Robot Programming

Zhidong Wang

Nov. 10, 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/

On Steps

toward Robo-Kick Simulator

Essential Technologies

Data + Modeling + Graphics(CG)

Data

- **Data Main-Body** **データ本体**
- Properties, Status 特性、状態
- Associated Links 関連情報リンク

Data Type: Struct

構造体

Represent data as a structure

```
struct  complex {
          double        re;
          double        im;
};
```

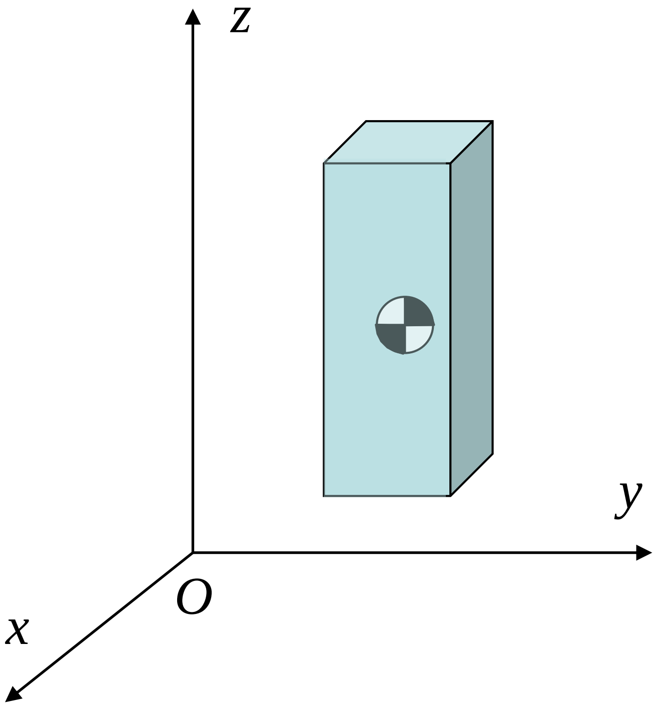
構造体の定義

double

```
struct  complex  a, b;

a.re  = 1.0;
a.im =  2.3;

b.re  =  a.re + 2.0;
b.im  =  a.im – 3.0;
```

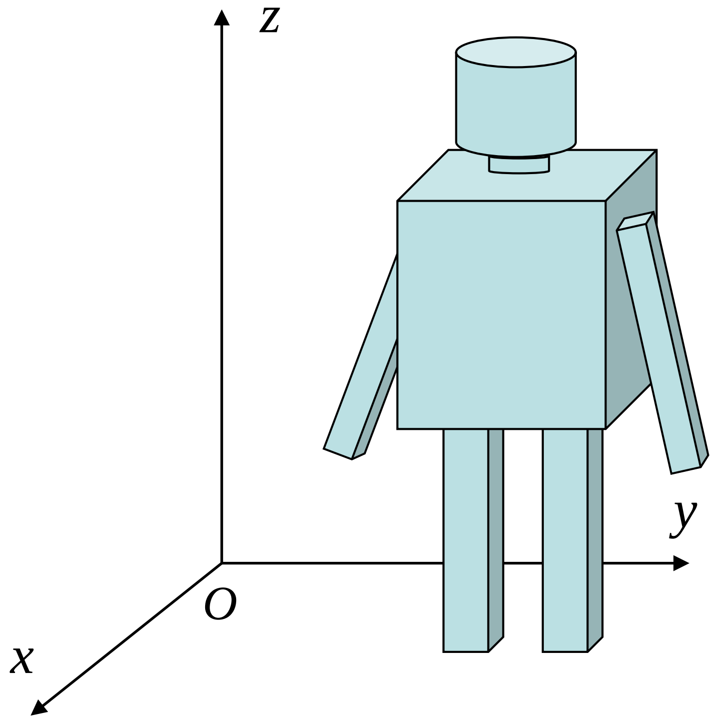
The better thing to use data type: Struct

```
struct  rigidBody {
          double        x,      y,     theta;
          double       dx,    dy,   dtheta;
          double      ddx, ddy, ddtheta;
};

struct  rigidBody   foot;

foot.x  = 1.0;         foot.y  =  0.0;
foot.theta = 90.0;

foot.dx  = 0.5;      foot.dy  =  1.0;
foot.dtheta = -5.0;
…….
```



The better thing to use data type: Struct

```
struct  rigidBody {
             double        x,      y,     theta;
             double       dx,    dy,   dtheta;
             double      ddx, ddy, ddtheta;
};

struct   robot {
              int     id;
             struct  rigidBody     head,  body;
             struct  rigidBody      armL,  armR;
             struct  rigidBody       legL,  legR;
};

strcut   robot    rbt1;

rbt1.id = 1;
rbt1.legL.x  = 1.0;        rbt1.armR.y  =  3.0;
…….
```



Data Type: Struct

```
struct  person {
          int       id;
          char   name[40];
          long   phone;
};

struct  person  student;

printf(“%d  %s  %ld¥n”,  student.id,
                                 student.name,
                                 student.phone);
```

Sub Project

Build a program which has 5 persons’ data with struct type. The program will print out all 5 persons’ information.

Data Type: Struct

```
struct  person {
          int      id;
          char   name[40];
          long   phone;
};

struct  person  student;

scanf(“%d  %s  %ld”,  &student.id,
                                student.name,
                               &student.phone);
```

Sub Project

Build a program which has 5 persons’ data with struct type. The program will print out all 5 persons’ information.

Array of Struct Variables

```
 double a;
 double  b[5];

 struct person student;
 struct person std[5];

std[0].id=0;
std[0].phone= 08011112222
```

Read and Write to a File

Read/Write Data from File

```
FILE     *fp;
int        id;
char      name[40];

fp = fopen( “datafile.txt”, “r” );

fscanf( fp, “%d  %s”,  &id, name );
fclose(fp);
```

File Point

Read Only Option

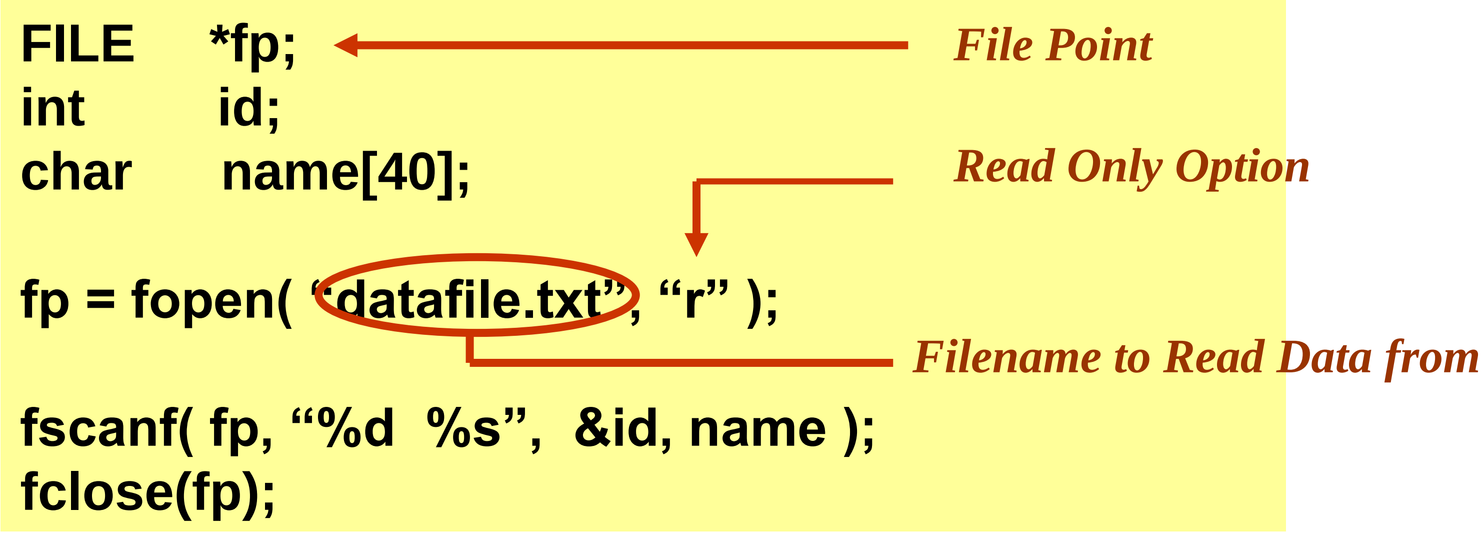
Filename to Read Data from

```
FILE     *fp;
int        id = 1234;
char      name[] = “I,robot”;

fp = fopen( “data¥¥datafile2.txt”, “w” );

fprintf( fp, “%d  %s”,  id, name );
fclose(fp);
```

Write Only Option

“rw” *read and write Option*

Read/Write Data from File



```
FILE     *fp;
int        id;
char      name[40];

fp = fopen( “datafile.txt”, “r” );

fscanf( fp, “%d  %s”,  &id, name );
fclose(fp);
```

Sub Project 2

Build a program which read 5 persons’ data from a file.

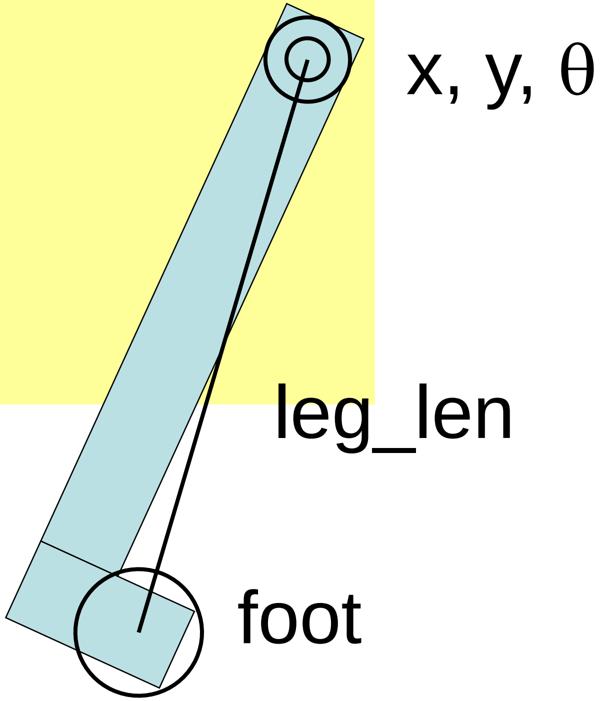
(The program can input new data and save them to a file.)

Data Structure: Ball

```
struct  ball {
        int       id;
        double   r;          /* radius  */
        double   m, I;       /* mass and inertial */
        double   x,     y,     tht;
        double   dx,   dy,   dtht;
        double   ddx, ddy, ddtht;
};

struct  ball  b1;
```

Data Structure: Leg

```
struct  leg {
          int                 id;
          double          joint_x, joint_y, leg_tht;
          struct ball   foot;
          double          leg_len;
};

struct  leg   r_leg;
```



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

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

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

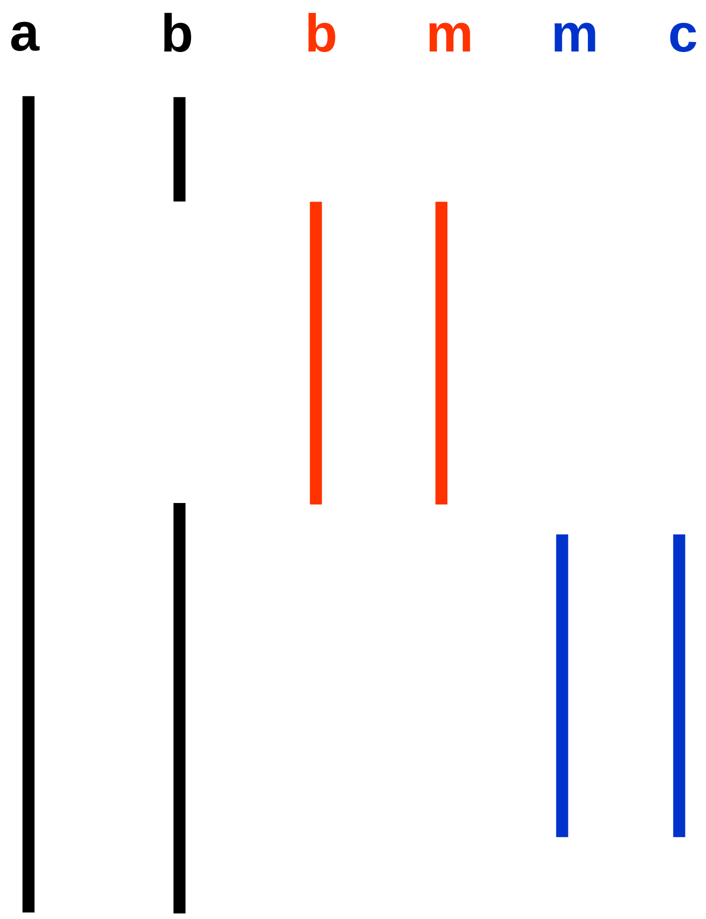
```
関数値のデータ型名　　関数名（引数1のデータ型名　引数1，
                                   …
                                  引数 n のデータ型名　引数n）
{
     関数内で用いるデータの宣言部分

     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
}
```

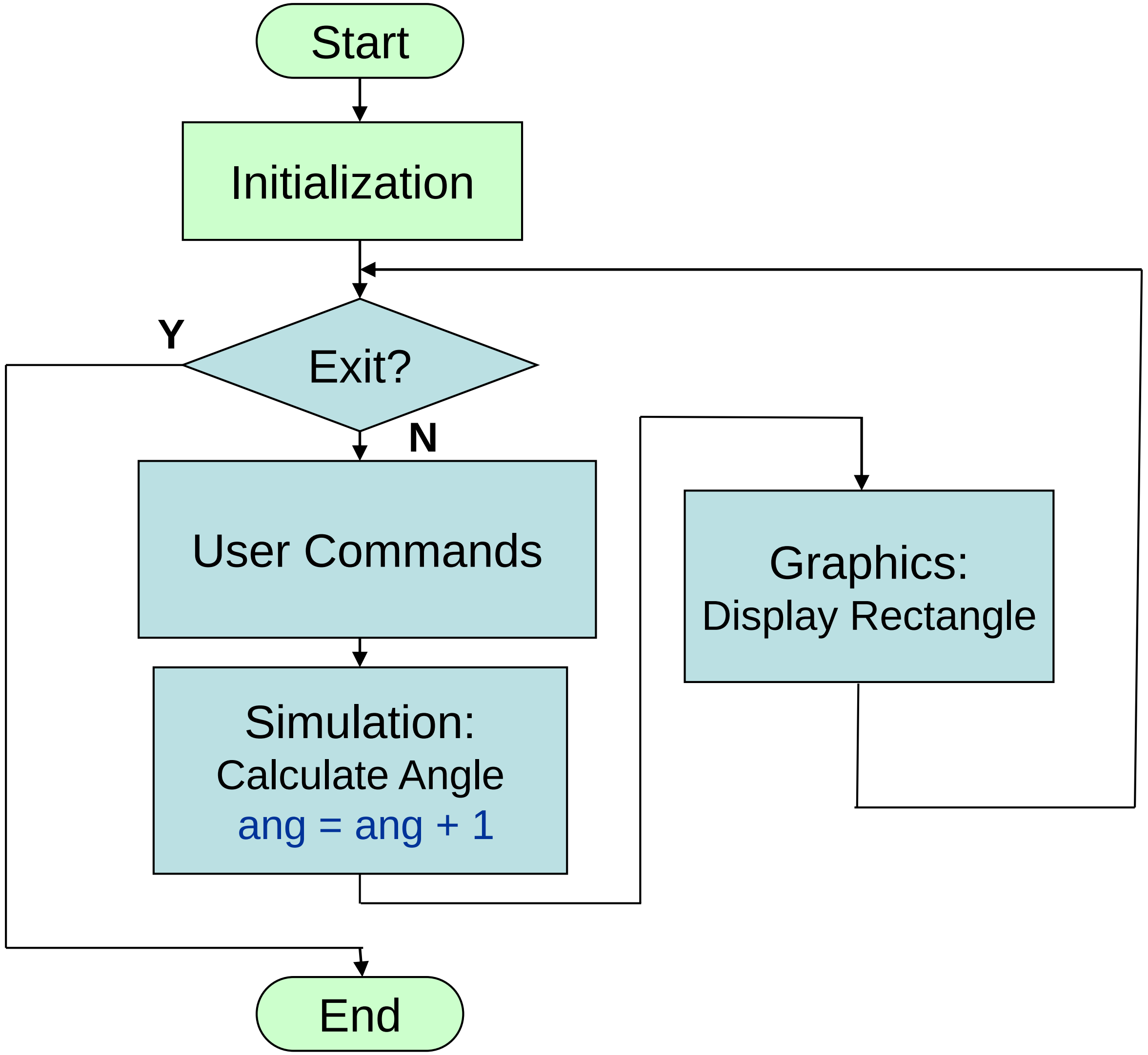
Global and Local Variable

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);
…
}
```



Program Structure of Sample 3



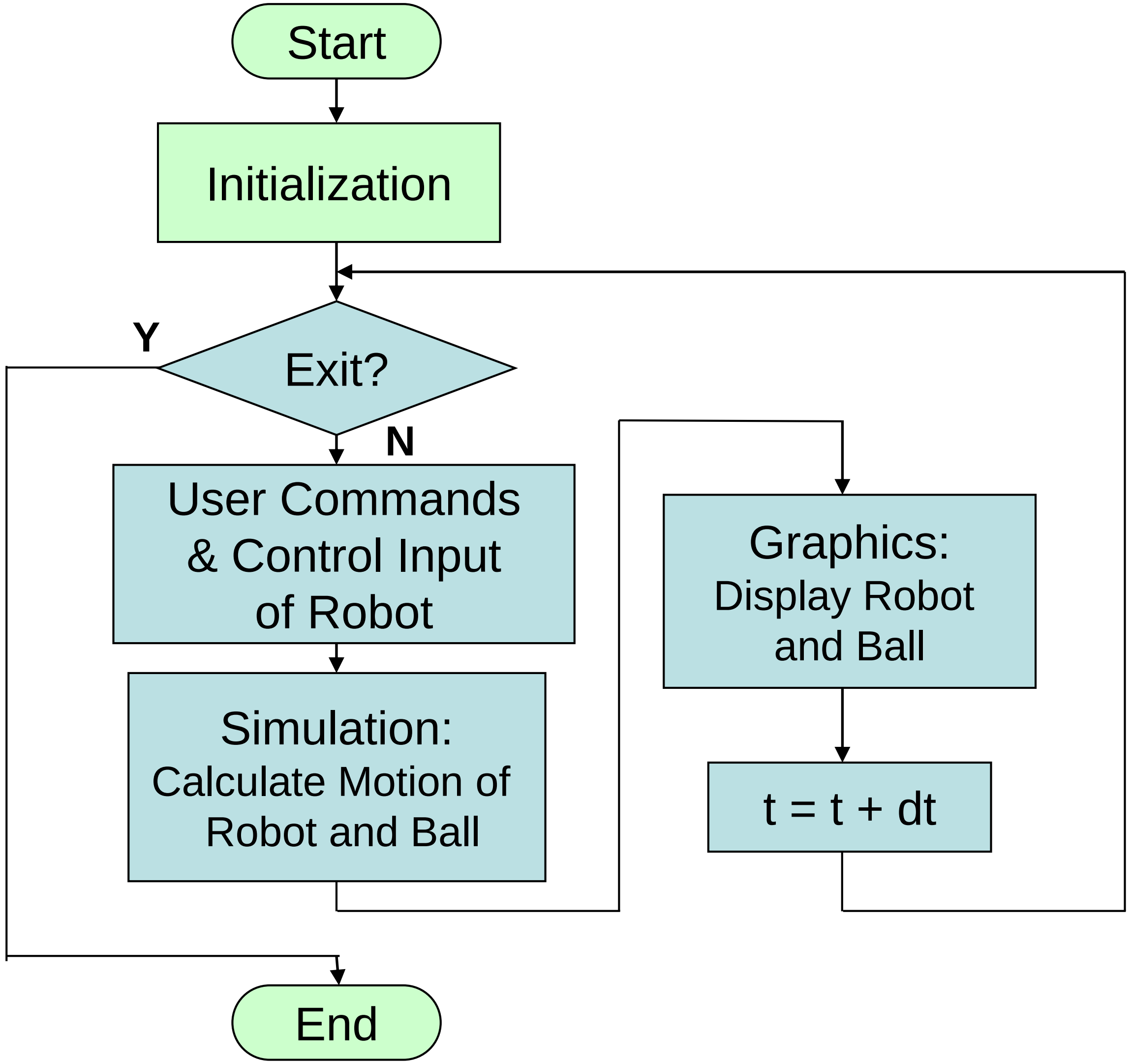
Calculation of Object Motion

```
static GLfloat ang = 0.0;


void simu(void)
 {
        ang = ang + 1.0;
        if ( ang > 360.0 )
                ang = ang - 360.0;
        glutPostRedisplay();
}
```

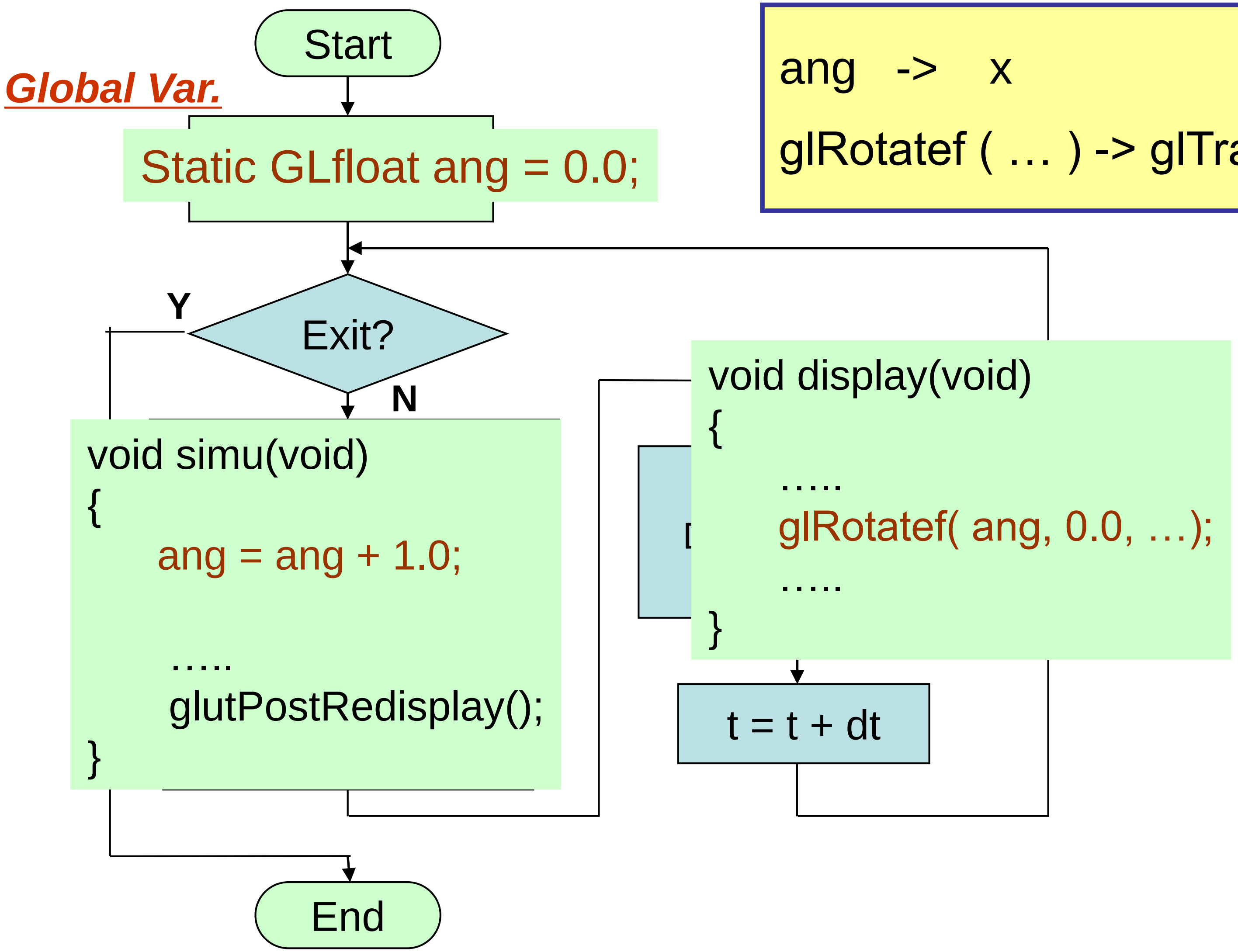
<- Initial Condition

<- ang increases 1 each loop

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

Example of Robo-Kick Simulator



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

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

      …..
      glutPostRedisplay();
}
```

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

ang -> x

glRotatef (…) -> glTranslatef(…)

Dynamics
and its Implementation

Essential Technologies

Data + Modeling + Graphics(CG)

Calculation of Dynamics

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

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

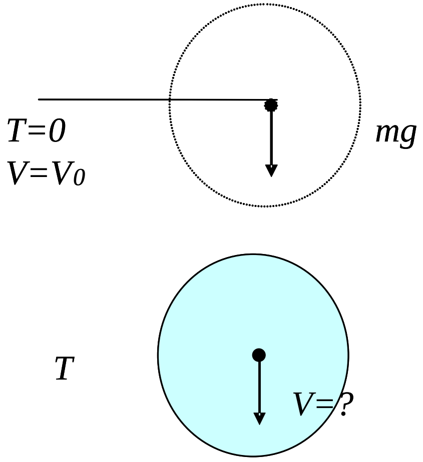
$$\dot{x} = \dot{x}_0 + \int \ddot{x}\, dt, \qquad \dot{y} = \dot{y}_0 + \int \ddot{y}\, dt$$

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

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

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

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

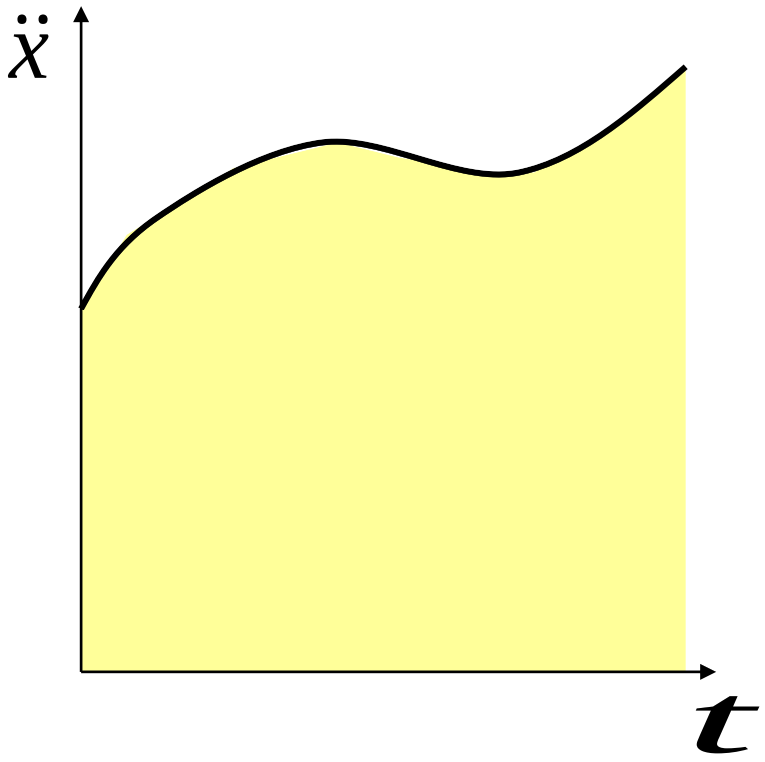


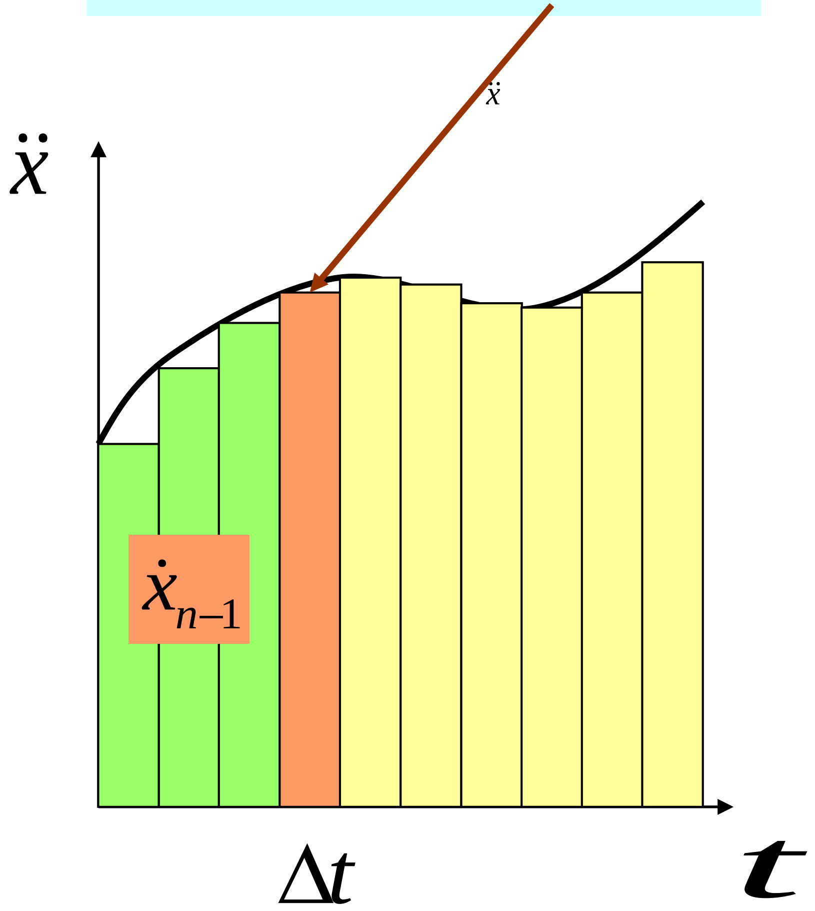
$$h = \frac{1}{2} g t^2$$?

Calculation of Numerical Integration

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

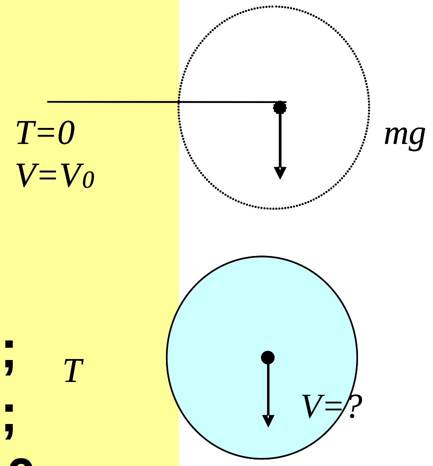
$$\dot{x}_n = \dot{x}_{n-1} + \ddot{x}_{n-1}\Delta t$$

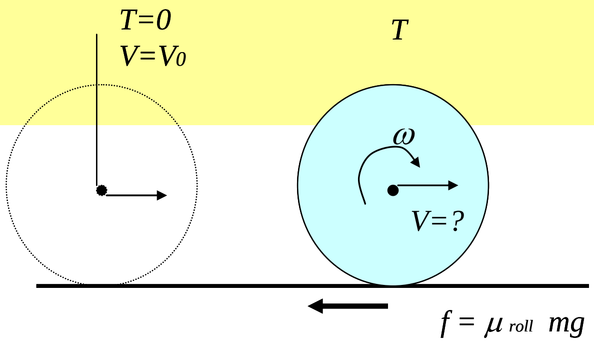




Calculation of Dynamics

```
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*