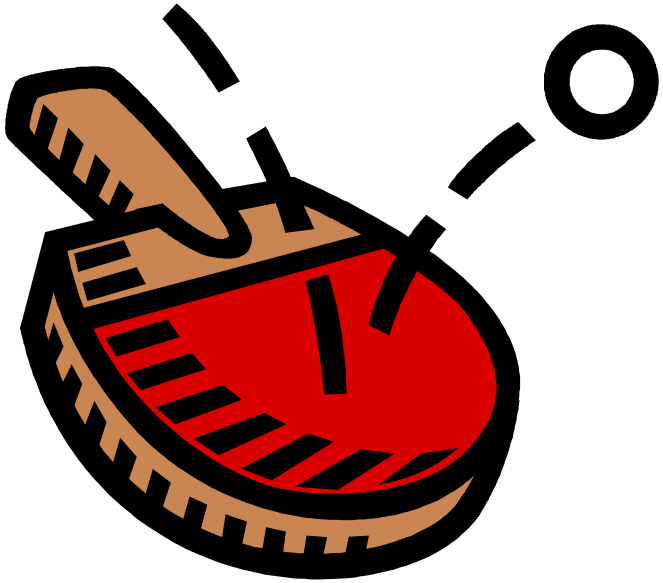
ボールを跳ね返らせる

2012/3/13 梶本（学部3年生への課題の解説）

はじめに



- 最終課題は次のようなものでした。
- 「自由落下運動するボールをリフティングしつづけるゲームを作ってください」
- もしかすると途方にくれているかもしれないので、「ボールを跳ね返らせる」ことについて少し解説します。
- 前提知識は高校の物理（の初歩の初歩）です。

ニュートンの運動方程式

- F=ma
  - は誰でも知っていると思います。
  - Fは力、mは質量、aは加速度、です。

  - でも、この式の意味は分かっていないかもしれません。
  - この式は、
  - 「力Fを、質量mの物体に加えると、加速度aが生じる」
  - という、
- 「因果関係」
  - を表したものです。

ニュートンの運動方程式

- 逆に、
- 「質量mの物体に、加速度aを生じさせる、得体のしれない何か」
- を
- 「力」
- と定義しよう、
- という意味でもあります。

- なぜなら、「力」というのは、
- 正体不明の、直接観察できない「概念」だからです。

ばね

- 「力はハカリで観察できるよ」
- と思うかもしれませんが、
- 実際には、例えばハカリが測定しているのは「バネの伸び」、つまり「変位量」です。
- 力「そのもの」が眼に見えることはありません。

- 話を元に戻します

ニュートン

- F=ma
- を使えば、物体に生じる加速度を計算できます。

- つまり、
- a=F/m
- です。

加速度⇒速度⇒位置

- 加速度が求まれば、それを積分して速度が求まります。

- $v(t) = v(t=0) + \int a(\tau)d\tau$　　(積分範囲は$\tau=0 \sim t$)

- 速度が求まれば、それを積分して位置が求まります。

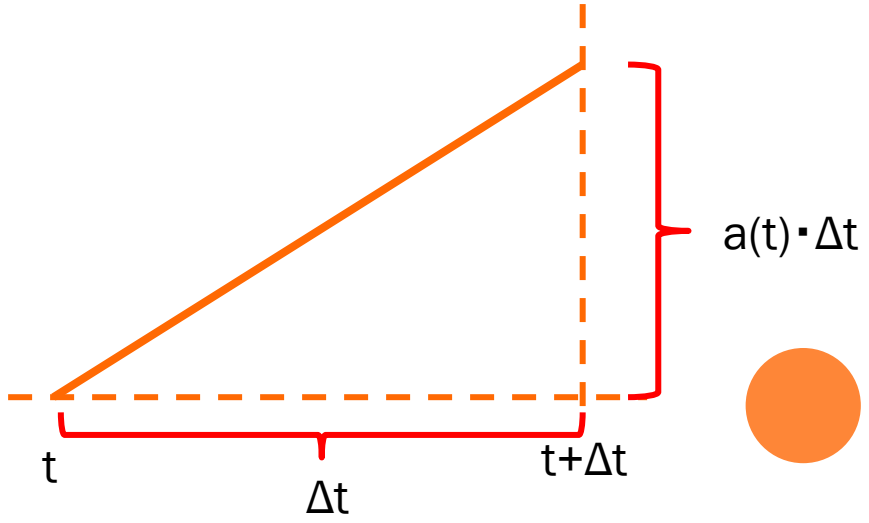
- $y(t) = y(t=0) + \int v(\tau)d\tau$　　(積分範囲は$\tau=0 \sim t$)

加速度⇒速度⇒位置（コンピュータ）

- 今考えたいのはコンピュータによるシミュレーションです。
- そこでは、「時間」を微小な区間に区切って計算していきます。

- ある時刻の加速度をa(t)、速度をv(t)、位置をy(t)、時間の区切りをΔtとすると、Δtの時間が経つと、次のような変化が生じます。

- $v(t+\Delta t) = v(t) + a(t)\cdot\Delta t$
- $y(t+\Delta t) = y(t) + v(t)\cdot\Delta t$



加速度⇒速度⇒位置（コンピュータ）

- プログラムの骨組みは、次のようなループになります。

```
v=0;     //速度の初期値
y=0;     //位置の初期値
While(1){
        F = ○○○;      //加わる力を求める
        a = F/m;       //力から加速度を得る
        v += a･Δt;     //加速度から速度を更新する
        y += v･Δt;     //速度から位置を更新する
}
```

あとは力さえ求まれば、「勝手に」状態は更新されていきます。

自由落下運動

- 今回の課題では、まずボールが自由落下するのでした。

- 自由落下運動する場面では、質量に比例した重力が加わります。

- $F = mg$

- ですから、加速度は次のようになります。

- $a = F/m = mg/m = g$

- 先ほどのプログラムは次のようになります。

```
v=0;        //速度の初期値
y=0;        //位置の初期値
While(1){
        a = g;                  //重力加速度
        v += a・Δt; //加速度から速度を更新する
        y += v・Δt; //速度から位置を更新する
}
```

別の考え

- もちろん我々は、「一定加速度の場合の位置の変化」の方程式を知っています。

- $y = \frac{1}{2} at^2$

- これを使ってプログラムすることも出来るでしょう。

```
t=0;
While(1){
        t++;
        y = 0.5 * g*t *t;
}
```

これも一応正しい結果を返します。正解です

別の考え

```
t=0;
While(1){
        t++;
        y = 0.5 * g*t *t;
}
```

- でも、我々はこのやり方を嫌います。

- なぜなら「$1/2\ at^2$」というのは、
- 一定の加速度、すなわち一定の力が加わっている特殊な状況でないと成立しないからです。

加速度⇒速度⇒位置（コンピュータ）

```
v=0;      //速度の初期値
y=0;      //位置の初期値
While(1){
          F = ○○○;        //加わる力を求める
          a = F/m;         //力から加速度を得る
          v += a･Δt;       //加速度から速度を更新する
          y += v･Δt;       //速度から位置を更新する
}
```

- それに対してこちらの方法は、
- 力から加速度、加速度から速度、速度から位置が得られる、
- という、「普遍の真理」だけを書いています。

- 「極小の法則」だけを書くと、「現象」が「勝手に」現れる、というのがシミュレーションの力です。
- （なお数式であらかじめ解いておいた方がよい場合もあります）

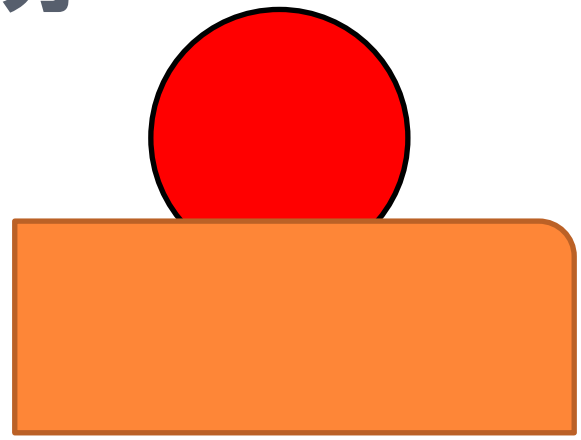
跳ね返り

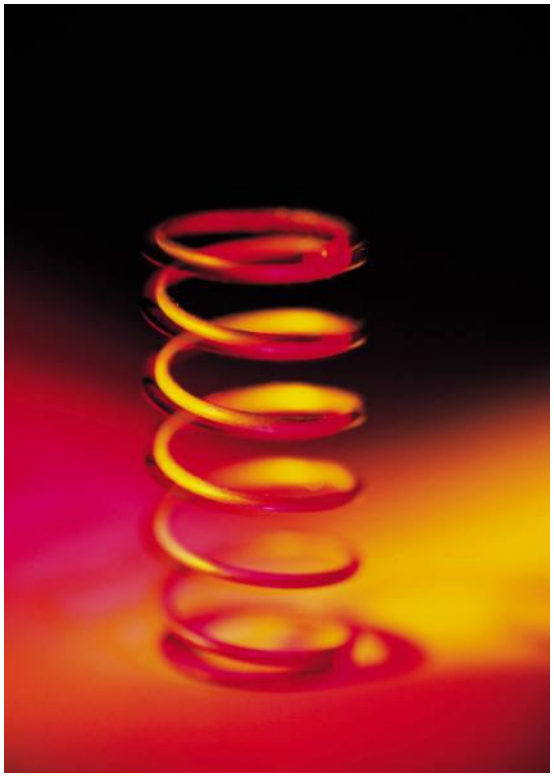
- 今回の課題では、地面でボールが跳ね返りました。これを同じ枠組みでシミュレーションしましょう。

- 我々のシミュレーションでは、
- とにかく「力」がわかれば良い
- のでした。先ほどはこの力は「重力」でした。

- では、跳ね返る瞬間の「力」とは何でしょうか。

反発力



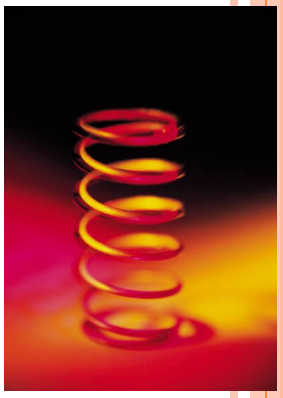


- ごく簡単な仮定をしましょう。

- 反発力は、変形量に比例する。

- つまり、ボールが「バネ」であると考えるわけです。

跳ね返り



- 変形量とは何でしょうか。ここでも簡単な仮定を置きましょう。
- 変形量とは、沈み込み量である

- すると上の図から、
- d=r-y
- が変形量になります。ただしもちろん、y<rの場合に限ります。

シミュレーション

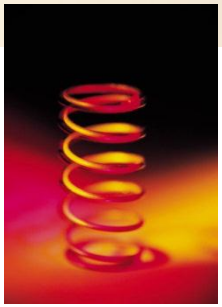
- ボールに加わる力は、「重力」と「反発力」です。
- シミュレーションは次のようになるでしょう。

```
v=0;      //速度の初期値
y=0;      //位置の初期値
While(1){
        F = -mg;            //下向きの重力
        if(y<r){
                F += k(r-y);        //反発力
        }
        a = F/m;            //力から加速度を得る
        v += a・Δt;         //加速度から速度を更新する
        y += v・Δt;         //速度から位置を更新する
}
```

これだけで、「自由落下＆跳ね返る」ボールがシミュレートできます。

シミュレーション

- kを色々変えてみましょう。
- kはすでに気づいているように、「バネ定数」です。
  - 大きなkは「硬い」振る舞いを生みます。
  - 小さなkは「柔らかい」振る舞いを生みます。



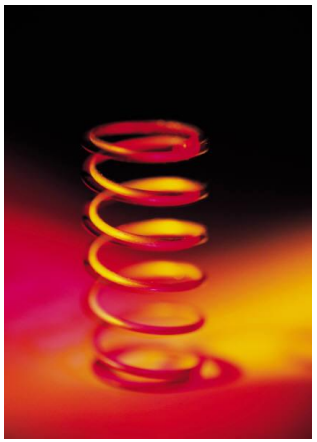
- ただ、このシミュレーションには一つ問題があります。

完全なバネ＝減衰しない

- このシミュレーションでは、バネは完璧に距離に比例する反発力を生みます。

- よく知られているように、そうした「完璧なバネ」は、無限に減衰しない振動を生みます。

- つまり、このボールは無限にバウンドしてしまいます。



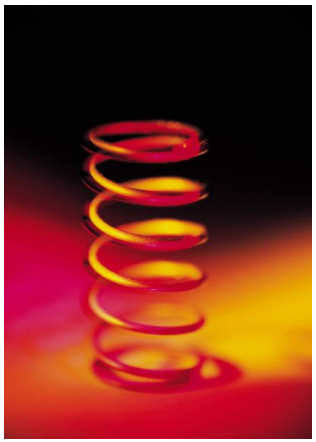
減衰させる

- これを改善するには、バネによって生じる力を弱めれば良いです。

- ただし、バネ定数kを減らしても、それはバネが「柔らかく」なっただけで、完璧なバネであることには変わりありません。

- そこで、「バネ」とは異なる物理的な性質を与える必要があります。

- それが「粘性項」です。

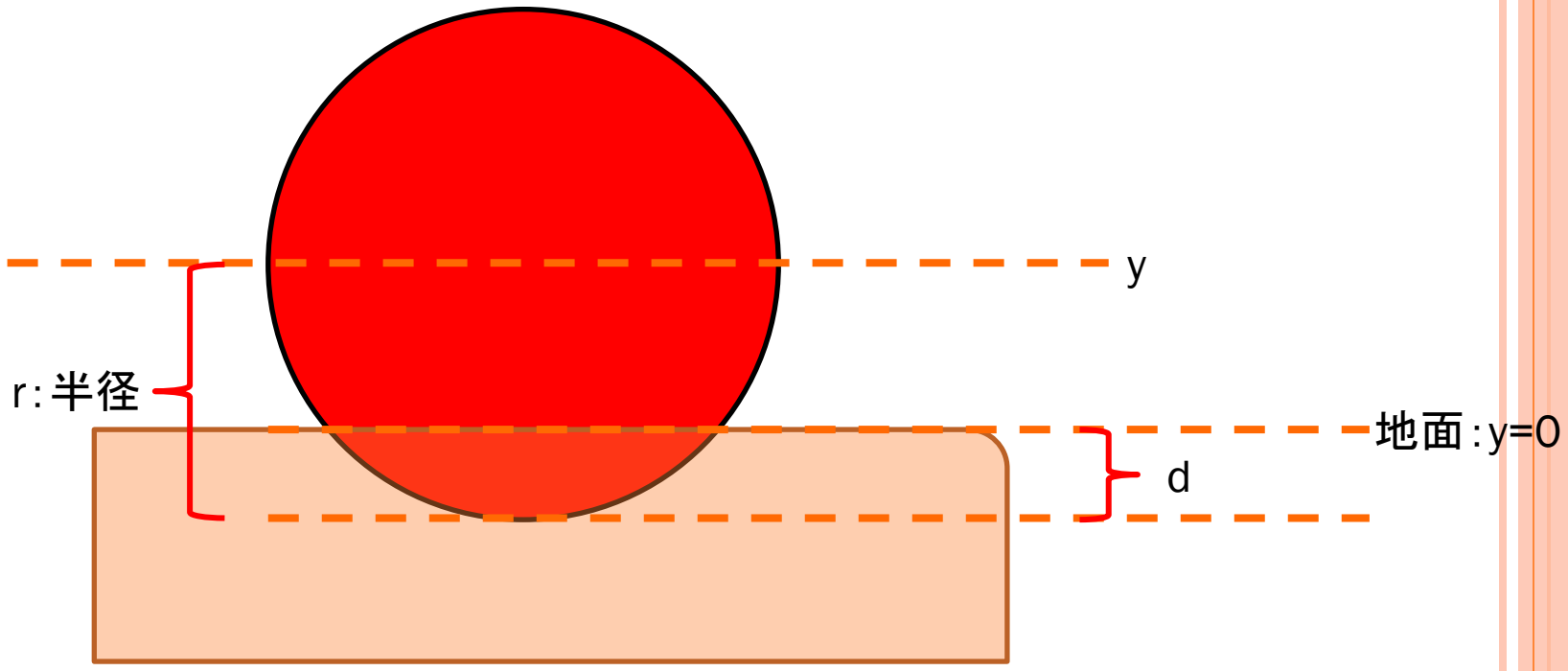


粘性（ダンパ）

- 粘性、というのは、要はブレーキです。エネルギーを吸収する働きを持ちます。ダンパ(damper)とも言います。実際、自動車に入っている衝撃吸収用のダンパと同じです。

- 最も代表的な粘性は「速度に比例する」ものです。

跳ね返り（粘性つき）



- 粘性を考慮すると、ボールに加わる力は、
  - バネ成分（d=r-y に比例）
  - ダンパ成分（v に比例）
- の二つになります。ただしもちろん、y<rの場合に限ります。

シミュレーション

- シミュレーションは次のように変更されるでしょう。

```
v=0;      //速度の初期値
y=0;      //位置の初期値
While(1){
        F = -mg;            //下向きの重力
        if(y<r){
                F += k(r-y) - cv;     //反発力
        }
        a = F/m;            //力から加速度を得る
        v += a・Δt;         //加速度から速度を更新する
        y += v・Δt;         //速度から位置を更新する
}
```

Cはダンパの係数です。
Kとcを調整すると、色々な跳ね返りがシミュレートできます。
実際のソース（課題4）を見て、納得してください。

シミュレーション（多次元）

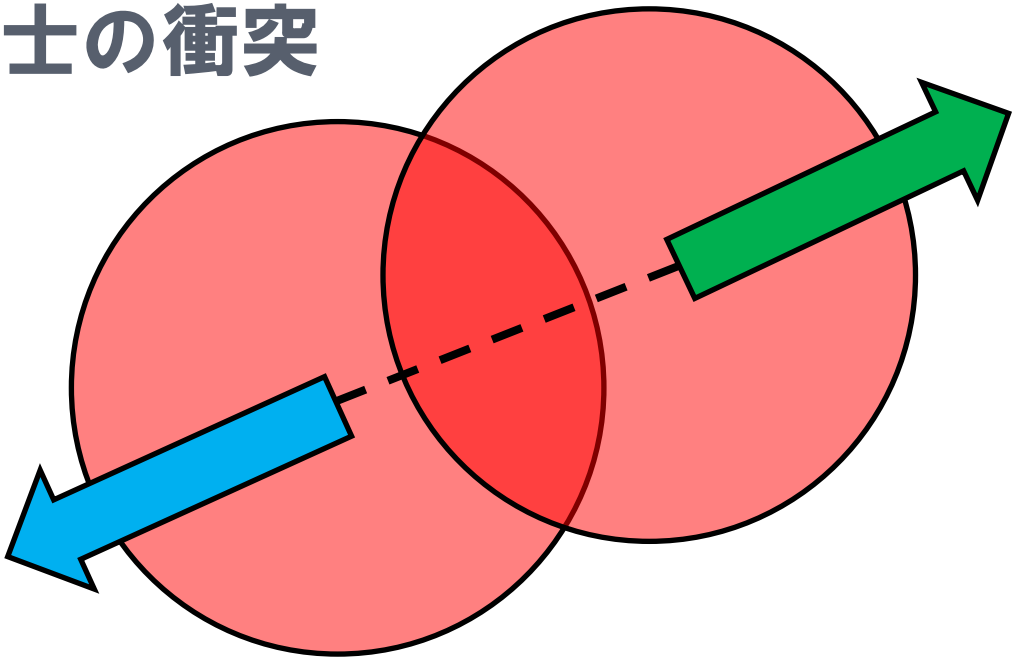
- これまでの話はすべて「y軸」だけで行って来ましたが、全て簡単に3次元に拡張できます。

- y軸に加わる力は、y軸の速度を生み、y軸の速度はy軸の位置の変化を生みます。x軸、z軸も同じです。

- ですから変数を3倍に増やすだけで3次元に対応できることになります。

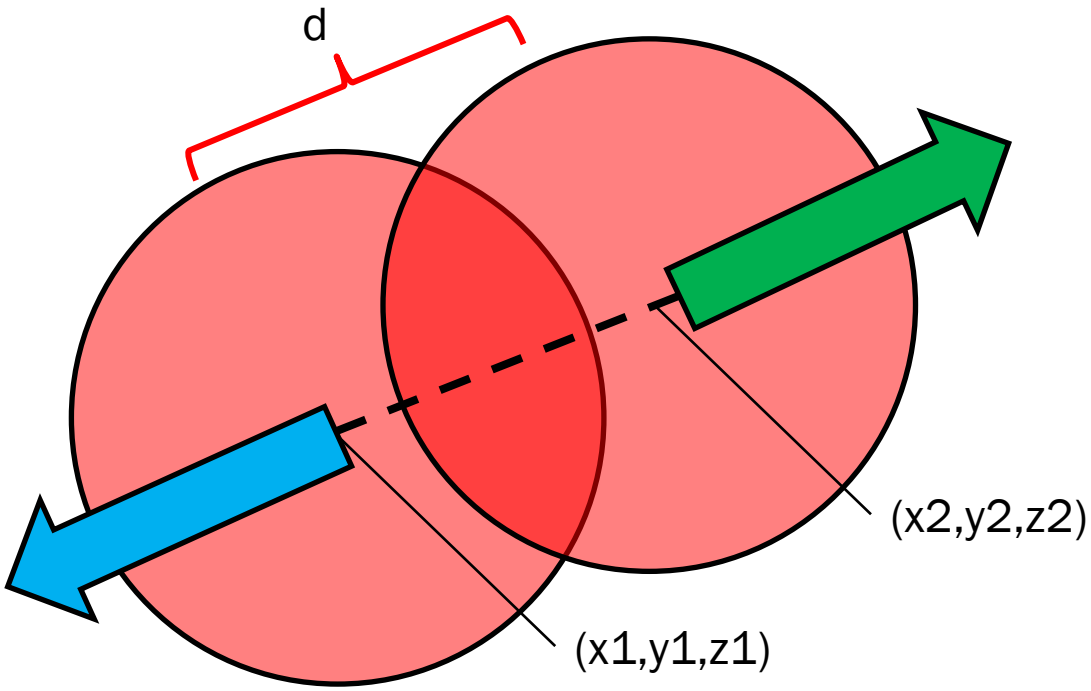
- 変数は例えば、
- ax,ay,az,　vx,vy,vz,　x,y,z
- になるでしょう。

ボール同士の衝突



- これまでと同じ考え方で、「ボール同士の衝突」もシミュレートできます。しかも3Dで。
- 図のように、ボールの「重なり」を考え、「重なり」に比例した力が加わると考えれば良いです。

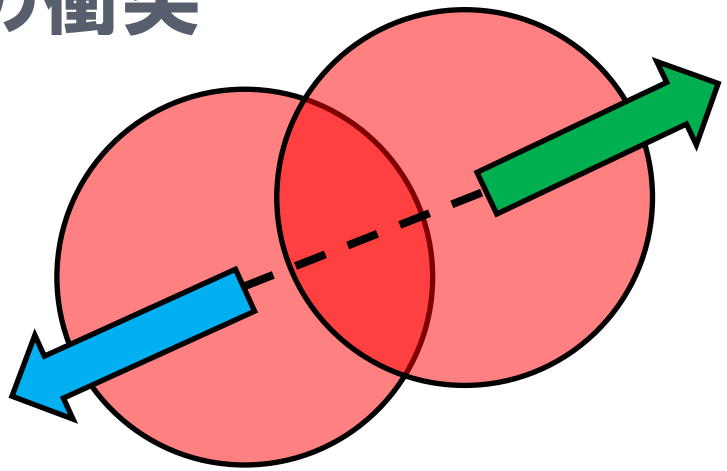
ボール同士の衝突



- ボールの中心同士の距離は分かりますよね。
- $d = \sqrt{(x1-x2)^2+ (y1-y2)^2+ (z1-z2)^2}$

- これが半径の二倍未満なら、その分の反発力が生じるとします。
- $F = k(2r-d)$
- （ここでは先ほどの粘性項は省略しています）

ボール同士の衝突



- ただし、生じる力の方向は、中心同士を結ぶ線上になります。
- この方向の単位ベクトルは、
- $e_{12} = 1/d \times (x1-x2, y1-y2, z1-z2)$
- このベクトルが、先ほどの力に掛け合わされて、「向きを持った力」が出ます。
- $F(\text{ベクトル}) = F \cdot e_{12}$
- この場合ボール2に加わる力になり、ボール1には逆向きの力が加わります。

シミュレーション

- ここまでを理解できれば、
  - 地面に衝突したら跳ね返り
  - ボール同士も衝突する
- シミュレーションは容易に出来ます。

- 骨組みだけ示します。

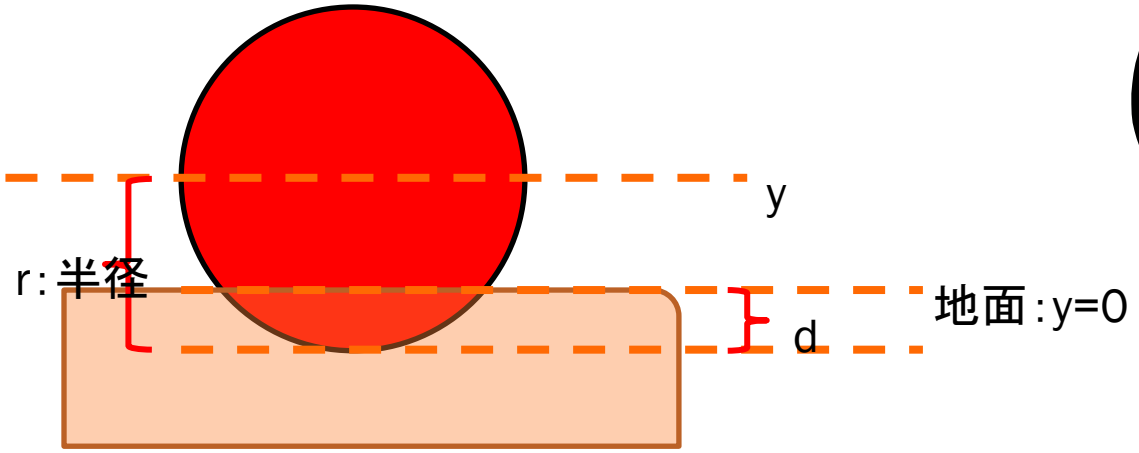
シミュレーション

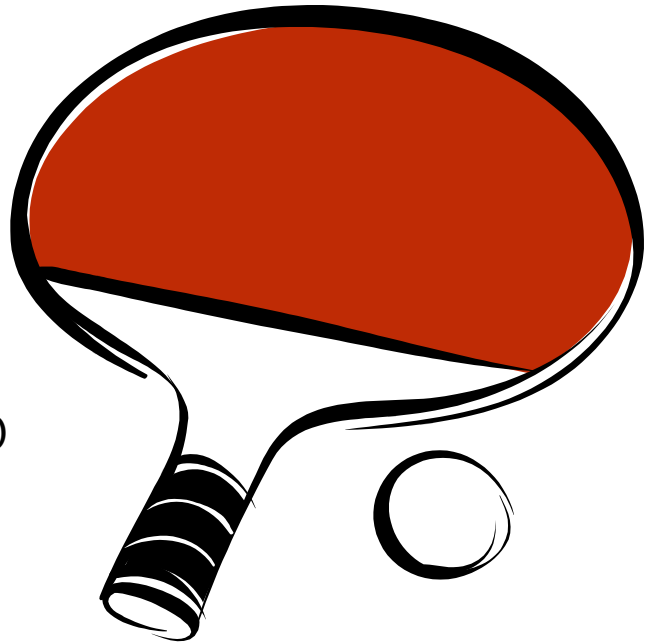
```
While(1){
        ○○//下向きの重力を計算
        ○○// 床からの反発力を加算
        ○○//ボール同士の距離に応じた反発力を加算
        ax = Fx/m;      //力から加速度を得る
        vx += ax･Δt;    //加速度から速度を更新する
        x += vx･Δt;     //速度から位置を更新する
        ay = Fy/m;      //力から加速度を得る
        vy += ay･Δt;    //加速度から速度を更新する
        y += vy･Δt;     //速度から位置を更新する
        az = Fz/m;      //力から加速度を得る
        vz += az･Δt;    //加速度から速度を更新する
        z += vz･Δt;     //速度から位置を更新する

}
```

実際のソース（課題5）を見て、納得してください。

発展：動く地面

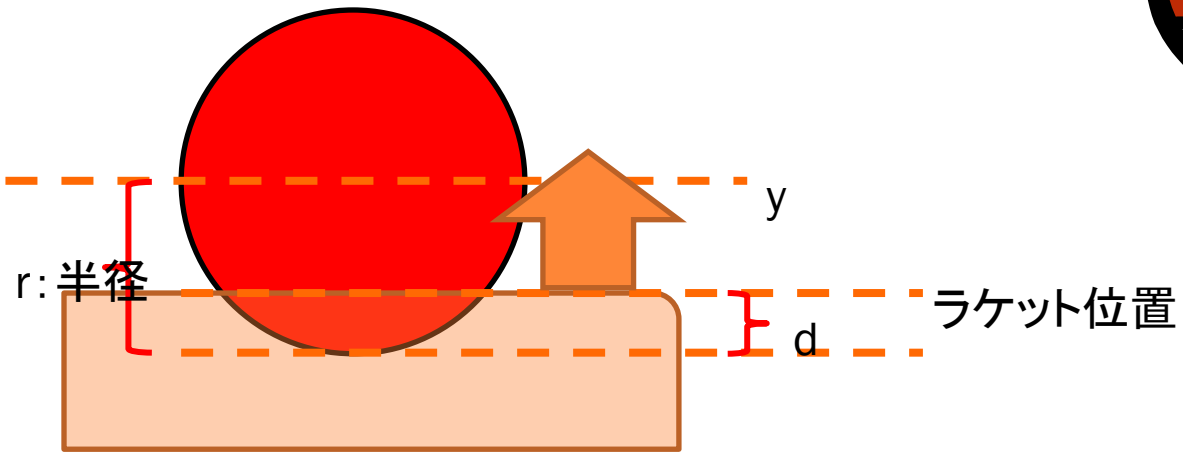


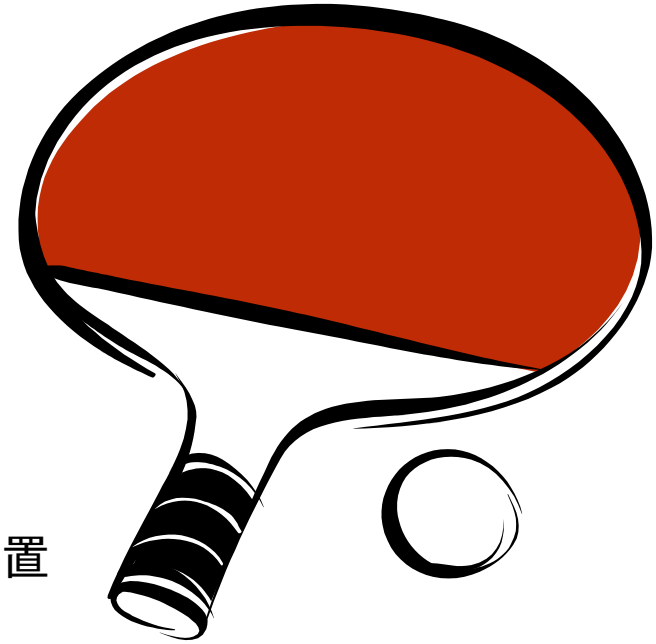


- 今後の課題では、「ラケットでボールを打ち返す」ことを実現しないといけません。これはどうしたら良いでしょう？

- ラケットの速度に応じた処理が必要でしょうか？
- なんだかそんな気もしますね。

発展：動く地面





- 種を明かすと、ラケットの速度に応じた処理は不要です。
- これまでの「地面での跳ね返り」と全く同様に、沈み込み量に比例した力を発生させればよいのです。
- ラケットの速度のせいで、沈み込み量自体が変化するので、速度による効果はちゃんと現れるのです。

地面を振動させてみる

- 確かめるために、地面をy方向に振動させてみましょう（自分でもやってください）。

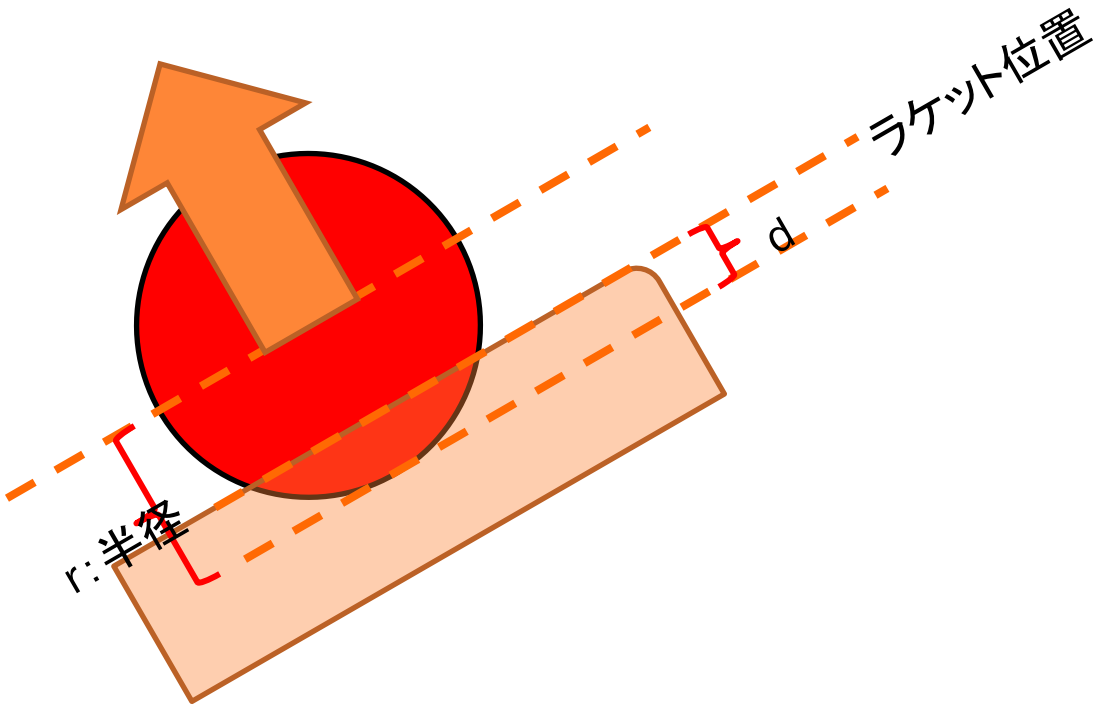
- 先ほどは地面がy=0の位置にあったので、沈み込み量は
- r(半径)-y(ボールの高さ)
- でした

- 今回は、地面の位置がA・sin(t)となるので、次のように変わるでしょう。

```
v=0;        //速度の初期値
y=0;        //位置の初期値
While(1){
        F = -mg;                    //下向きの重力
        if(y<r){
                F += k(r-y+A・sin(t)) - cv;          //反発力
        }
        a = F/m;  //力から加速度を得る
        v += a・Δt;  //加速度から速度を更新する
        y += v・Δt;  //速度から位置を更新する
}
```

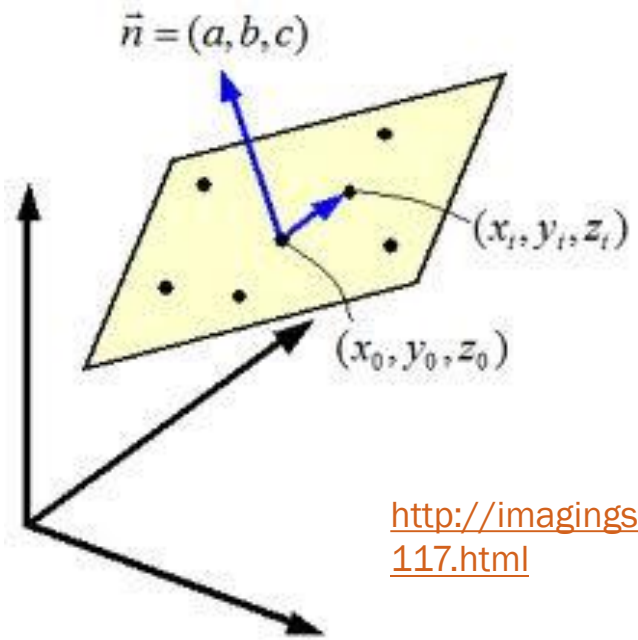
- 地面の振幅や、振動周波数を色々変えてみましょう。
- 地面自体も描画しましょう。

傾いた地面



- 今回の課題では、地面（ラケット）は傾きます。
- 反発力は、地面とは垂直の方向、つまり「法線ベクトル」の方向に生じると考えれば良いです。
- ここでもやはり、傾く速度や、ボールの来る方向は考えなくて良いです。
- （ボールの回転等、より正確なシミュレーションでは必要になってきますが、大幅に簡略化しています）

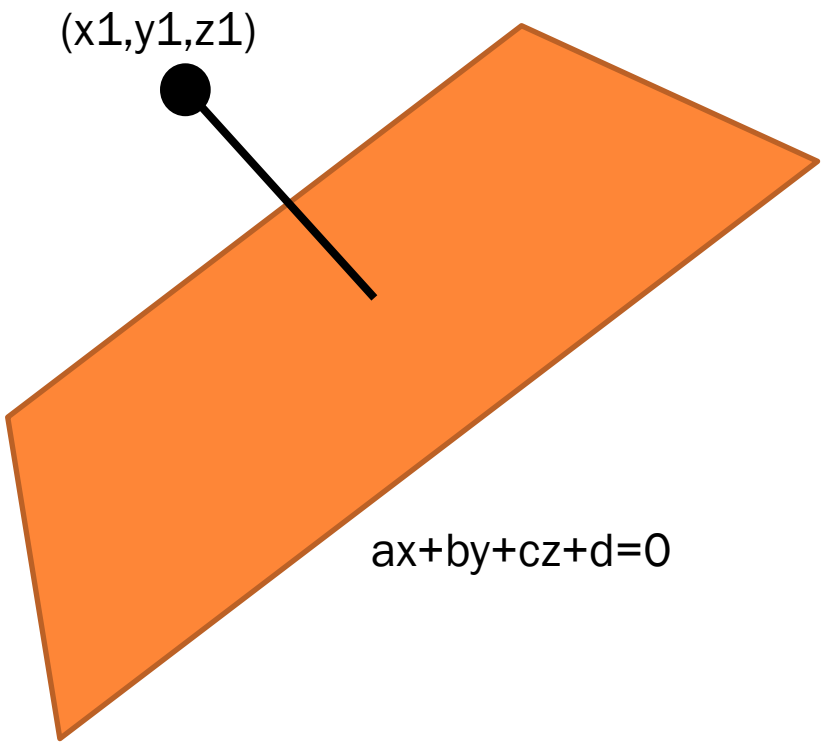
傾いた地面からの反発力



http://imagingsolution.blog107.fc2.com/blog-entry-117.html

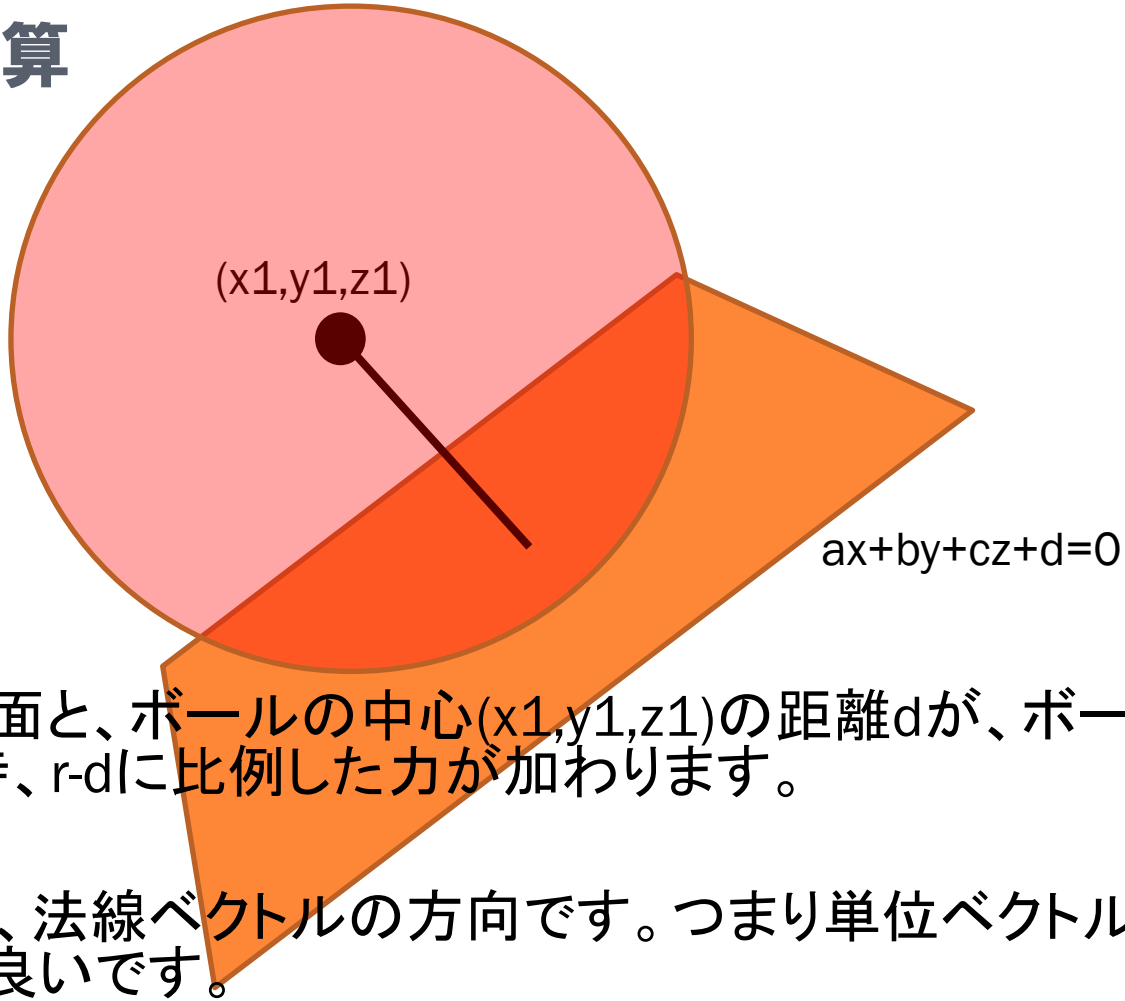
- ボールの地面（今後はラケット面と呼びましょう）の法線ベクトルをn=(a,b,c)とします。これは単位ベクトルであるとします。
- ラケット面の方程式は、
- ax+by+cz+d=0
- でしたね。ここのdは、この平面がどこを通過するかによって決まります。

傾いた地面からの反発力



- ラケット面と、ボールの中心(x1,y1,z1)の距離はどう計算できるでしょうか。これは「平面と点との間の距離」ですよね。
- 高校の教科書に公式が書いてあります。
- Webでも適当に見つかります。

反発力の計算



- このラケット面と、ボールの中心(x1,y1,z1)の距離dが、ボールの半径r未満の時、r-dに比例した力が加わります。
- 力の方向は、法線ベクトルの方向です。つまり単位ベクトル(a,b,c)をかければ良いです。
- あとは、力⇒加速度⇒速度⇒位置、といういつものシミュレーションです。
- 粘性項も入れるとよいでしょう。

残りの要素

- ここまででおそらく一仕事でしょう。
- ラケットとしての面白さを出すには、これ以外に、ラケットの大きさを考える必要があります。今は「無限平面」です。
- たとえばラケットを「円盤」として考えると、ボールがこの範囲内に入っているかどうかはどう判定したら良いでしょう？考えてみてください。
- ボールのスピンなども考えられれば面白いですが、今回はやめておきましょう。

少しずつ

- すべて少しずつやってください。
- たとえば、
  - 地面を垂直に振動させて、それっぽい振る舞いをさせる、
  - 地面を傾けた時に、それっぽい振る舞いをさせる、
- それぞれに2～3日ずつかかってしまうかもしれません。でも一気にやったら多分できませんが、分割すればそれぞれは解ける問題です。

- どんな大きなプロジェクトも、実現すべきことがどのような要素でできているかを見極め、工程表を作れば、できます。逆に、反射神経でできる事は限られています。学生実験などはこうした考えに基づいてデザインされています。

- 今回の課題は、こうした「工程表づくり」を自分でできるようになるための課題でもあります。