2017 年度情報メディア基盤ユニット
7 月 14 日分課題【総合演習】

授業関連資料は http://www.sato-lab.jp/imfu からダウンロード出来ます。なお、問題は難易度順に並んでいるわけではありません。問 1～8 は、先生か TA の人に確認してもらってから、キャリアポートフォリオにも解答をアップして下さい。問 9 からの問題は自分が作ることのできた最後のものをキャリアポートフォリオにアップして下さい。

1. 【目コピ問題】atan2 という関数を利用すると、角度を求めることが出来ます。例えば、2 点(x_0, y_0)と(x_1, y_1)と結ぶ直線と点(x_0, y_0)を通り X 軸に平行な直線のなす角度（単位はラジアン）は$\text{atan2}(y_1 - y_0, x_1 - x_0)$で求めることが出来ます。これを利用すると、画像などを回転させて、ある特定の方向（例えば、マウスカーソルの方向）に向けるようにすることが出来ます。このことを実行したプログラムが次の未完成プログラムです。このプログラムでは、マウスカーソルの方向に、画像の先頭方向が向くようになっています。空欄を埋めて、プログラムを完成させて下さい。なお、読み込みに使っているファイルは下方向が先頭になっているものと考えます。

未完成プログラム

```
PImage ship;
void setup(){
  size(600,600);
  ship= loadImage("cannon-a.png");
}
void draw(){
  background(255);
  imageMode(CENTER);
  float dx = ___(a)___;
  float dy = ___(b)___;
  float  angle = atan2(dy,dx)+___(c)___;
  translate(width/2,height/2);
  ___(d)___;
  image(ship,0,0);
}
```

2. 【目コピ問題】このプログラムでは、ウインドウの真ん中を中心とする扇型を描いています。開始方向は X 軸の正方向で、マウスカーソルの位置が終了点となります。空欄を埋めて、プログラムを完成させて下さい。ねんのため、関数 dist(x0,y0,x1,y1)は 2 点(x0,y0)、(x1,y1)の距離を求める関数です。

未完成プログラム

```
void setup(){
  size(400,400);
}
void draw(){
  background(255);
  float xCenter = width/2;
  float yCenter = height/2;
  float daimeter = 2*dist(mouseX,mouseY,xCenter,yCenter);
  float angle = atan2(___(a)___,___(b)___);
  if(angle < ___(c)___){
```

```
    angle = angle + 2*PI;
  }
  fill(0);
  stroke(0);
  arc(xCenter,yCenter,daimeter,daimeter,  (d)  ,  (e)  );
}
```

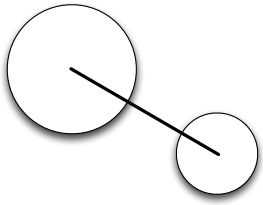
3. 【目コピ問題】授業でも説明したように、物体の移動をさせる方法の一つに、速度を変化させて、速度を利用して物体の位置を更新していくというものがあります。マウスをクリックしたら、マウスの方向に向かって、円盤が飛び出すようなプログラムを完成させて下さい。重力の効果を入れているので、まっすぐには進みません。なお、このサンプルでは、適当に定数を決めています。

未完成のプログラム

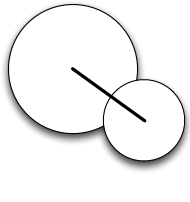
```
float vx;
float vy;
float xBall;
float yBall;
float speed;
float gravity=0.1;
float xStart; // ボールが飛び出す最初の座標
float yStart;

int state;

void setup(){
  size(600,300);
  xStart = 0.1 * width;
  yStart = 0.5 * height;
  xBall = xStart;
  yBall = yStart;
  speed = 5;
  vx = 0;
  vy = 0;
  state = 0;
}
void draw(){
  background(255);
  if(state == 0){
    fill(0);
    ellipse(xBall,yBall,10,10);
  }else{
    xBall += vx;
    yBall += vy;
    vy += gravity;
    fill(0);
    ellipse(xBall,yBall,10,10);
  }
}
void mouseClicked(){
  xBall = xStart;
  yBall = yStart;
  float dx =  (a)  - xStart;
  float dy =  (b)  - yStart;
  float angle = atan2(dy,dx);
```

```
  vx = speed * cos(angle);
  vy = speed * sin(angle);
  state = __(c)__;
}
```

4. 【目コピ問題】２つの円が衝突しているかどうかは、２円の中心の距離とこの２つの円の半径の和を比べることで判定ができます（右図参照)。これを利用して、２つの円が衝突しているときと、そうでないときで円の色を変えるプログラムとなっています。空欄を埋めて、プログラムを完成させて下さい。



中心間の距離が半径の和よりも大きければぶつかっていない



中心間の距離が半径の和よりも小さければぶつかっている

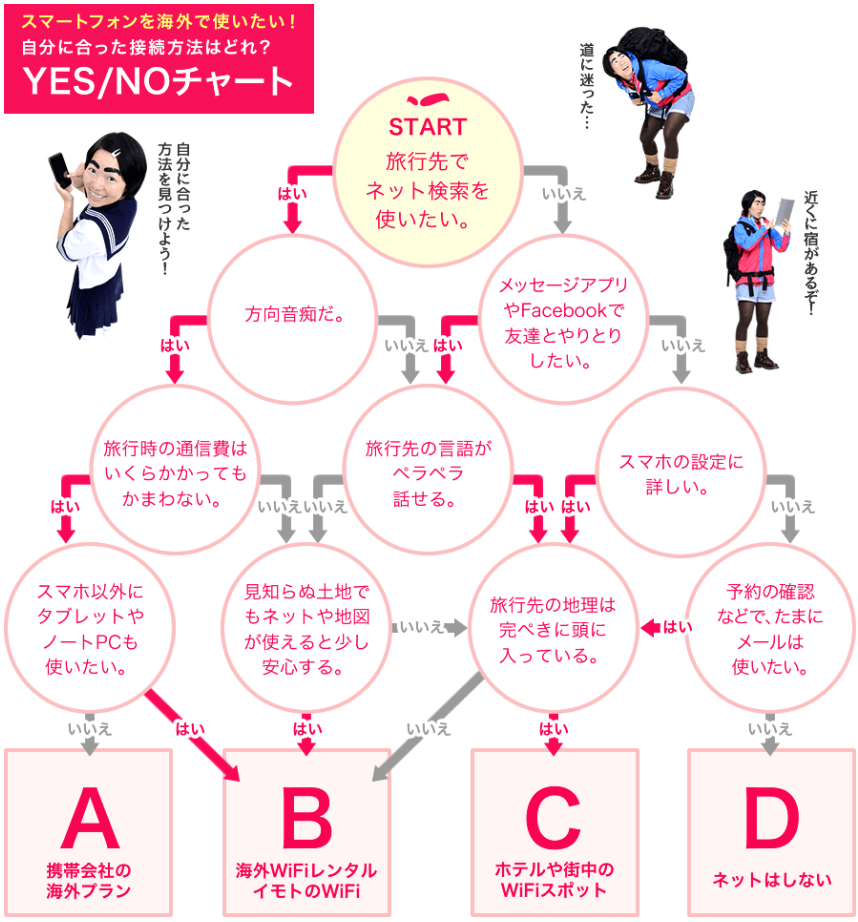
未完成プログラム

```
float xCenter;
float yCenter;
__(a)__r0;
__(a)__r1;
void setup() {
  size(600, 600);
  xCenter = width/2;
  yCenter = height/2;
  r0 = 0.25*width;
  r1 = 20;
}
void draw() {
  background(255);
  if (__(b)__(mouseX, mouseY, xCenter, yCenter) < __(c)__) {
    fill(255, 10, 10);
  } else {
    fill(10, 255, 10);
  }
  ellipse(xCenter, yCenter, 2*r0, 2*r0);
  ellipse(mouseX, mouseY, 2*r1, 2*r1);
}
```

5. 【工夫問題】次のページにある画像のような Yes/No チャートを Processing のプログラム上で出来るようにして下さい。図は http://www.globaldata.jp/kijiworld/no1410c/のものを利用したのですが、見つからなくなってしまいました。

6. 【目コピ問題】問１のプログラムに変更を加えて、マウスの方向を向きながら、マウス位置の方向に移動してようにして下さい。

7. 【工夫問題】問６のプログラムに変更を加えて、マウスをクリックしたら弾を発射するようにして下さい。弾の画像には shot3.png を利用して下さい。



8. 【目コピ問題】以前の演習問題で、Y 軸に平行な線分と円の衝突（接触）判定を扱いました。プログラムを作る中では、両端点を指定し、その 2 点を結ぶ線分と円の衝突判定を行う方法が必要となります。これを行う方法の一つとして、以下のようなものが考えられます。説明の際には、両端点をP_0, P_1とし、円の中心をQとし、円の半径をrとします。以下のような場合には、線分と円が衝突しています。

   ① P_0とQの距離とP_1とQの距離がともにrより小さい。

   ② 直線P_0P_1と点Qとの距離がrよりも小さく、さらに$\angle P_0P_1Q$と$\angle P_1P_0Q$がともに鋭角となっている。

$\angle P_0P_1Q$が鋭角かどうかは、ベクトルの内積$\overrightarrow{P_1P_0} \cdot \overrightarrow{P_1Q}$の値の正負で判定することができます。内積の値が正であれば鋭角、負の値であれば鈍角となっています。この方針で線分と円が衝突しているかどうかを判定するプログラムが、つぎのサンプルです。以下の文章の空欄に適切な関数名や変数などを入れて下さい。

このサンプルプログラムでは、__(a)__関数によって、線分と円の衝突判定を行っている。この関数では、中心の座標（__(b)__, __(c)__）で半径が__(d)__の円と両端点の座標が(__(e)__, __(f)__)と(__(g)__, __(h)__)の線分の接触判定を行っている。この関数の戻り値は接触しているときには__(i)__、そうでないときには__(j)__となっている。この関数は、このサンプル内で定義している二つの関数を__(k)__と__(l)__を利用している。

サンプルプログラム

```
float vx;
float vy;
float xBall;
float yBall;

float dotProduct(float x1, float y1, float x2, float y2) {
  return x1*x2 + y1*y2;
}

float distance(float x0, float y0,
  float x1, float y1,
  float x2, float y2) {
  float vx1 = x0 - x1;
  float vy1 = y0 - y1;
  float vx2 = x2 - x1;
  float vy2 = y2 - y1;
  float c = dotProduct(vx1, vy1, vx2, vy2);
  return sqrt((vx1*vx1 + vy1*vy1) - (c*c/(vx2*vx2+vy2*vy2)));
}

boolean  isINCollionWith(float xCenter, float yCenter, float radius,
  float x1, float y1, float x2, float y2) {
  if (dist(xCenter, yCenter, x1, y1) <= radius) {
    return true;
  }
  if (dist(xCenter, yCenter, x2, y2) <= radius) {
    return true;
  }
  if (distance(xCenter, yCenter, x1, y1, x2, y2) > radius) {
    return false;
  }
  float vx0 = x2 - x1;
  float vy0 = y2 - y1;
  float vx1 = xCenter - x1;
  float vy1 = yCenter - y1;
  float vx2 = xCenter - x2;
  float vy2 = yCenter - y2;
  if (dotProduct(vx0, vy0, vx1, vy1) < 0) {
    return false;
  }
  if (dotProduct(-vx0, -vy0, vx2, vy2) < 0) {
    return false;
  }
  return true;
}

void setup() {
  size(600, 600);
  xBall = width/2;
  yBall = 0.9 * height;
  vx = 0;
  vy = -1;
}

void draw() {
  background(255);
  float t = 2*PI*frameCount/360.0;
```

```
  float radius = width/4;
  float angle = PI/2 + (PI/2-PI/12)*sin(t);
  float x0 = width/2 + radius * cos(angle);
  float y0 = height/2 +radius * sin(angle);
  float x1 = width/2 + radius * cos(angle+PI);
  float y1 = height/2 + radius * sin(angle+PI);
  line(x0, y0, x1, y1);

  xBall += vx;
  yBall += vy;
  fill(0);
  if (isINCollionWith(xBall, yBall, 10, x0, y0, x1, y1)) {
    fill(255, 10, 10);
  }
  ellipse(xBall, yBall, 20, 20);
}
```

授業中に配布したプリントのゲーム（4〜6 ページに出ている早撃ちゲーム）に以下のような改良を加えてものを作成して下さい。このプリントの最後に関連する部分を再度載せておきます。

9. 【工夫問題】正方形の表示時間を乱数で変更するようにして下さい。
10. 【工夫問題】hit 回数を表示するようにして下さい。
11. 【工夫問題】3 回ミス hit したら、ゲームが終了するような状態遷移図を描いて下さい。
12. 【工夫問題】(3)で描いた状態遷移図を利用して、3 回ミス hit したら、ゲームが終了するようにして下さい。
13. 【工夫問題】hit 回数に応じて、正方形の表示時間が短くなるようにして下さい。
14. 【工夫問題】単純な正方形ではなく、画像や別な形となるようにして下さい。
15. 【工夫問題】繰り返しゲームが出来るようにし、開始画面にそれまでの最大 hit 回数が表示されるようにして下さい。
16. 【工夫問題】効果音を付け加えて下さい。

宿題、事前学習はありません、最終課題の制作に力を入れて下さい。