと衝突対象IDを設定

```
// 衝突応答のためのグループ
enum CollisionGroup{
    GROUP1 = 1,  // 0001
    GROUP2 = 2,  // 0010
    GROUP3 = 4   // 0100
};

world->addRigidBody(body1, GROUP1, GROUP1);
world->addRigidBody(body2, GROUP2, GROUP2|GROUP3);
```

IDとして2の累乗を使うと便利

自身のID　衝突対象ID



球同士,立方体同士でしか衝突判定しないように設定した例

# 剛体シミュレーション

- 衝突検出(Collision Detection : CD)
  - すべてのオブジェクト同士で衝突判定をする必要がある. →非効率的
  - 高速化のための様々な手法がある

- **衝突応答**(Collision response)

  剛体同士が衝突した後の運動計算
  - 衝突角度と物体の反発係数に従い, 速度ベクトルを変更する
  - 最も単純なケース:ボールが平面に垂直に衝突

$$v_r = -ev$$

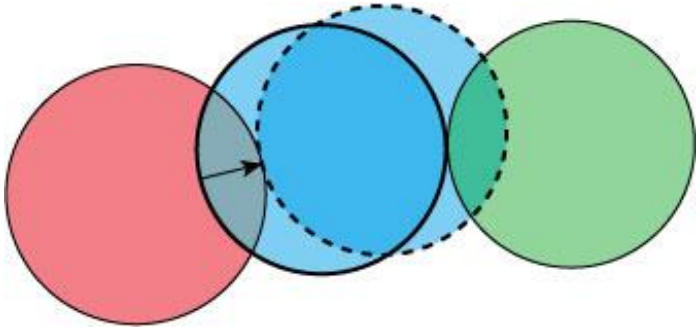
# 衝突応答:ペナルティ法

- めり込み処理の問題
  衝突検出は離散時間で行われるので, 物体同士がめり込むことがあり得る

  ⇩

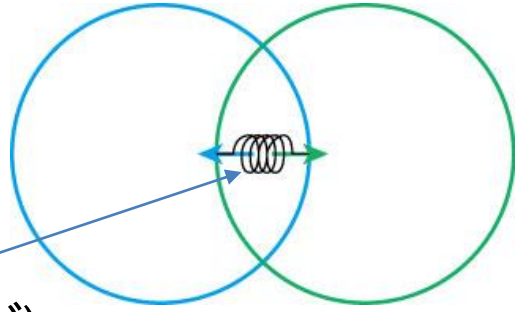
  すべてのめり込みを矛盾なく解決することは難しい

  ⇩



赤と緑の球が固定のとき, 青球のめり込みをどう解決するか?

**ペナルティ法**：めり込みを許す代わりにめり込み量に応じた力を与える
(bulletではSoft Constraintを用いている ⇒ 次回解説)



めり込み量に応じた反力
(バネを設定するようなイメージ)

# 衝突応答:反発係数と摩擦係数

- 物体同士が衝突したときにかかる力
  - 反発力：反発係数で設定

```
body1->setRestitution(0.9);
body2->setRestitution(0.5);
```

body1とbody2が当たった場合の反発係数は**両者を掛けた値**(0.9×0.5=0.45)となる

  - 摩擦力：摩擦係数で設定

```
body1->setFriction(0.8);
body2->setFriction(0.8);
```

body1とbody2が接触している場合の摩擦係数は**両者を掛けた値**(0.8×0.8=0.64)となる

# Bulletでの衝突検出＆衝突応答

ここからは各自の環境で
実際に作業

実験ページの「**3. 多数の剛体間の衝突判定, 衝突応答**」の**練習問題**を実際にやってみよう
(余裕のある人はoption課題もやってみよう!)