Robot Programming

Zhidong Wang

Nov. 17, 2021

Project 2
Build a Robo-Kick Simulator

Skills on implementation algorithms simulating with dynamics of a robot and a ball

- 2D Runtime Graphics
- Simulation of a Ball with Dynamics
- Implement Data in Structure Style
- Simulation of a Robot Leg with Dynamics and Control
- Simulation of Interaction among robot leg, ball and environment

Step by Step

toward Robo-Kick Simulator

- Build the 2D graphics environment
- Show a Ball moving on the ground
- Simulate the Ball moving with friction force and gravity
- Simulate the Ball free-flying and bouncing
- Simulate the Ball kicked by a Foot
- Simulate the motion of the Leg and the Ball when the Leg kicks the Ball

http://www.robotics.it-chiba.ac.jp/wang/lect/

Basics of Runtime Simulation

Essential Technologies

Data + Modeling + Graphics(CG)

Step by Step : Graphics

toward Robo-Kick Simulator

- Build the 2D graphics environment by using OpenGL and GLUT Library

http://www.opengl.org

Instruction and Source Files Download

http://www.robotics.it-chiba.ac.jp/wang/lect/

Computer Graphics (1)

sample1.c をダウンロード、内容を理解する

- 四角形状を描画する部分を改造して、多角形を作成・描画してみる
- 等辺36角形を作成し、円を近似的に描画する

  （ for 文を利用する）
- チャレンジトピック：円の描画部分を改造し、パックマン（Pac-Man)を作成してみる

2D Graphics

Sample1.c

```
glBegin(GL_LINE_LOOP);
glVertex2d(-0.9, -0.9);
glVertex2d(0.9, -0.9);
glVertex2d(0.9, 0.9);
glVertex2d(-0.9, 0.9);
glEnd();
```

Sample3.c

```
glRectf( -15.0, -15.0, 15.0, 15.0 );
```

2D Graphics (Draw a ball)

```
glBegin(GL_LINE_LOOP);
glVertex2d(-0.9, -0.9);
glVertex2d(0.9, -0.9);
glVertex2d(0.9, 0.9);
glVertex2d(-0.9, 0.9);
glEnd();
```

```
glBegin(GL_LINE_LOOP);
for( i=0, i<36, i++)
    glVertex2d( r[ i ].x, r[ i ].y );
glEnd();
```

2D Graphics (Draw a ball)

Code list 1

```
glBegin(GL_LINE_LOOP);
for( i=0, i<36, i++)
    glVertex2d( r[ i ].x, r[ i ].y );
glEnd();
```

Code list 2

```
glBegin(GL_LINE_LOOP);
for( i=0, i<=36, i++)
    glVertex2d( r[ i ].x, r[ i ].y );
glVertex2d( 0.0, 0.0 );
glEnd();
```

What is the difference between these two code lists?

Computer Graphics (2)

sample3.cをダウンロード実行し、内容とある程度理解し、改造する

（マウスの左ボタンと右ボタンをクリックしてみる）

- 四角形の描画の部分をsample1.cの描画部分に置き換えて、多角形か円形に描画できるようにする

- 正方形か円形を横に移動できるようにする

Dynamics
and its Implementation

Essential Technologies

Data + Modeling + Graphics(CG)

Calculation of Dynamics

$$\ddot{x} = f_x / m, \qquad \ddot{y} = f_y / m$$

$$\ddot{\theta} = \tau_z / I_z$$

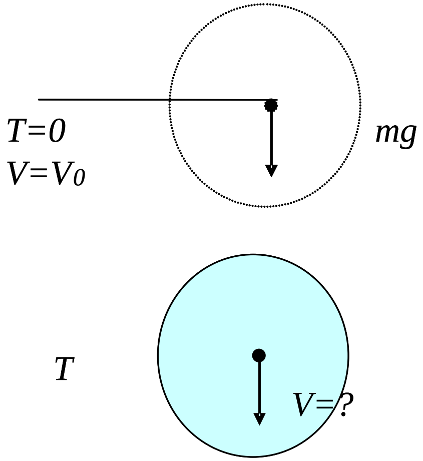
$$\dot{x} = \dot{x}_0 + \int \ddot{x}\, dt, \qquad \dot{y} = \dot{y}_0 + \int \ddot{y}\, dt$$

$$\dot{\theta} = \dot{\theta}_0 + \int \ddot{\theta}\, dt$$

$$x = x_0 + \int \dot{x}_0\, dt + \iint \ddot{x}\, dt^2$$

$$y = y_0 + \int \dot{y}_0\, dt + \iint \ddot{y}\, dt^2$$

$$\theta = \theta_0 + \int \dot{\theta}_0\, dt + \iint \ddot{\theta}\, dt^2$$

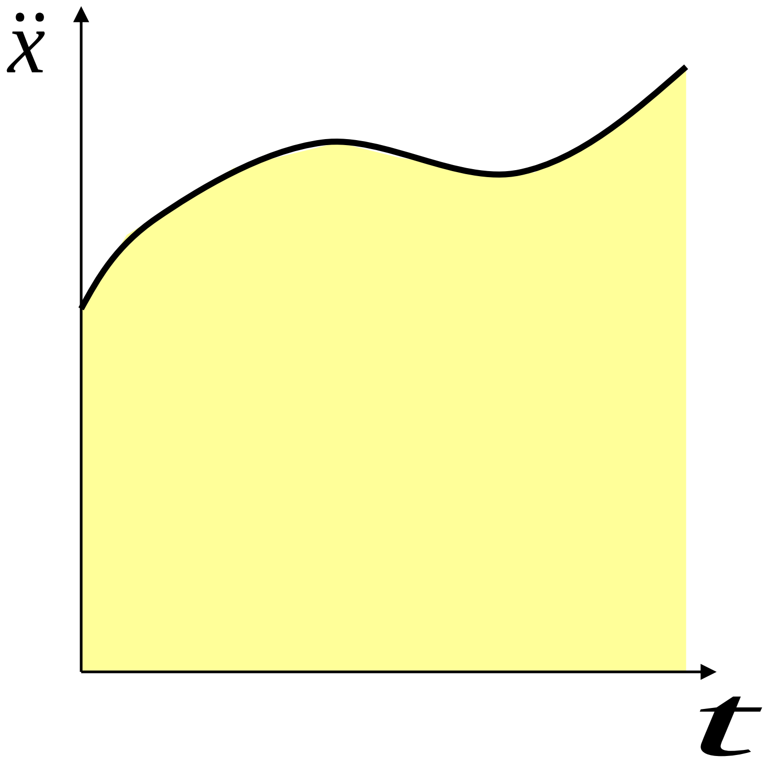


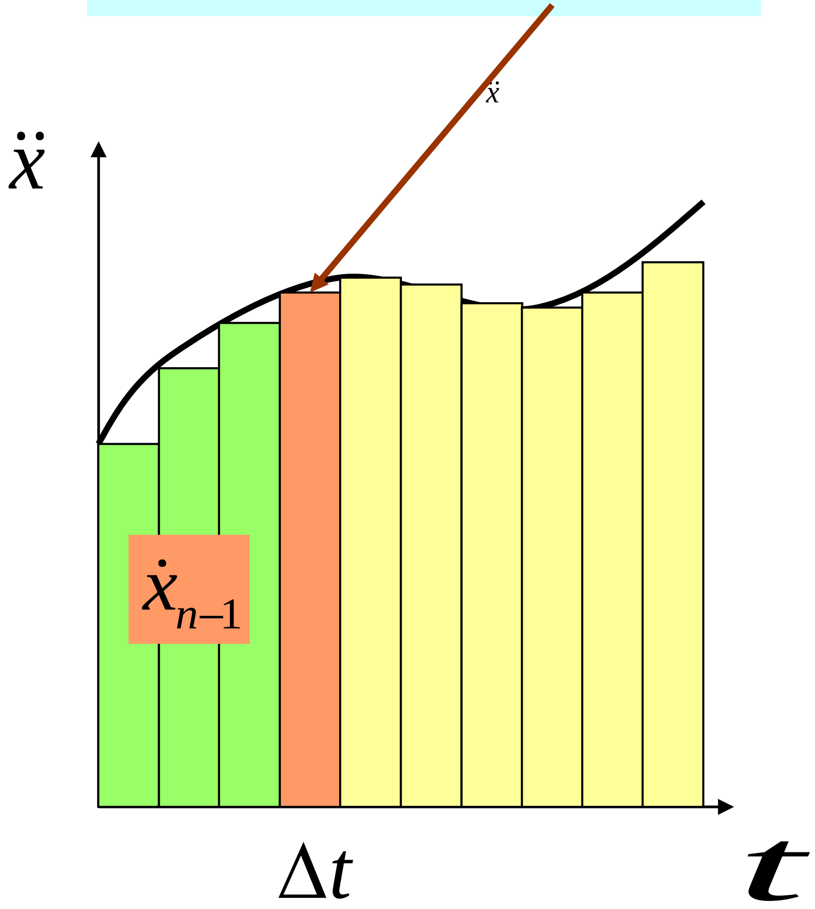
$$h = \frac{1}{2} g t^2$$?

Calculation of Numerical Integration

$$\dot{x} = \dot{x}_0 + \int \ddot{x}\, dt$$

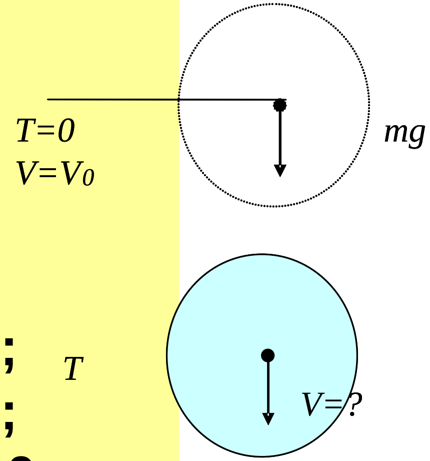
$$\dot{x}_n = \dot{x}_{n-1} + \ddot{x}_{n-1}\Delta t$$

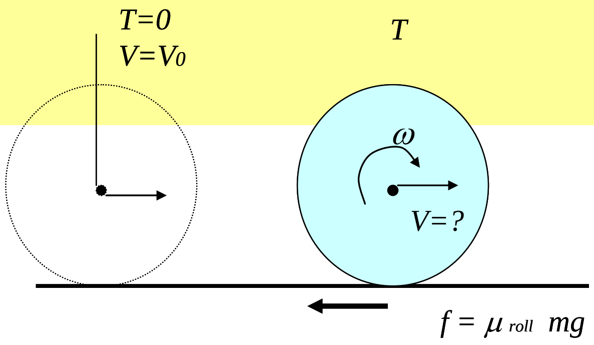




Calculation of Dynamics

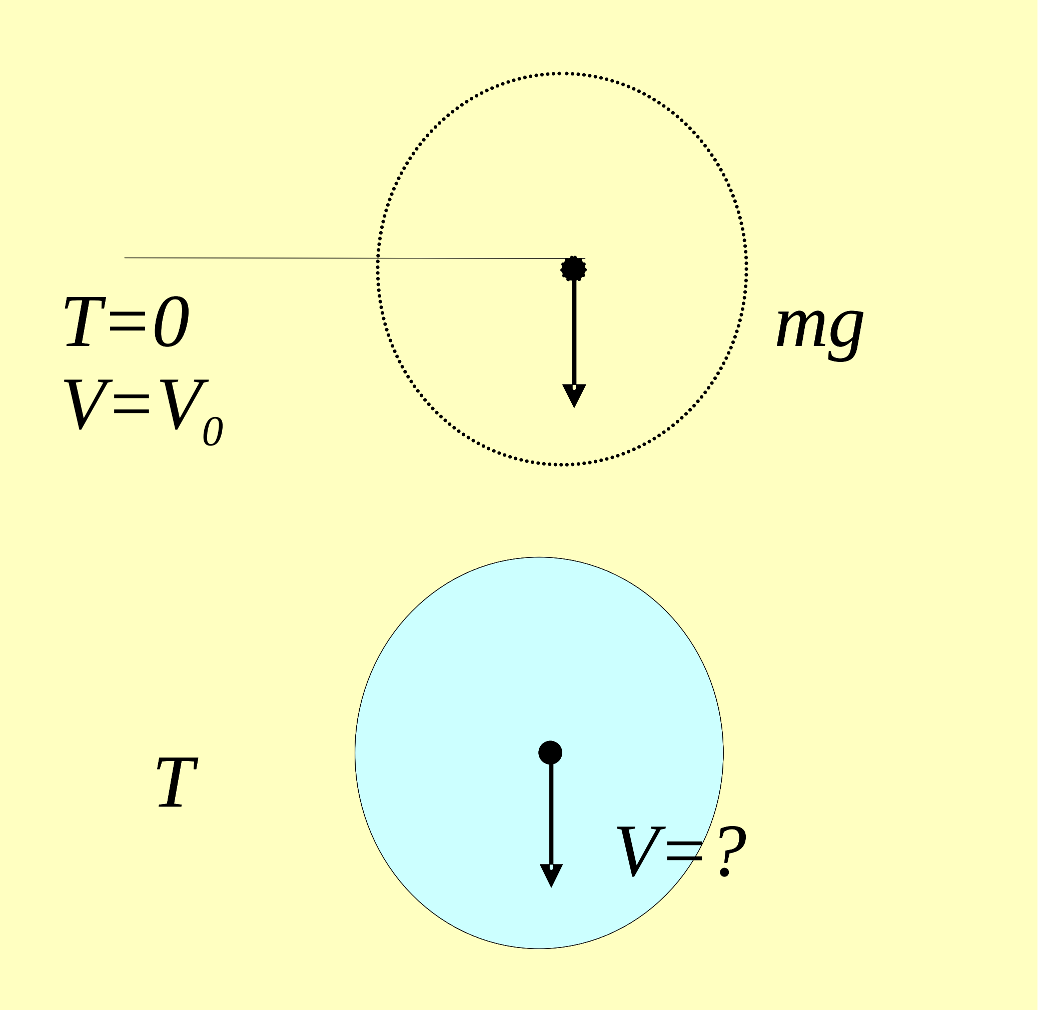
```
b1.ddx    = f_x / b1.m;
b1.ddy    = f_y / b1.m;
b1.ddtht = tai_z / b1.I;

b1.x    = b1.x   + b1.dx * dt     +  b1.ddx * dt * dt / 2.0;
b1.y    = b1.y   + b1.dy * dt     +  b1.ddy * dt * dt / 2.0;
b1.tht = b1.tht + b1.dtht * dt  +  b1.ddtht * dt * dt / 2.0;

b1.dx   = b1.dx   + b1.ddx * dt;
b1.dy   = b1.dy   + b1.ddy * dt;
b1.dtht = b1.dtht + b1.ddtht * dt;

t = t + dt;
```





μ_{roll} : *rolling friction coefficient*

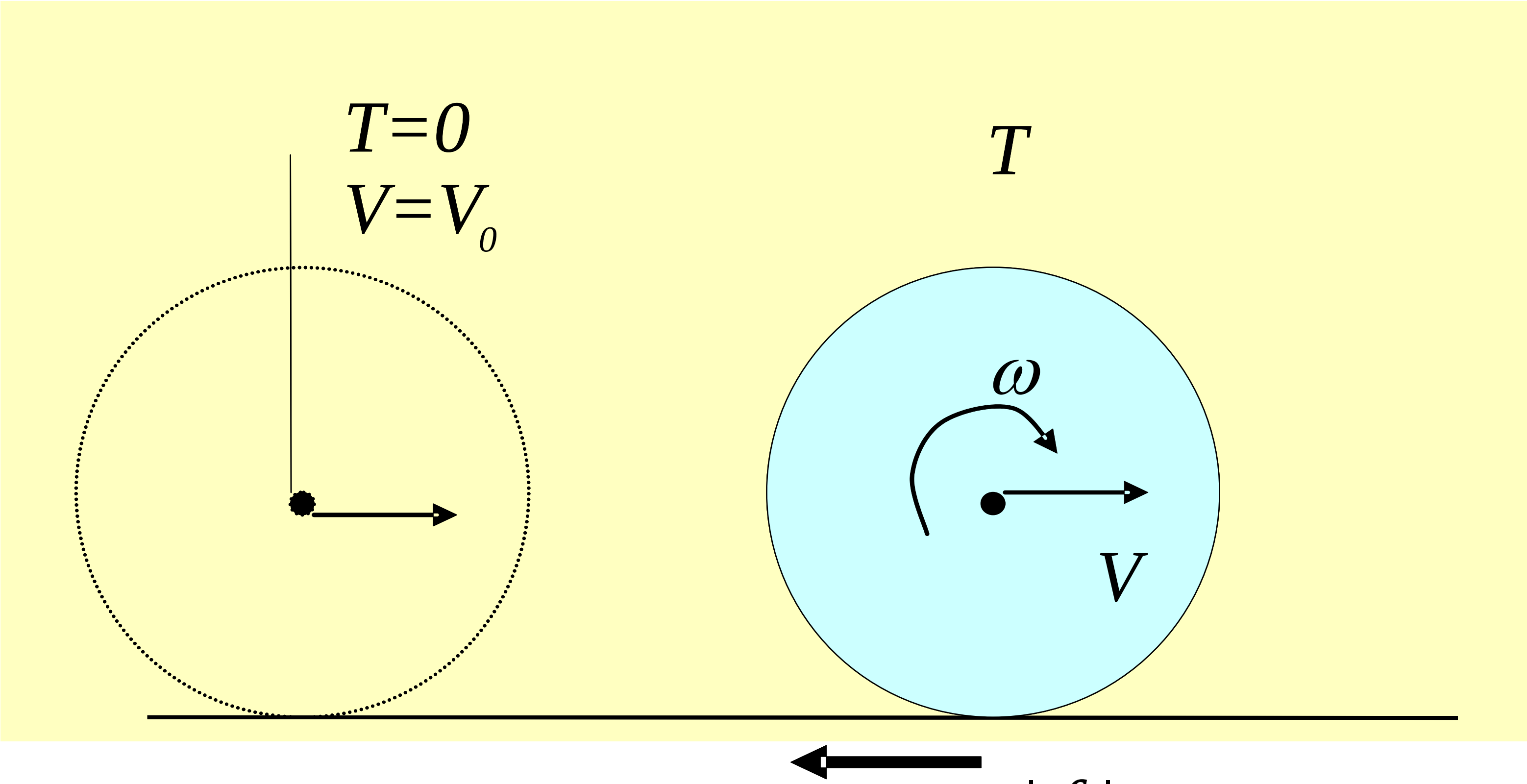
Free Motions of Ball



$$f_x=0.0$$

$$f_y= - mg$$

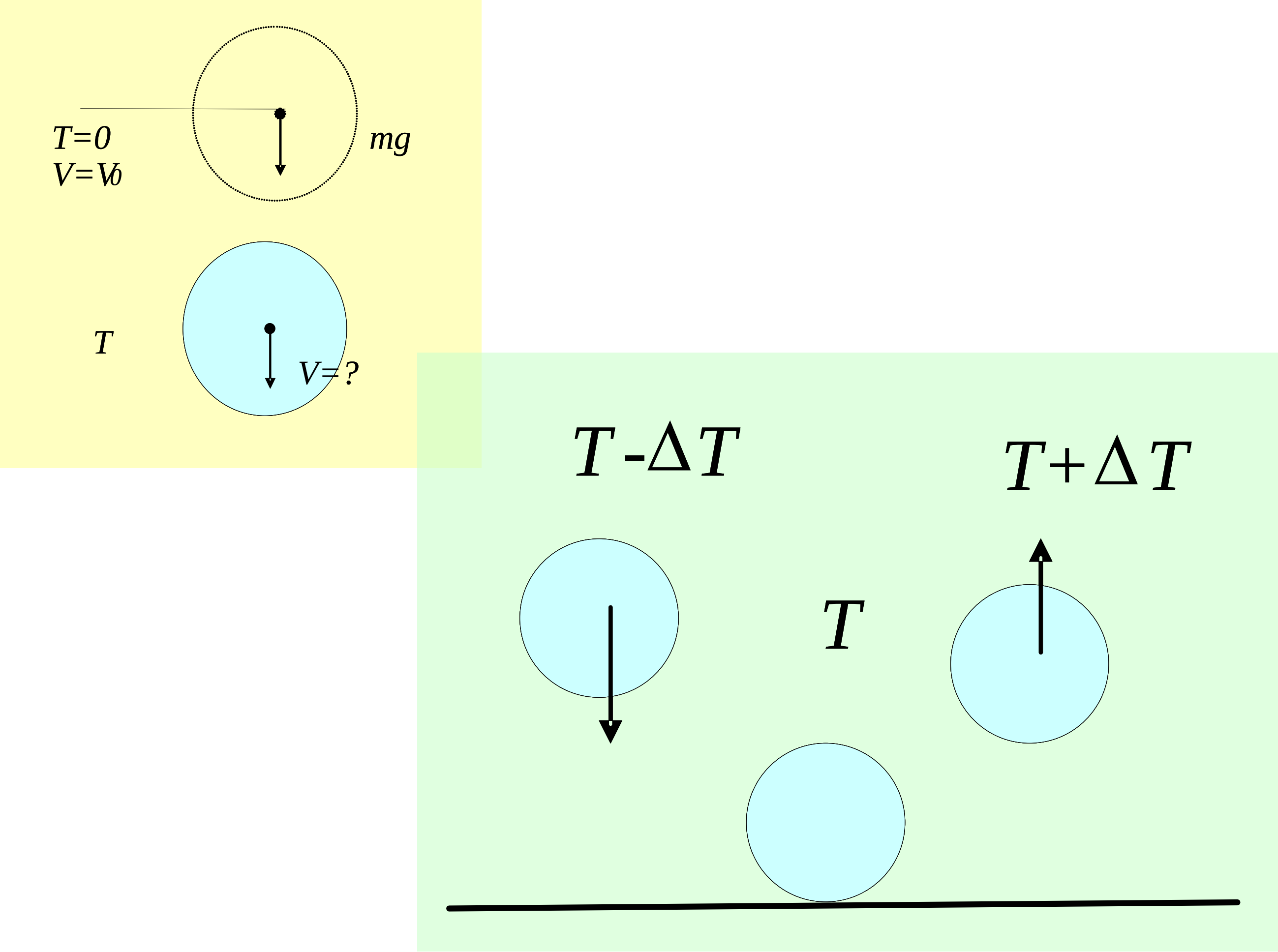
Rolling Motions of Ball

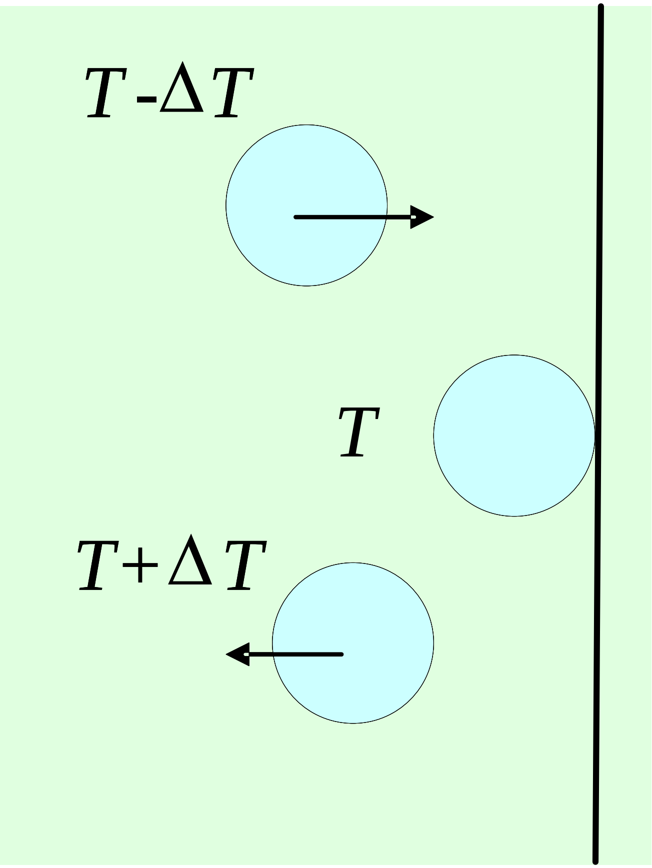


$$|f| = \mu_{roll} \ \ mg$$

μ_{roll} : *rolling friction coefficient*

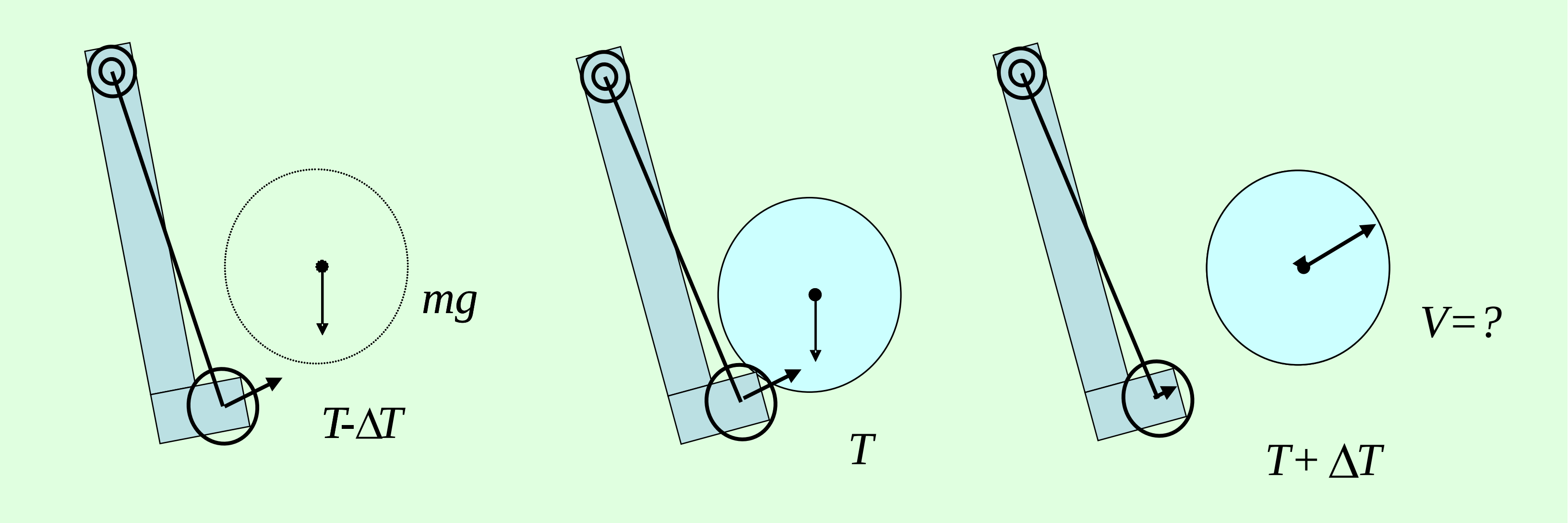
Bouncing Motions of Ball

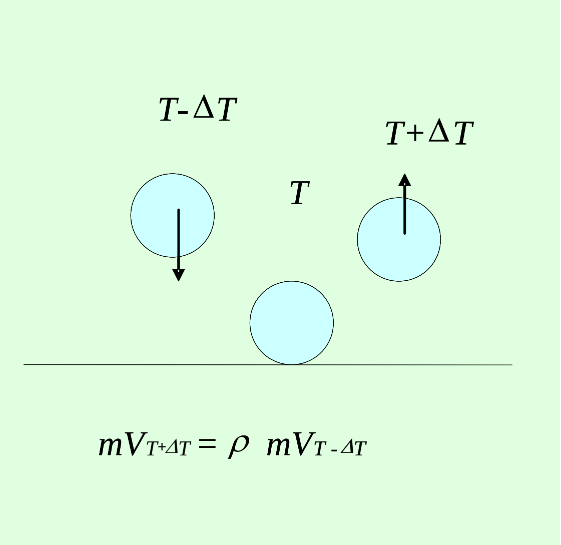




$$mV_{T+\Delta T} = \rho\ mV_{T-\Delta T}$$

Kicking Motions of Ball and Foot

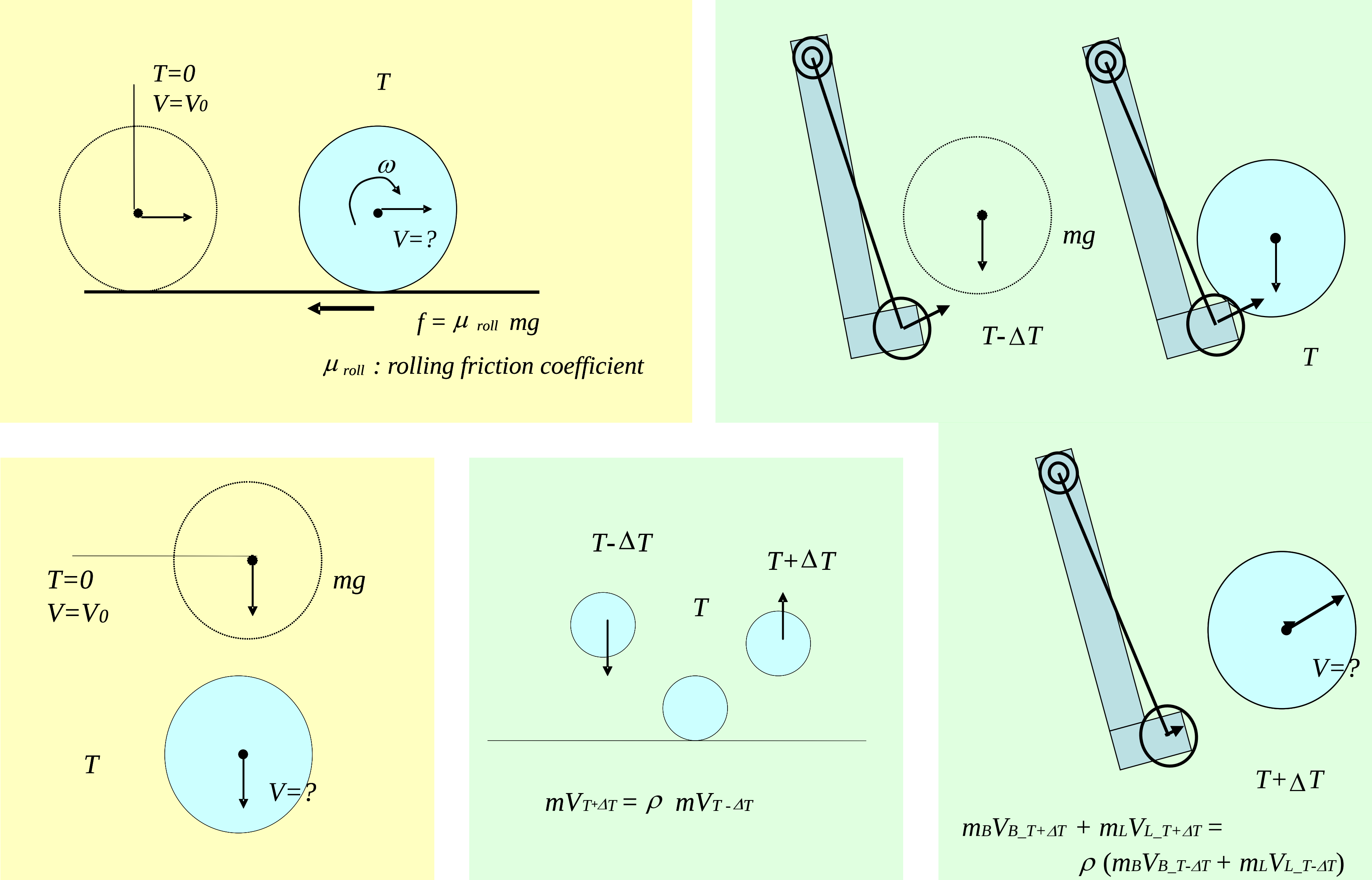




$$mV_{T+\Delta T} = \rho\ mV_{T-\Delta T}$$

$$m_B V_{B\_T+\Delta T} + m_L V_{L\_T+\Delta T} = \rho\,(m_B V_{B\_T-\Delta T} + m_L V_{L\_T-\Delta T})$$

Motions of Ball and Foot



$f = \mu_{roll}\ mg$

μ_{roll} : rolling friction coefficient

$$mV_{T+\Delta T} = \rho\ mV_{T-\Delta T}$$

$$m_B V_{B\_T+\Delta T} + m_L V_{L\_T+\Delta T} = \rho\ (m_B V_{B\_T-\Delta T} + m_L V_{L\_T-\Delta T})$$

Programmer-Defined Function

Define a Function

```
関数値のデータ型名    関数名（引数１のデータ型名　引数１，
                              引数２のデータ型名　引数２，
                              …
                              引数nのデータ型名　引数n）
{
      関数内で用いるデータの宣言部分

      関数の実行部分

     return( 関数値);
}
```

```
int add(int x, int y)
{
     int  sum;

     sum = x + y;

     return (sum);
}
```

```
long factorial(int x) /*  func of x!   */
{
     int  i;  long f;

     i = 0; f=1;
     while( i< x )       ++i; f=f*i;

     return (f);
}
```

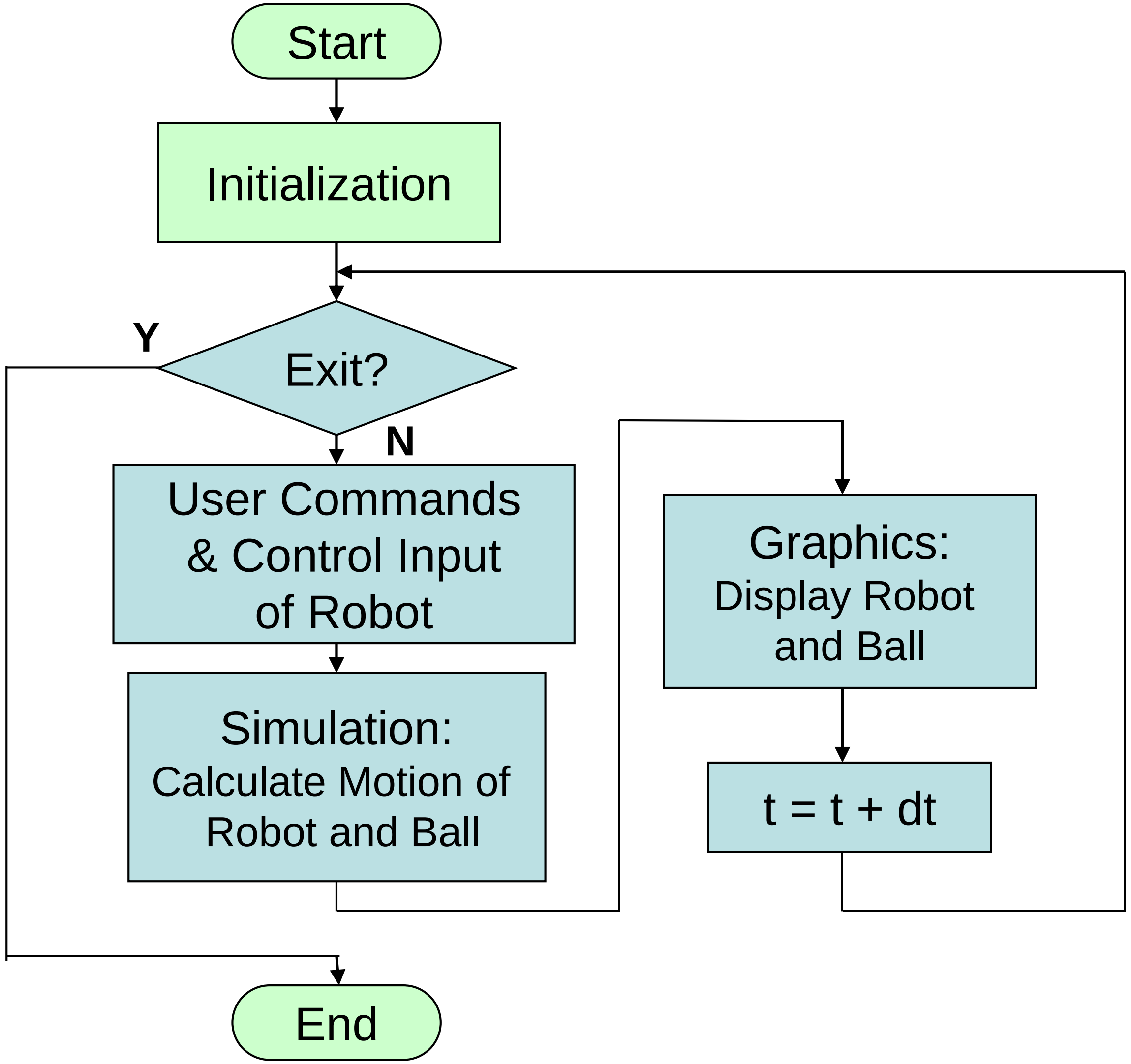
Function

```
関数値のデータ型名　　関数名（引数1のデータ型名　引数1，
                                   …
                                 引数 n のデータ型名　引数n）
{
     関数内で用いるデータの宣言部分

     関数の実行部分

    return( 関数値);
}
```

```
void　　手続き名（引数1のデータ型名　引数1，
                         …
                       引数nのデータ型名　引数n）
{
     手続き内で用いるデータの宣言部分

     手続きの実行部分
}
```

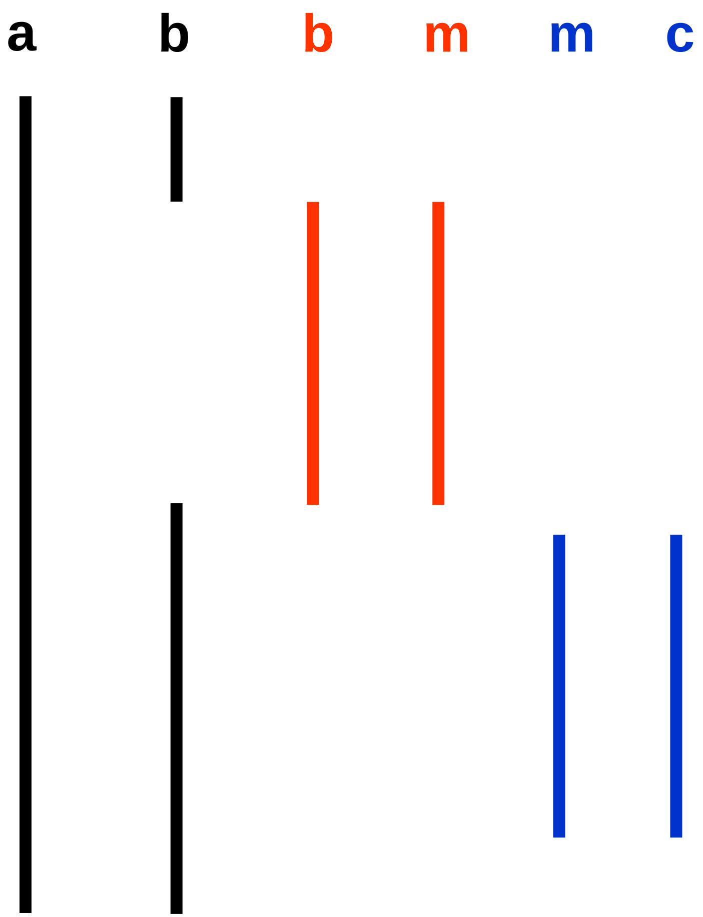
Global and Local Variable in Simulation

Program Structure of Robo Kick Simulator


Start
Initialization
Exit?
Y
N
User Commands & Control Input of Robot
Simulation: Calculate Motion of Robot and Ball
Graphics: Display Robot and Ball
t = t + dt
End

Variables in Functions

```
#include <stdio.h>
double a, b;     /* global var   */
double f1(int x)  /* func   */
{
     int  b;  double m; /* local var */
   …
}

main()                  /* main func   */
{
     int m, c;      /* local var  */
    …
     f1(m);
…
}
```



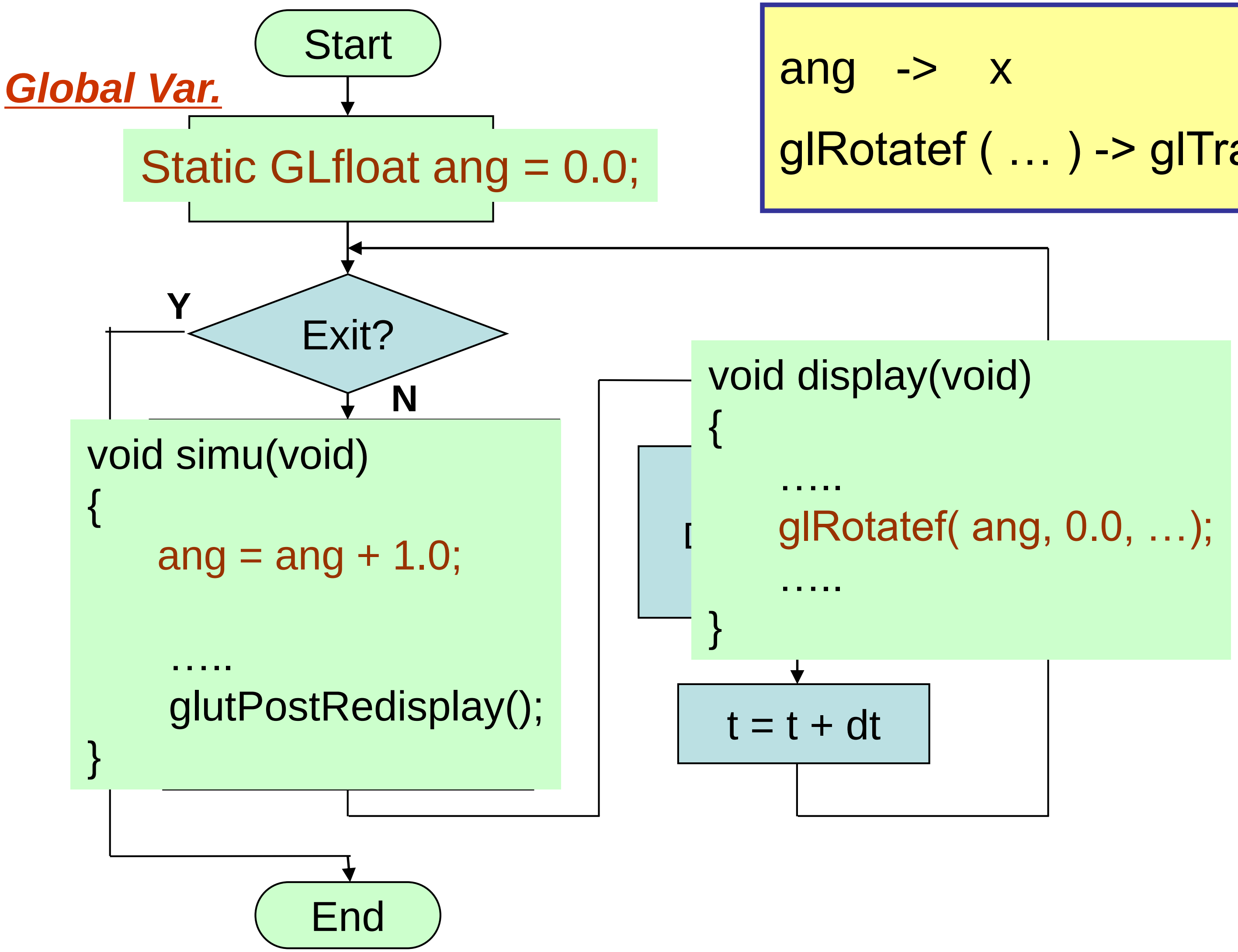
Calculation of Object Motion

```
static GLfloat ang = 0.0;    <- Initial Condition


void simu(void)
 {
        ang = ang + 1.0;     <- ang increases 1 each loop
        if ( ang > 360.0 )
                ang = ang - 360.0;
        glutPostRedisplay();
}
```

Example of Robo-Kick Simulator

ang -> x

glRotatef (…) -> glTranslatef(…)



Global Var.

```
Static GLfloat ang = 0.0;
```

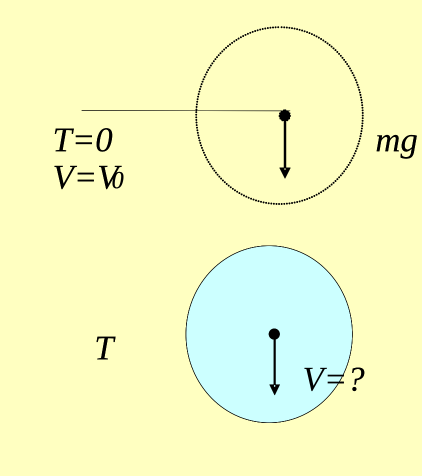
```
void simu(void)
{
     ang = ang + 1.0;

      …..
      glutPostRedisplay();
}
```

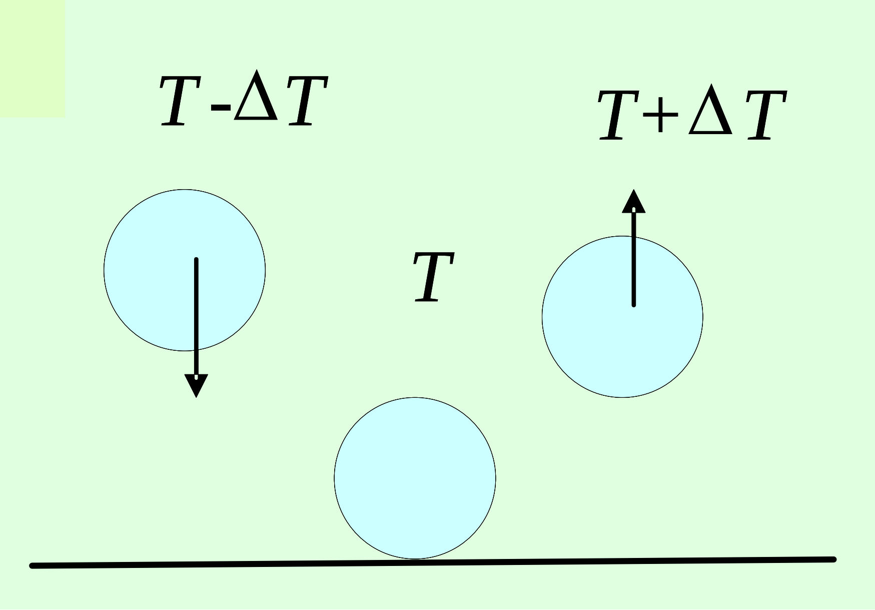
```
void display(void)
{
     …..
     glRotatef( ang, 0.0, …);
     …..
}
```

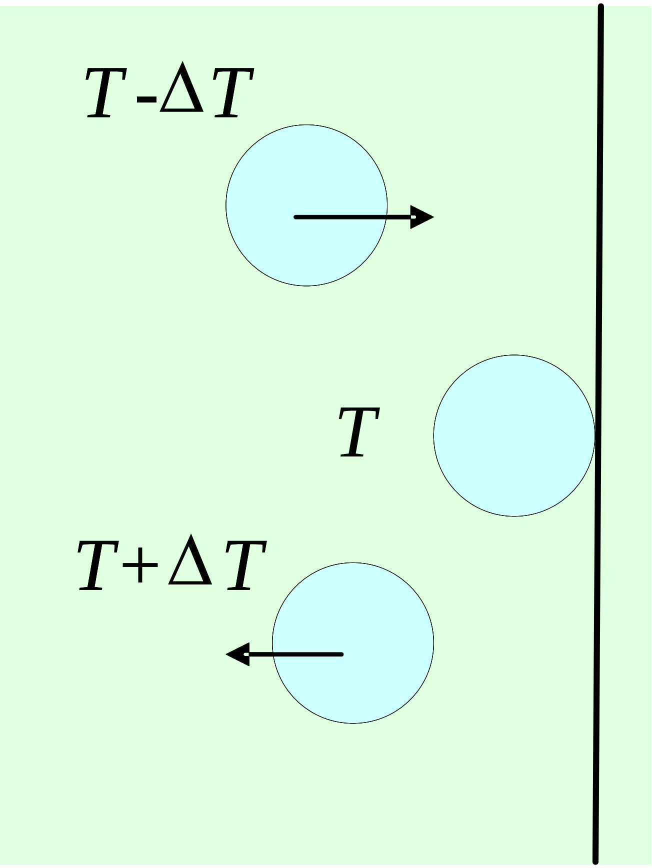
Changing the Direction of Rotation

and Simulating Bouncing Ball

Bouncing Motions of Ball

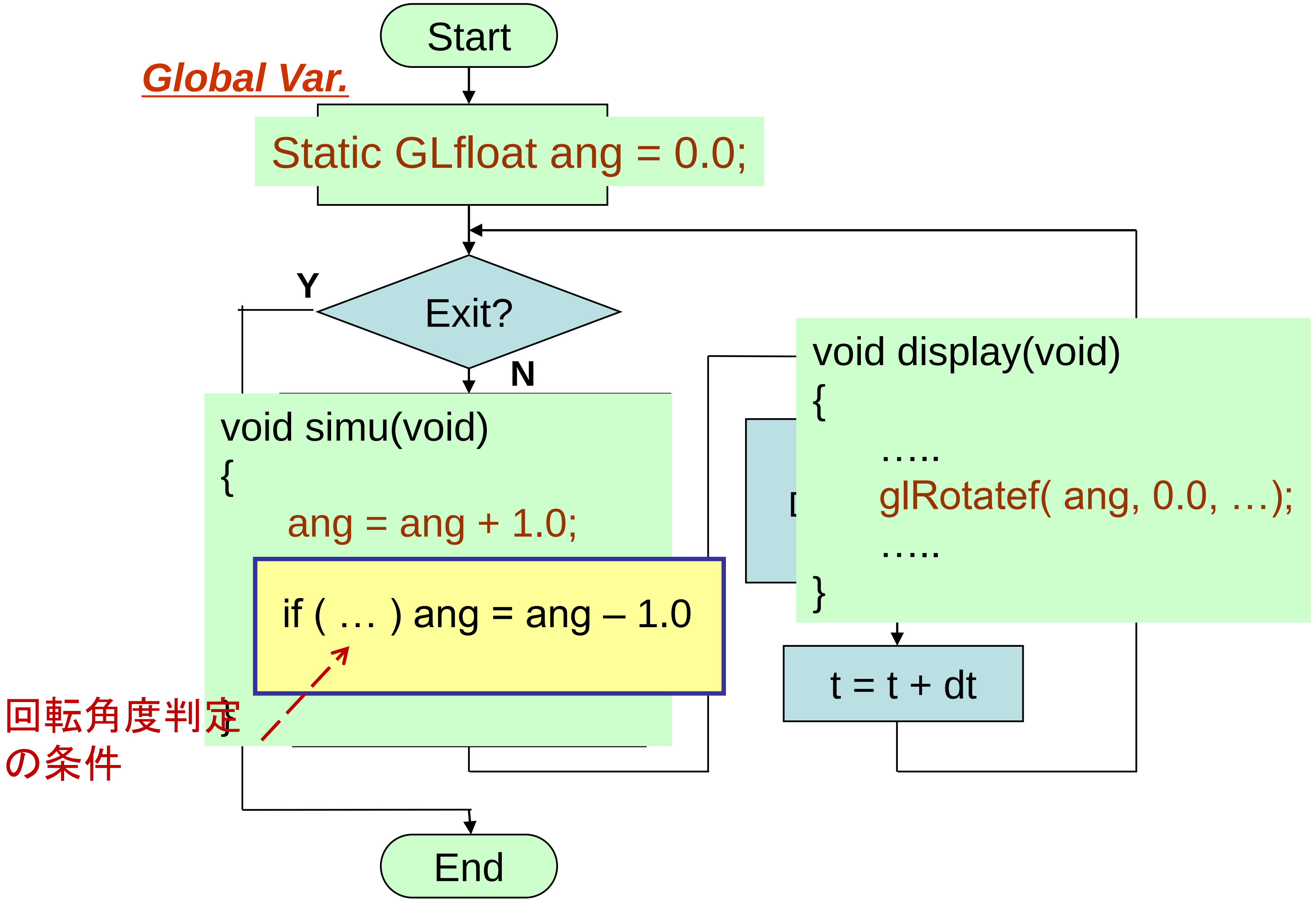






$$mV_{T+\Delta T} = - mV_{T-\Delta T}$$

How to change the direction of rotation



Global Var.

Start

```
Static GLfloat ang = 0.0;
```

Exit?

Y

N

```
void simu(void)
{
      ang = ang + 1.0;
      if ( … ) ang = ang – 1.0
}
```

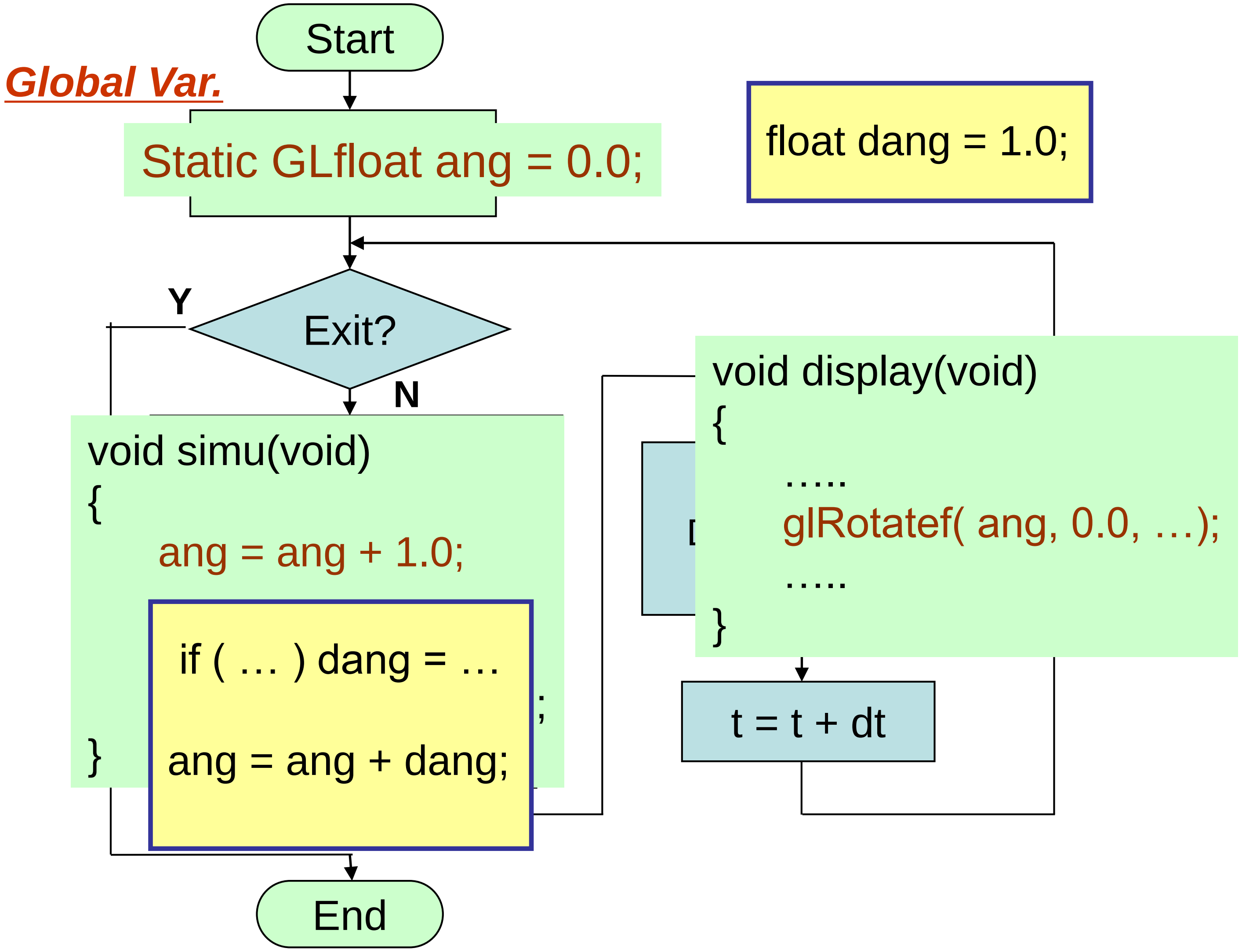
回転角度判定の条件

```
void display(void)
{
      …..
      glRotatef( ang, 0.0, …);
      …..
}
```

t = t + dt

End

How to change the direction of rotation



Start

Global Var.

```
Static GLfloat ang = 0.0;
```

```
float dang = 1.0;
```

Exit?

Y

N

```
void simu(void)
{
    ang = ang + 1.0;
}
```

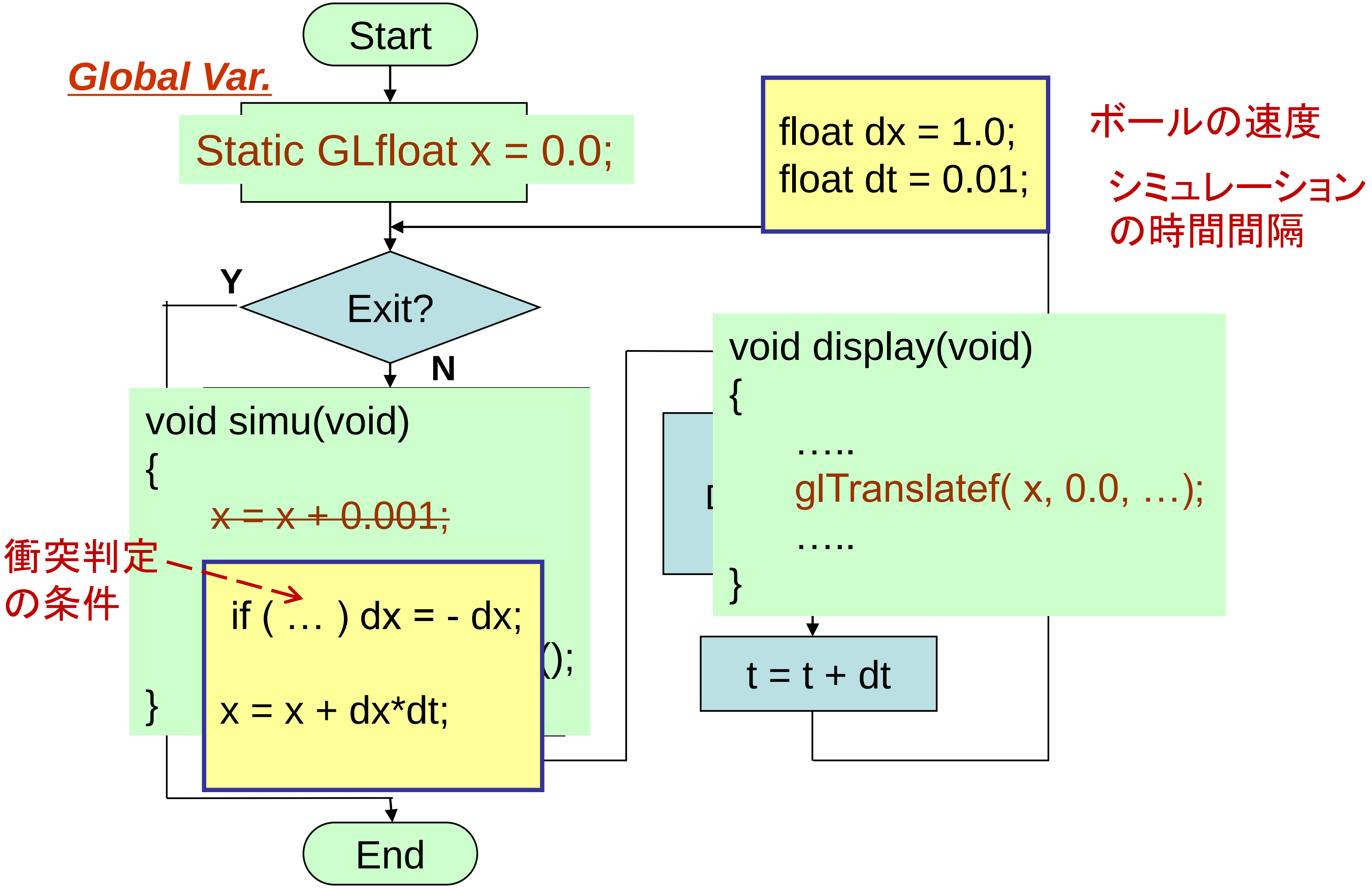
```
if ( … ) dang = …

ang = ang + dang;
```

```
void display(void)
{
    …..
    glRotatef( ang, 0.0, …);
    …..
}
```

t = t + dt

End

How to change the direction of translation



Start

Global Var.

Static GLfloat x = 0.0;

```
float dx = 1.0;
float dt = 0.01;
```

ボールの速度

シミュレーション
の時間間隔

Exit?

Y

N

```
void simu(void)
{
    ~~x = x + 0.001;~~
```

衝突判定
の条件

```
if ( … ) dx = - dx;

x = x + dx*dt;
```

();

}

```
void display(void)
{
    …..
    glTranslatef( x, 0.0, …);
    …..
}
```

t = t + dt

End