mlibを用いた物理シミュレーション制作（1）

物理運動のシミュレーション

微分方程式→差分方程式→漸化式

等速度運動

運動方程式	微分方程式	差分方程式	漸化式　$x(n\Delta t) = x_n$
$m\frac{d^2}{dt^2}x(t) = 0$	$\frac{d}{dt}x(t) = v$	$\frac{x((n+1)\Delta t) - x(n\Delta t)}{\Delta t} = v$	$x_{n+1} = x_n + v\Delta t$

等加速速度運動（慣性の動き）

$m\frac{d^2}{dt^2}x(t) = ma$	$\frac{d}{dt}v(t) = a$	$\frac{v((n+1)\Delta t) - v(n\Delta t)}{\Delta t} = a$	$v_{n+1} = v_n + a\Delta t$ $x_{n+1} = x_n + v_n\Delta t$

自由落下運動（鉛直上向きを正）

$m\frac{d^2}{dt^2}z(t) = -mg$	$\frac{d}{dt}v^z(t) = -g$	$\frac{v^z((n+1)\Delta t) - v^z(n\Delta t)}{\Delta t}$ $= -g$	${v^z}_{n+1} = {v^z}_n - g\Delta t$ $z_{n+1} = z_n + {v^z}_n\Delta t$

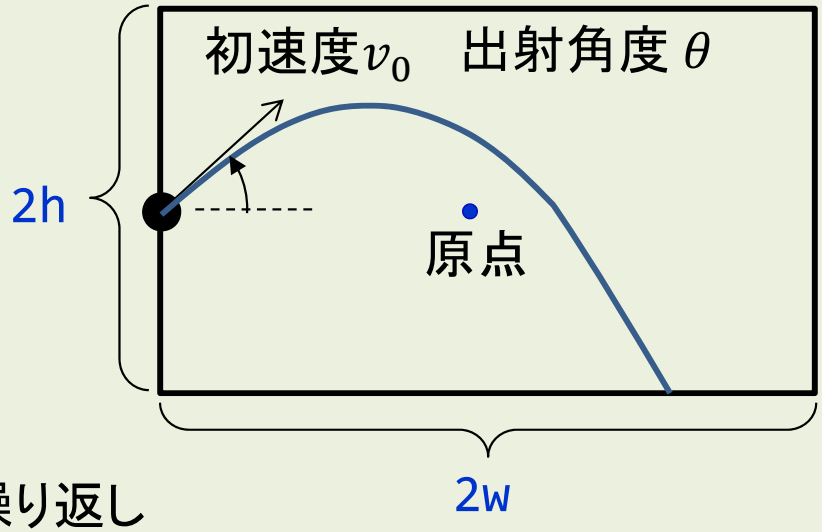
例題６－１

ボールを出射角度 θ 、初速度v_0で投げ上げたボールの放物運動のアニメーションを作成せよ。尚、θ、v_0はそれぞれ、エディットボタン0,1から入力する。また、重力加速度は$9.8\mathrm{m}/s^2$とする。

1. ボールの運動方程式を立てる。
   $ma_x = 0$　　$ma_z = -mg$
2. ボールの新しい位置を、現在位置からの増分で表す

$$x_{n+1} = x_n + v^x \Delta t$$
$$z_{n+1} = z_n + v^z{}_n \Delta t \qquad v^z{}_{n+1} = v^z{}_n - g\Delta t$$

3. 実座標の計算領域を設定，幅2w，高さ2h



4. エディットボックスから初速度、出射角度入力し、
   $v^x = v_0 \cos\theta$　，$v^z = v_0 \sin\theta$

5. while文でボールのz座標が画面範囲を超えるまで繰り返し
   過去のボール消去
   ボールの座標更新
   ボールのz方向速度更新
   実座標をピクセル座標に変換
   ボールを描画（Circle 関数）
   過去の座標に現在の座標を代入

例題６－１のプログラム例

```
#include "def.h"   //ここは必ず必要
#include "mlib.h" //ここは必ず必要

#define PI 3.1415926535
#define g  9.8

void main(int Number){

 int m,n,mm=0,nn=0,r=10;
 double x,z,vx,vz,w=100,h=100;
 double v0,theta,dt;

 v0=Get_double(0);    //速度入力
 theta=Get_double(1)/180*PI;
          //入力角度をラジアンに変換

 vx=v0*cos(theta); // x方向初速度
 vz=v0*sin(theta); // z方向初速度

 dt=5e-3;    //時間離散間隔設定

 x=-w; // ボールの初期x座標
 z=0;  // ボールの初期z座標
```

```
while(z>-h) { //z座標が画面の下部を過ぎるまで繰り返し

  Plot_pen(0,2,7); //白色に設定（バックと同じ）
  Circle(mm-r,nn-r,mm+r,nn+r,1); //過去の玉を消去

  x=x+vx*dt;  //  x座標更新

  vz=vz-g*dt; //  z方向速度更新
  z=z+vz*dt;  //  z座標更新

  m=sGW.w*(1+x/w) /2;  //実座標をピクセル座標に変換
  n=sGW.h*(1-z/h) /2;    //実座標をピクセル座標に変換
                         //(z座標は上向きを正))

  Plot_pen(0,2,3); //緑色に指定
  Circle(m-r,n-r,m+r,n+r,1);   //新しい位置に玉を描画

  UpdateWindow(hWnd);  //画面更新

  mm=m; //新しいピクセル座標を過去のピクセル座標に
  nn=n; //新しいピクセル座標を過去のピクセル座標に
 }
}
```

壁による跳ね返り

撃力　－　瞬間的に大きな力が加わることにより、運動する物体の力の作用線方向の速度成分を変化させる

壁面に対する力の作用線方向　→　壁面に垂直な方向

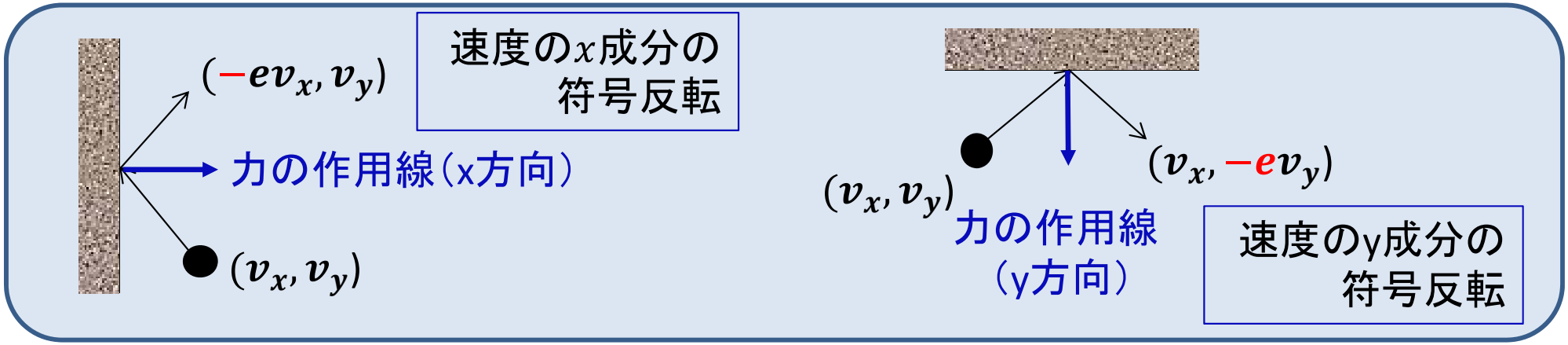
壁面に平行な速度成分

速度成分は変化しない　　$v_{\parallel} \rightarrow v_{\parallel}$

壁面と直交する速度成分

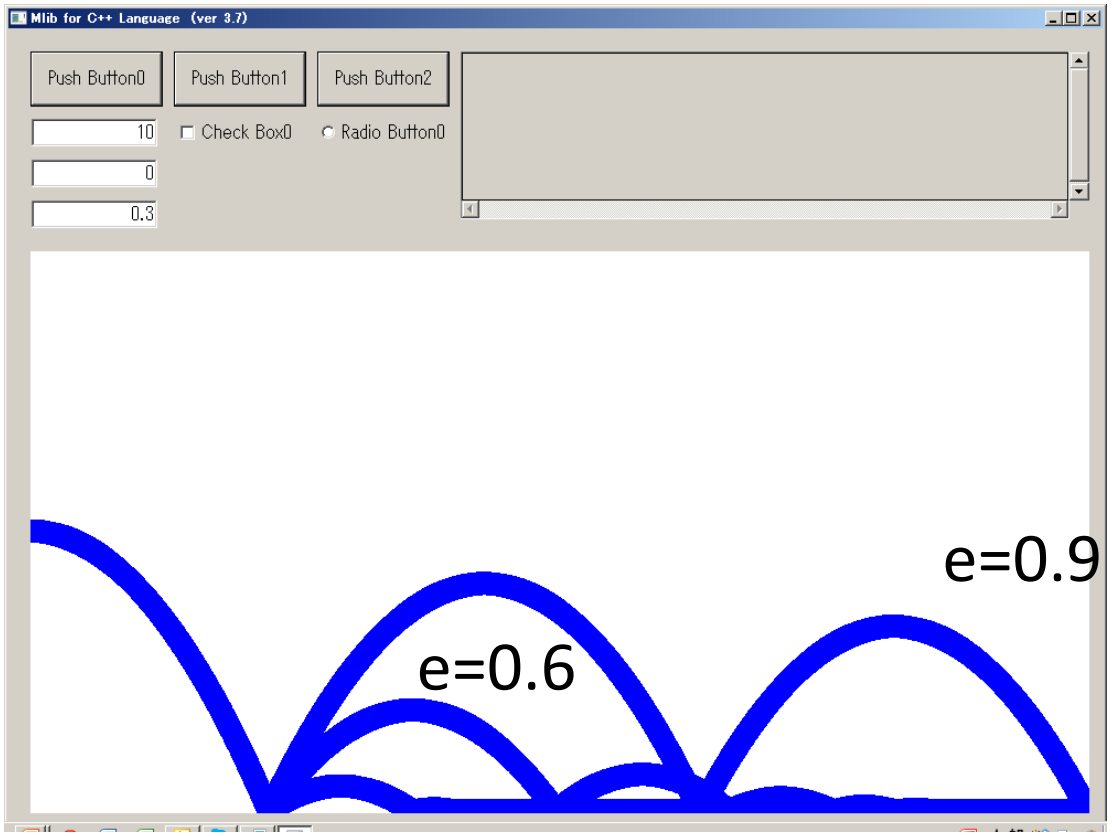
速度成分の符号は反転　　$v_{\perp} \rightarrow -ev_{\perp}$

速度成分の大きさは跳ね返り係数eに応じて小さくなる



例題６－２

ボールを出射角度 θ 、初速度v_0で投げ上げたボールの放物運動のアニメーションを作成せよ。ただし、床面での反射係数をeとする。尚、θ、v_0、eはそれぞれ、エディットボタン0,1,2から入力する。また、重力加速度は$9.8\mathrm{m}/s^2$とする。



例題６－２

```
#include "def.h
#include "mlib.h"
#define PI 3.1415926535
#define g  9.8

void main(int Number){

 int m,n,mm=0,nn=0,r=10;
 double x,z,vx,vz,w=100,h=100;
 double v0,theta,dt;
 double e;



  v0=Get_double(0); //速度入力
  theta=Get_double(1)/180*PI;
  e=Get_double(2); //跳ね返り係数

  vx=v0*cos(theta);
  vz=v0*sin(theta);
  dt=5e-3;

  x=-w; z=0;
```

```
 while(x<w) {
  Plot_pen(0,2,7);
  Circle(mm-r,nn-r,mm+r,nn+r,1); //過去の球を消去
   x=x+vx*dt;   //   x座標更新
   vz=vz-g*dt;   //  z方向速度更新

  if (z+vz*dt<-h) {vz=-e*vz;}  // 物体が床に当たったとき、
                               // 速度に反射係数を乗算
    z=z+vz*dt;

   m=sGW.w/2*(1+x/w);  //座標変換
   n=sGW.h/2*(1-z/h);

   Plot_pen(0,2,3);              //緑色に指定
   Circle(m-r,n-r,m+r,n+r,1);   //新しい位置に図を描画

   UpdateWindow(hWnd);  //画面更新

   mm=m;
   nn=n;
  }
}
```

物体どうしのCollision 判定

質点

大きさを持たない点　　位置(x_1, y_1)

(x_1, y_1) ● ●(x_2, y_2)

点どうしの当たり判定は、座標が完全に一致

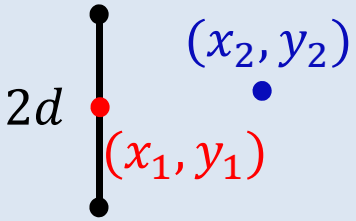
$x_1 = x_2$,　$y_1 = y_2$

線分

太さを持たない
有限長の線

中心位置 (x_1, y_1)　　長さ $2d$

線分と平行な方向の座標が一致 $(x_1 = x_2)$　かつ
線分の中心と質点との距離が線分半分の長さ以下

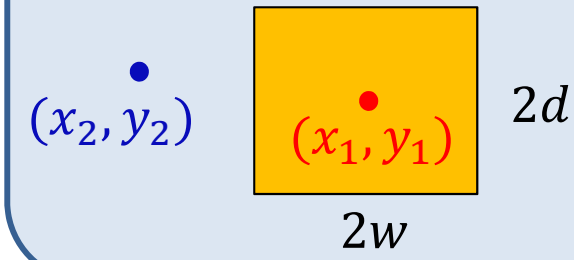


$y_1 - d < y_2 < y_1 + d$ ⇒ $|y_1 - y_2| < d$

四角

中心位置 (x_1, y_1)　　高さ $2d$　　幅 $2w$

質点2が、四角の中に入る条件



$x_1 - w < x_2 < x_1 + w$
かつ
$y_1 - d < y_2 < y_1 + d$

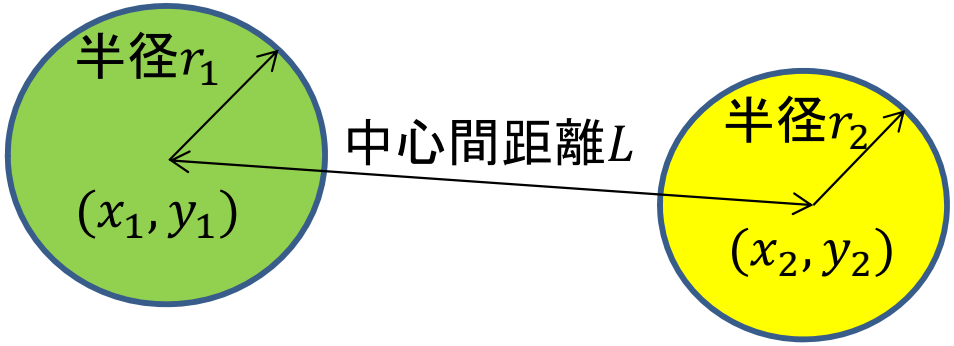
⇒

$|x_1 - x_2| < w$
かつ
$|y_1 - y_2| < d$

大きさを持つ物体どうしのCollision 判定

物体が円の場合

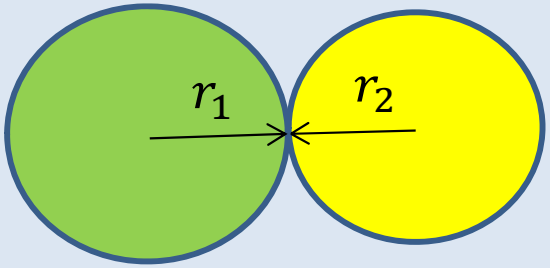
パラメータは中心間距離と半径



中心間距離は

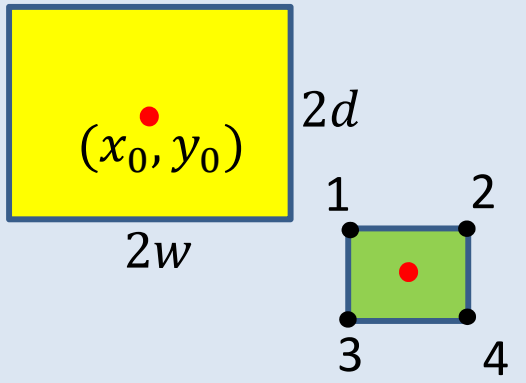
$$L = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

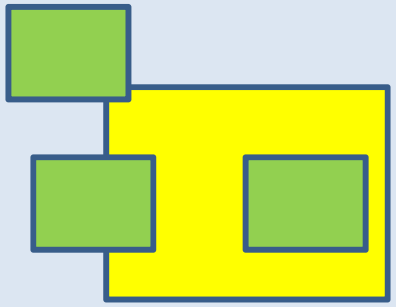
衝突の条件は　$L < r_1 + r_2$



物体が長方形の場合

小さい方の長方形について、その角の4点を質点として、
そのうちいずれかが四角の中にあれば衝突とみなす。





($|x_0 - x_1| < w$ and $|y_0 - y_1| < d$) or

($|x_0 - x_2| < w$ and $|y_0 - y_2| < d$) or

($|x_0 - x_3| < w$ and $|y_0 - y_3| < d$) or

($|x_0 - x_4| < w$ and $|y_0 - y_4| < d$)

Collision 判定(1)　座標による判定

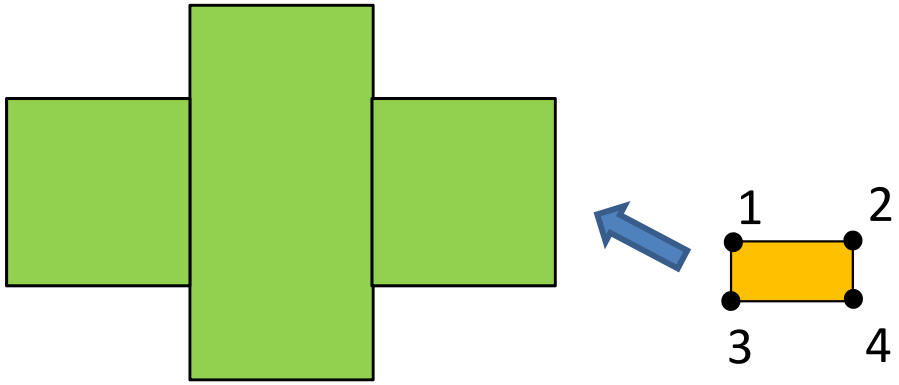
物体2 (x_2, y_2) / 物体1 (x_1, y_1)

物体1＼物体2	点	線分 $2d_2$	半径r_2	四角 $2w_2$ $2d_2$
点	$((x_1==x_2)$ && $(y_1==y_2))$	$(\ (\mathrm{abs}(y_1 - y_2) < d_2)$ && $(x_1==x_2)\ )$	$\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} < r_2$	$(\ (\mathrm{abs}(x_1 - x_2) < w_2)$ && $(\mathrm{abs}(y_1 - y_2) < d_2)\ )$
線分 $2d_1$	$2d_1$が物体2のサイズより小さいとき→線分1を質点2点の集合体とみなす $((\ (\mathrm{abs}(y_1 - d_1 - y_2) < d_2)$ or $(\mathrm{abs}(y_1 + d_1 - y_2) < d_2))$ and $(x_1==x_2)\ )$	$\sqrt{(x_1-x_2)^2 + (y_1-d_1-y_2)^2} < r_2$ and $\sqrt{(x_1-x_2)^2 + (y_1+d_1-y_2)^2} < r_2$	$(\ (\mathrm{abs}(x_1 - x_2) < w_2)$ and $((\mathrm{abs}(y_1 - d_1 - y_2) < d_2)$ or $(\mathrm{abs}(y_1 + d_1 - y_2) < d_2))\ )$	
半径r_1		$\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} < r_1 + r_2$	$d_2, w_2 < r_1$のとき 四角を角の4つの点の集合体とみなし、各点毎に円と点の判定を行い、どれか一つでも含まれて入ればよい	
$2w_1$ $2d_1$				$d_2, w_2 < d_1, w_1$のとき 物体2の四角を角の4つの点の集合体とみなし、各点毎に物体1の四つの角と点の判定を行い、どれか一つでも含まれて入ればよい

・複雑な形どうしの判定は多くのif文が必要、一旦書いてしまえばあとは楽

Collision 判定の簡略化

ある大きさを持つ物体どうしのCollision判定は複雑になっていく

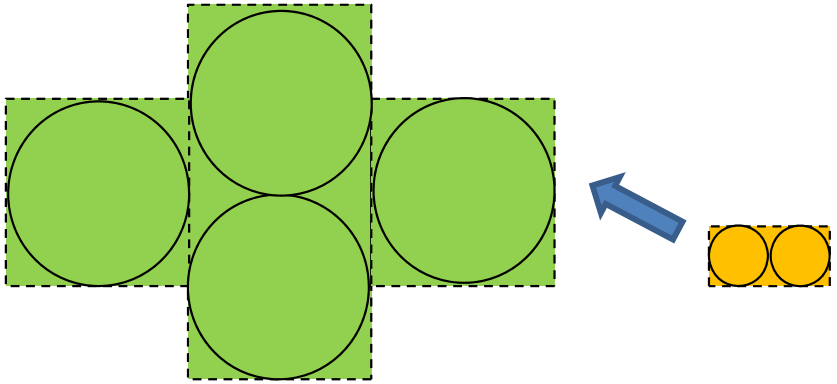


質点1が緑の範囲に入る条件　or
質点 2が緑の範囲に入る条件　or
質点 3が緑の範囲に入る条件　or
質点 4が緑の範囲に入る条件

点と長方形では2個の条件が必要なので、　２＊３＊４＝24個の条件

一般に、円どうしの判定はシンプル

物体1, 2 を円の集合体とみなして、円で判定

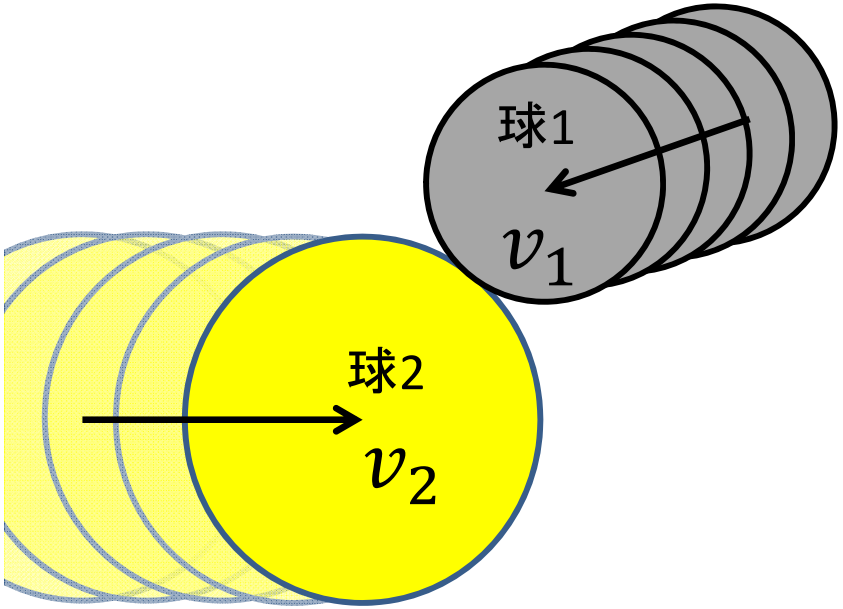


円どうしは1個の条件なので、

４＊２＝８個の条件

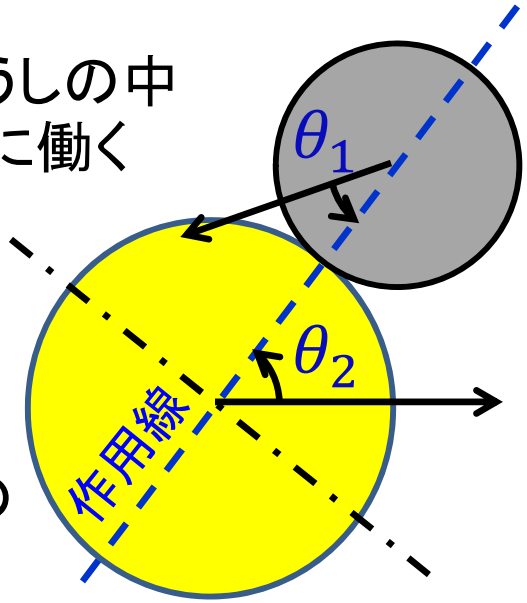
実際物体の角付近の当たり判定は厳しくなくてもよいので、
円で当たり判定するのが基本

球体同士の衝突後の運動

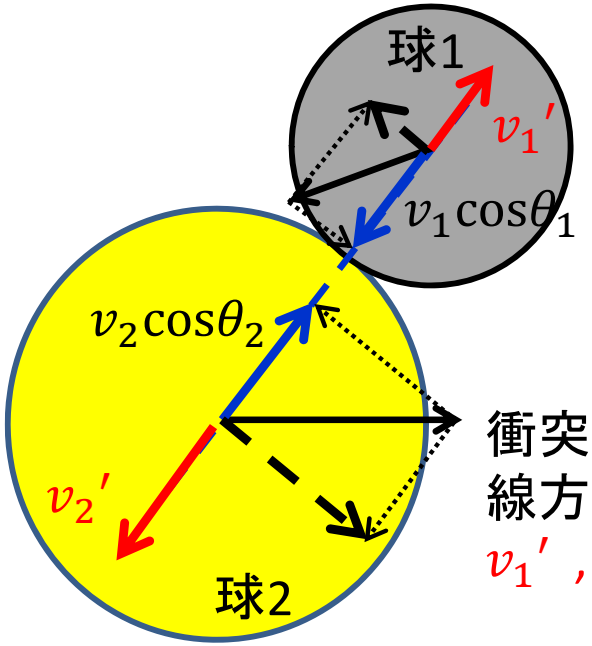


球に作用する力は球どうしの中心を通る線（作用線）上に働く

作用線と直交する速度成分はそのまま



作用線と運動方向のなす角θ_1、θ_2が重要



衝突後の作用線方向の速度をv_1' , v_2' とする

作用線上で運動量保存則

$$m_1v_1\cos\theta_1 + m_2v_2\cos\theta_2 = m_1v_1' + m_2v_2'$$

跳ね返り係数の定義

$$e = -\frac{v_1' - v_2'}{v_1\cos\theta_1 - v_2\cos\theta_2}$$

$$v_1' = v_1\cos\theta_1 + \frac{m_2(1+e)(v_2\cos\theta_2 - v_1\cos\theta_1)}{m_1+m_2}$$

$$v_2' = v_2\cos\theta_2 - \frac{m_1(1+e)(v_2\cos\theta_2 - v_1\cos\theta_1)}{m_1+m_2}$$

作用線と直交する速度は変化なし、衝突後の球の運動はv_1' , v_2' との合成ベクトルで決まる

演習６－３

X軸のみを移動できる二つの球について、一方の球の速度を左右キーで加減できるものとする。球どうしは衝突により跳ね返り係数を1として跳ね返る。また、球と両側の壁面でも跳ね返り係数を1として跳ね返るものとする。この運動をシミュレートせよ。

過去の座標を実座標で表現しているので注意

```
#include "def.h"
#include "mlib.h
#define W 100
#define H 100

int m2x(double x){//座標変換
  return sGW.w/2*(x/W+1);}
int n2y(double y){//座標変換
  return sGW.h/2*(1-y/H);}

void main(int Number){
 short pl,pr;
  double x=80,y=0,xx=x;
  double vx=0,r=10,m=10; //球Oの情報
  double x1=-80,xx1=0,vx1=0.1;
  double r1=10,m1=10; ; //球1の情報
  double d,pp,e;

  e=1; //跳ね返り係数
```

```
 while(1) {
   Plot_pen(0,2,7); //白色に設定
   Circle(m2x(xx-r),n2y(y-r),
m2x(xx+r),n2y(y+r),1);
   Circle(m2x(xx1-r1),n2y(y-r1),
m2x(xx1+r1),n2y(y+r1),1);

   pl=GetAsyncKeyState( VK_LEFT  );
   pr=GetAsyncKeyState( VK_RIGHT );
   if (pl<0) 球Oの左方向速度増加
   if (pr<0) 球Oの右方向速度増加
   d=   球の中心間の距離
   if(d<r+r1) {
      pp=(1+e)*(vx1-vx); //衝突の係数

         vx =  球Oの衝突後速度更新
         vx1=  球1の衝突後速度更新
}
//次に続く
```

演習6－3（つづき）

```
    x=x+vx;    //球0の衝突後x座標
   x1=x1+vx1;    //球1の衝突後x座標

   if
   if   球0，1と両側の壁の衝突
   if   衝突により速度を反転
   if

        Plot_pen(0,2,3); //緑色に指定
        Circle(m2x(x-r),n2y(y-r),m2x(x+r),n2y(y+r),1);
        Plot_pen(0,2,2); //赤色に指定
        Circle(m2x(x1-r1),n2y(y-r1),m2x(x1+r1),n2y(y+r1),1);

   xx=x;    //新しい実座標を過去の実座標にする
   xx1=x1; //新しい実座標を過去の実座標にする

   vx=vx*0.9999;   //ころがり抵抗
   vx1=vx1*0.9999;//ころがり抵抗

   UpdateWindow(hWnd);   //画面更新
 }
}
```

中間課題

以下の2項目を必ず含み

- 等速もしくは等加速度運動
- 物体と物体間、物体と壁等の衝突の問題

かつ、以下の2項目のどちらかを含んだプログラムを完成させる。

1. 物理現象をシミュレートする
2. ゲーム性を持たせる

注意点

- 過去の課題や他のプログラムをコピーしたと認められる場合は0点
- 評価対象はオリジナリティ、プログラムの完成度

締切　12月27日まで。これ以降遅れた場合評価しない。
提出物　Cソースファイル、def.h、課題内容の説明ファイル
提出先　miwa@gunma-u.ac.jp　メールタイトル：中間課題

来週、どんな内容にするか全員に聞くので、A41枚にまとめておくこと