2017/07/11
情報メディア学科佐藤尚

2017 年度情報メディア基盤ユニット補足資料

物体を移動させる方法としては、次のような 4 つの方法が考えられます。

1) 一定の割合で目標位置に近づけていく、
2) 位置を計算する関数を決めて動かす、
3) 速度を指定して物体を動かす、
4) 速度と加速度を指定して物体を動かす。

1)の方法でプログラムを作るためには、2 点を内分する点を求める公式を利用します。つぎのような式を利用すると 2 点の間の点の座標を求めることが出来ます。この式では、パラメータ t の値は 0 以上 1 以下とすることが一般的です。

$$\begin{pmatrix} x \\ y \end{pmatrix} = (1-t)\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + t\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

t の値が 0.5 であれば、2 点のちょうど真ん中の点となり、0.1 や 0.9 などあれば端点の近くに点がきます。この式を使って、作ったサンプルが次のサンプルプログラムです。

サンプルプログラム：内分の公式を利用して移動

```
1  float xBall;
2  float yBall;
3  void setup(){
4    size(600,600);
5    xBall = width/2;
6    yBall = height/2;
7  }
8  void draw(){
9    background(255);
10   fill(0);
11   float t = 0.5;
12   xBall = (1-t) * xBall + t*mouseX;
13   yBall = (1-t) * yBall + t*mouseY;
14   ellipse(xBall,yBall,20,20);
17 }
```

2)の方法では、色々な関数を利用することが考えられます。ここでは sin 関数を利用して、左右に移動しながら、落ちていくようなサンプルプログラムを作ってみました。

サンプルプログラム：三角関数を利用して移動

```
1  float xBall;
2  float yBall;
```

```
3  void setup(){
4    size(400,400);
5    xBall = width/2;
6    yBall = 0;
7  }
8  void draw(){
9    background(255);
10   float t = 2*PI*frameCount/(width/2);
11   xBall = width/2 + (width/3) * sin(t);
12   yBall++;
13   fill(0);
14   ellipse(xBall,yBall,20,20);
15 }
```

3)の速度を指定して物体を動かすというは、一定の速さ *v*m/s で移動している物体は一秒後には x+v(=x + 1 *v)に移動するという事実に基づいています。現在でのフレームでの位置に移動速度を足せば、次のフレームでの位置が求まります。これを使って、最初と同じような動きをするサンプルプログラムを作ってみました。

サンプルプログラム：速度を利用した物体の移動

```
1  float xBall;
2  float yBall;
3  float vx;
4  float vy;
5  float speed;
6  void setup(){
7    size(400,400);
8    xBall = width/2;
9    yBall = 0;
10   speed = 2;
11 }
12 void draw(){
13   background(255);
14   float d = dist(mouseX,mouseY,xBall,yBall);
15   float vx = speed * (mouseX - xBall) / d;
16   float vy = speed * (mouseY - yBall) / d;
17   xBall = xBall + vx;
18   yBall = yBall + vy;
19   fill(0);
20   ellipse(xBall,yBall,20,20);
21 }
```

速度による移動は幅広く応用可能です。移動速度を三角関数によって決めたサンプルを示します。

サンプルプログラム：速度を利用した物体の移動（三角関数）

```
1  float xBall;
2  float yBall;
3  float vx;
```

```
float vy;
float speed;
void setup(){
  size(400,400);
  speed = 2;
  vx = 0;
  vy = 0;
  xBall = width/2;
  yBall = height/2;
}
void draw(){
  background(255);
  xBall = xBall + vx;
  yBall = yBall + vy;
  fill(0);
  ellipse(xBall,yBall,20,20);
}
void mouseClicked(){
  float angle = radians(frameCount % 360);
  println(frameCount);
  vx = speed * cos(angle);
  vy = speed * sin(angle);
  xBall = width/2;
  yBall = height/2;
}
```

サンプルプログラム：速度を利用した物体の移動（三角関数＆沢山）

```
float[] xBall;
float[] yBall;
float[] vx;
float[] vy;
float speed;
void setup(){
  size(400,400);
  speed = 2;
  vx = new float[12];
  vy = new float[12];
  xBall = new float[12];
  yBall = new float[12];
  for(int i=0;i < xBall.length;i++){
    vx[i] = 0;
    vy[i] = 0;
    xBall[i] = width/2;
    yBall[i] = height/2;
  }
}
void draw(){
  background(255);
  for(int i=0;i < xBall.length;i++){
    xBall[i] = xBall[i] + vx[i];
    yBall[i] = yBall[i] + vy[i];
  }
  fill(0);
  for(int i=0;i < xBall.length;i++){
    ellipse(xBall[i],yBall[i],20,20);
```

```
  }
}
void mouseClicked(){
  for(int i=0;i < xBall.length;i++){
    float angle = 2*PI*i/xBall.length;
    vx[i] = speed * cos(angle);
    vy[i] = speed * sin(angle);
    xBall[i] = width/2;
    yBall[i] = height/2;
  }
}
```

ゲームのサンプル

サンプルプログラム：早撃ちゲーム

```
PFont font;
float xTarget;
float yTarget;
int targetWidth;
int targetHeight;
color targetColor;

final int GAME_READY = 0;
final int GAME_UPDATING = 1;
final int GAME_RUNNING = 2;
final int GAME_HIT = 3;

int currentState;
int time0_msec;

void startElapsedTime(){
  time0_msec = millis();
}

int elapsedTime_msec(){
  return millis()-time0_msec;
}

void setup(){
  size(400,400);
  smooth();
  font = createFont("Serif",48);
  currentState = GAME_READY;
}

void updateTarget(){
  targetWidth = 50;
  targetHeight = 50;
  xTarget = random(targetWidth, width-targetWidth);
  yTarget = random(targetHeight, height-targetHeight);
  targetColor = color(10,10,255);
}

boolean isOnTarget(int x,int y){
  if((xTarget <= x && x < (xTarget+targetWidth)) &&
     (yTarget <= y && y < (yTarget+targetHeight))){
```

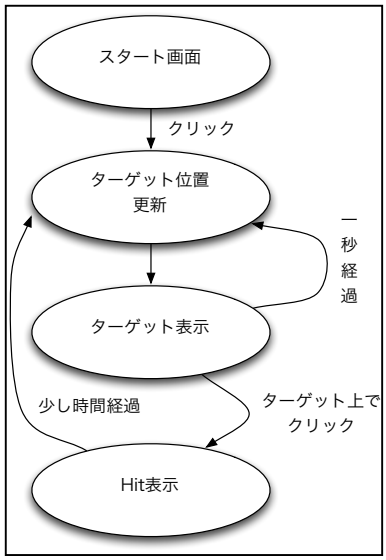
```
42      return true;
43     }else{
44      return false;
45     }
46  }
47
48  void showReadyMessage(){
49    textAlign(CENTER);
50    textFont(font,48);
51    fill(0);
52    text("Start to click",width/2,height/2);
53  }
54
55  void showHitMessage(){
56    textAlign(CENTER);
57    textFont(font,48);
58    fill(255,10,10);
59    text("Hit!!",width/2,height/2);
60  }
61
62  void showTarget(){
63    stroke(targetColor);
64    fill(targetColor);
65    rectMode(CORNER);
66    rect(xTarget,yTarget,
67        targetWidth,targetHeight);
68  }
69
70  void draw(){
71    background(255);
72    if(currentState == GAME_READY){
73      showReadyMessage();
74    }else if(currentState == GAME_UPDATING){
75      updateTarget();
76      currentState = GAME_RUNNING;
77      startElapsedTime();
78    }else if(currentState == GAME_RUNNING){
79      showTarget();
80      if(elapsedTime_msec() >= 1000){
81        currentState = GAME_UPDATING;
82      }
83    }else if(currentState == GAME_HIT){
84      showHitMessage();
85      if(elapsedTime_msec() >= 500){
86        currentState = GAME_UPDATING;
87      }
88    }
89  }
90
91  void mouseClicked(){
92    if(currentState == GAME_READY){
93      currentState = GAME_UPDATING;
94    }else if(currentState == GAME_RUNNING){
95      if(isOnTarget(mouseX,mouseY)){
96        currentState = GAME_HIT;
97        startElapsedTime();;
98      }
```

```
  }
}
```

上のプログラムは、

1) 最初は開始画面が表示する。
2) 表示されている正方形（ターゲット）をクリックしたら、hit 表示がされる。
3) hit 表示は少し経ったら(0.5 秒)、再び正方形（ターゲット）を表示する。
4) 正方形（ターゲット）は 1 秒間表示されたら、別な場所に移動する。

といった感じのゲームのようなプログラムです。



上の 1)から 4)から、このプログラムには、下の表のような 4 つの状態（状況）があると考えられます。マウスをクリックしたり、一定時間経ったら、これらの状態間を遷移します。状態から状態への遷移は、右の図のようになっています。このような状態間の遷移の状態を表す図のことを状態遷移図と呼んでいます。このような状態遷移図を作成すると、全体の進行状況を捉えやすくなります。このような状態遷移図をプログラム化するには、一般的に 2 つのやり方があります。

1) 表示や入力の処理をする関数内(draw などや mouseClicked 関数など)で、状態を条件分岐(if 命令など)で状態を変更する。
2) 状態遷移クラスを作成し、各状態を子クラスとして継承する。

このサンプルでは、1)の方法を利用してプログラムを作成しています。状態を表すために、各状態に数値を割り当て、状態を区別します。数字では、人間がどの状態かを識別ルことが難しいので、適当な変数名をつけた変数を利用しています。この値は変更さえることがないので、定数と呼ばれています。この状態を識別する定数として、上の表の状態定数名という名前の変数を使用しています。また、現在の状態は currentState という変数に保存されています。

状態名	説明	状態定数名
開始画面	開始画面を表示している	GAME_READY
ターゲット位置更新	正方形（ターゲット）の位置を変更している	GAME_UPDATING

ターゲット表示	正方形（ターゲット）を表示している	GAME_RUNNING
Hit 表示	Hit 表示を行っている	GAME_HIT

マウスをクリックして状態が変化する部分は、mouseClicked 関数で処理をしています。また、経過時間などで変化する部分は、draw 関数で処理をしています。

マウスクリック時の処理は、GAME_READY 状態であれば、直ちに GAME_UPDATE 状態に遷移します。また、GAME_RUNNING 状態であれば、マウスをクリックした場所が正方形（ターゲット）上であれば、GAME_HIT 状態に遷移します。経過時間を計るために、状態が遷移するときに、startElapsedTime 関数を呼び出し、elapsedTime_msec 関数で状態の経過時間をミリ秒単位で調べることが出来るようにしています。これらの処理は mouseClicked 関数内に書かれています。

GAME_UPDATE 状態であれば、正方形（ターゲット）の位置情報などを更新後、GAME_RUNNING 状態に遷移します。また、GAME_RUNNING 状態で、経過時間が 1 秒(1000 ミリ秒)経過したら、GAME_UPDATE 状態に遷移します。GAME_HIT 状態で、経過時間が 0.5 秒(500 ミリ秒)経過したら、GAME_UPDATE 状態に遷移します。これらの処理は、draw 関数内に書かれています。

アドベンチャーゲームなどは、この状態遷移から成り立っているゲームとなっています。複雑なゲームにおいて、完全な状態遷移図を書くことはかなり面倒ですが、大雑把なものであれば、状態遷移図を書くことは可能だと思います。何か適当なゲームを題材に、大雑把な状態遷移図を書いて見て下さい。または、自分でゲームのようなものを想定して、その状態遷移図を書いてもかまいません。

```
 1  void setup(){
 2    size(400,400);
 3  }
 4
 5  void draw(){
 6    background(255);
 7    fill(0,128,255);
 8    float t = (frameCount % 360)/360.0;
 9    float x = (1-t)*0 + t * width; // t * width
10    float y = (-(x-width/2)*(x-width/2)+width*width/4)/(width/2);
11    ellipse(x,y,20,20);
12  }
```

```
 1  void setup(){
```

```
  size(400,400);
}

void draw(){
  background(255);
  fill(0,128,255);
  float t = (frameCount % 360)/360.0;
  float y = (1-t)*0 + t * height; // t * height
  float x = (width/3)*sin(4*PI*t)+width/2;
  ellipse(x,y,20,20);
}
```

```
float x0,y0;
float x1,y1;

void setup(){
  size(400,200);

  y0 = y1 = height/2;
  x0 = 0.1 * width;
  x1 = 0.9 * width;
}

void draw(){
  background(255);
  fill(0,128,255);
  float t = t = (frameCount % 360)/360.0;

  float x = (1-t) * x0 + t * x1;
  float y = (1-t) * y0 + t * y1;

  fill(255,128,0);
  ellipse(x0,y0,20,20);
  ellipse(x1,y1,20,20);
  fill(0,128,255);
  ellipse(x,y,20,20);
}
```

```
float x0,y0;
float x1,y1;

void setup(){
  size(400,200);

  y0 = y1 = height/2;
  x0 = 0.1 * width;
  x1 = 0.9 * width;
}

void draw(){
  background(255);
  fill(0,128,255);
  float t = t = (frameCount % 360)/360.0;

  float x = lerp(x0,x1,t);//(1-t) * x0 + t * x1;
```

```
18    float y = lerp(y0,y1,t);//(1-t) * y0 + t * y1;
19
20    fill(255,128,0);
21    ellipse(x0,y0,20,20);
22    ellipse(x1,y1,20,20);
23    fill(0,128,255);
24    ellipse(x,y,20,20);
25  }
```

```
1   float xPos,yPos;
2
3   void setup(){
4     size(400,400);
5     xPos = width/2;
6     yPos = height/2;
7   }
8
9   void draw(){
10    background(255);
11    fill(0,128,255);
12
13    xPos = (1-0.1) * xPos + 0.1 * mouseX;
14    yPos = (1-0.1) * yPos + 0.1 * mouseY;
15    ellipse(xPos,yPos,20,20);
16  }
```

```
1   float vx,vy;
2   float angle;
3   float xPos,yPos;
4
5   void setup(){
6     size(400,400);
7     xPos = width/2;
8     yPos = height/2;
9     vx = vy = 0;
10  }
11
12  void draw(){
13    background(255);
14    fill(0,128,255);
15
16    xPos = xPos + vx;
17    yPos = yPos + vy;
18
19    ellipse(xPos,yPos,20,20);
20  }
21
22  void mouseClicked(){
23    float dx = mouseX - xPos;
24    float dy = mouseY - yPos;
25    angle = atan2(dy,dx);
26    vx = cos(angle);
27    vy = sin(angle);
28  }
```

```
float xPos;
float yPos;
float vx,vy;

void setup(){
  size(400,400);
  xPos = 0.9*width;
  yPos = height/2;
}

void draw(){
  background(255);
  fill(0,128,255);

  float angle = 2*PI*(frameCount % 360)/360;
  vx =-2*sin(angle);
  vy = 2*cos(angle);

  xPos += vx;
  yPos += vy;

  ellipse(xPos,yPos,20,20);
}
```

```
float ax,ay;
float vx,vy;
float xPos,yPos;

void setup(){
  size(400,400);
  ax = 0;
  ay = 0.04;
  vx = 2;
  vy = 0;
  xPos = 0;
  yPos = 0;
}

void draw(){
  background(255);
  fill(0,128,255);

  vx += ax;
  vy += ay;
  xPos += vx;
  yPos += vy;

  ellipse(xPos,yPos,20,20);
}
```

```
PVector ball = new PVector();
PVector v;
float speed;
```

```
4   float angle;
5
6   void setup(){
7     size(400,400);
8     ball.x = width/2;
9     ball.y = height/2;
10    speed = 2;
11    angle = PI/4;
12
13    v = new PVector(cos(angle),sin(angle));
14    v.mult(speed);
15  }
16
17  void draw(){
18    background(255);
19    fill(0,128,255);
20
21    ball.add(v);
22    ellipse(ball.x,ball.y,20,20);
23  }
24
25  void mouseClicked(){
26    float dx = mouseX - ball.x;
27    float dy = mouseY - ball.y;
28
29    angle = atan2(dy,dx);
30    v = new PVector(cos(angle),sin(angle));
31    speed = sqrt(dx*dx+dy*dy)/100;
32    v.mult(speed);
33  }
```

```
String[] messages = {"お金があるか？",
  "がっつり食べたい",
  "がっつり食べたい",
  "大学前の中華屋さん",
  "四食",
```

```
  "一食",
  "麺類が希望",
  "三食",
  "二食"};
int state;
int startTime;

void setup() {
  size(400, 200);
  state = 0;
}

void draw() {
  background(255);
  textAlign(CENTER, CENTER);
  fill(0);

  if(millis() - startTime > 10*1000){
    state = 0;
    startTime = millis();
  }
  text(messages[state], 0, 0, width/2, height/2);
}

void keyPressed() {
  if (key == 'y' || key == 'Y') {
    if (state == 0) {
      startTime = millis();
      state = 1;
    } else if (state == 1) {
      startTime = millis();
      state = 3;
    } else if (state == 2) {
      startTime = millis();
      state = 5;
    } else if (state == 6) {
      startTime = millis();
      state = 7;
    }
  } else if (key == 'n' || key == 'N') {
    if (state == 0) {
      startTime = millis();
      state = 2;
    } else if (state == 1) {
      startTime = millis();
      state = 4;
    } else if (state == 2) {
      startTime = millis();
      state = 6;
    } else if (state == 6) {
      startTime = millis();
      state = 8;
    }
  }
}
```