



---

# プログラミング演習 (5)

## 条件分岐と当たり判定(2)

---

中村, 小松  
小林, 橋本

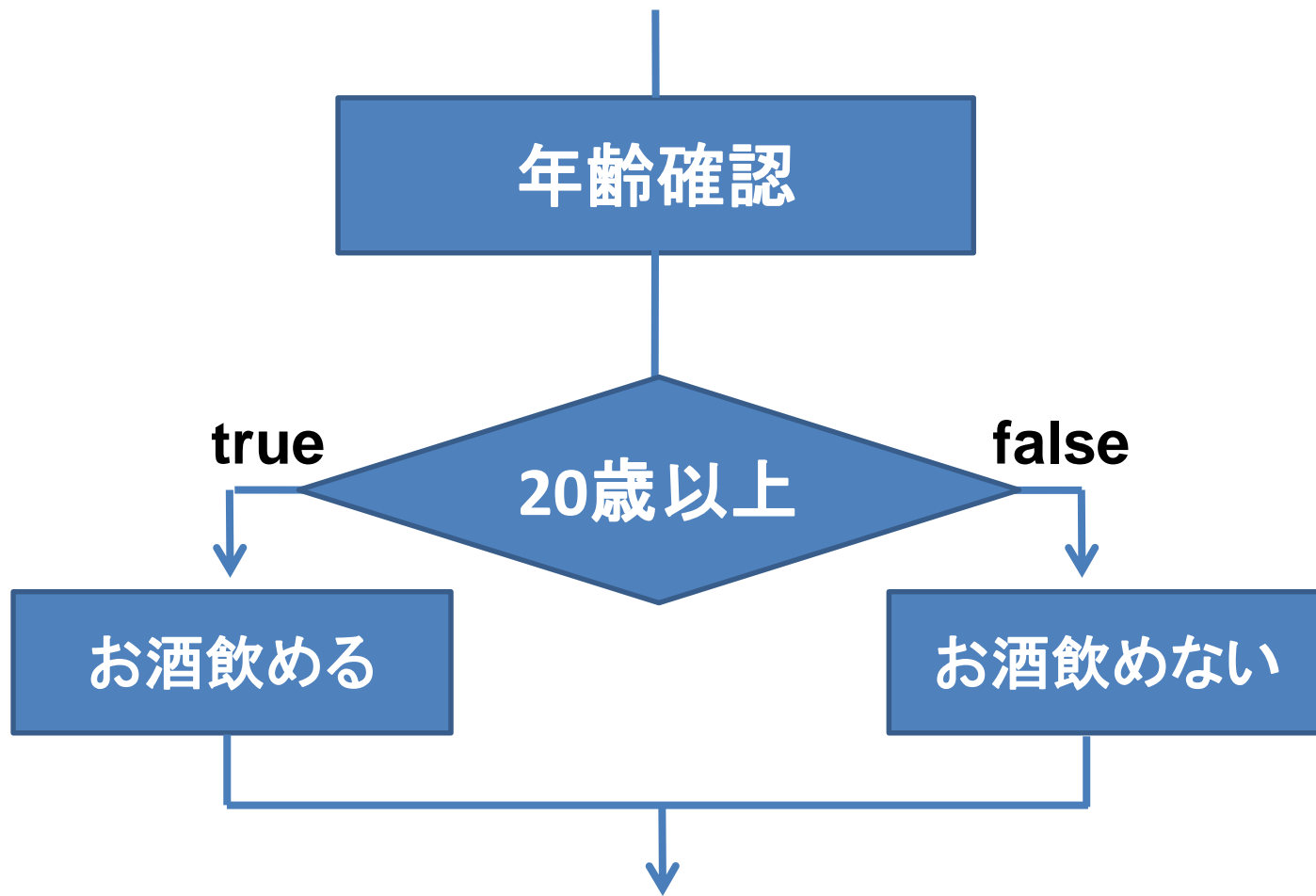


- Processing で当たり判定に挑戦！
  - 条件分岐を理解する
  - 何らかの条件を満たした時に色を変える！
  - マウスカーソルと動いている円がぶつかったら終了
  - シューティングゲームやもぐらたたきに挑戦！
  
- 課題：
  - Processing でゲームを作ろう！
  - 占いを作ってみよう

# フローチャートと条件分岐



- プログラムの流れ





```
if( 条件A ){  
    // 条件Aの時の処理内容  
} else if( 条件B ){  
    // 条件Aでなく, 条件Bの時の処理内容  
} else {  
    // 条件AおよびB以外の時の処理内容  
}
```

よくあるミス

```
if( 条件 );{ 条件A の処理 }
```

# 論理値, 真偽値 (boolean)



- true か false かを値として持つ
  - それ以外の値は持たない
- 制御文で「条件Aを満たす時」というのは「条件Aが真 (true) である」と同じ意味
- 制御文で「条件Aを満たさない時」というのは「条件Aが偽 (false) である」と同じ意味
  - $x > y$  の条件をみたす場合,  $x > y$  は true, みたさない場合は false
  - $x == y$  は  $x$  と  $y$  が同じ値の場合 true に, 違う値の場合に false となる

# 条件の記述方法



演算子	意味	プログラム上
$x > y$	x が y より大きい	左記の時に true それ以外で false
$x < y$	x が y より小さい	同上
$x \geq y$	x が y 以上	同上
$x \leq y$	x が y 以下	同上
$x == y$	x と y が等しい	同上
$x != y$	x と y が等しくない	同上
$!x$	x は false	同上

# ～かつ～はどうするのか？



- 論理積演算子と論理和演算子
- ～かつ～のとき                    &&
- ～または～のとき                ||

```
if ( x > 200 && x < 400 ) {
```

x が200より大きく, 400より小さい時はここに来る

```
}
```

```
if ( n < -10 || n > 10 ) {
```

nが-10より小さいか, 10より大きければここに来る

```
}
```

if( 200 < ~~x~~ < 400 )

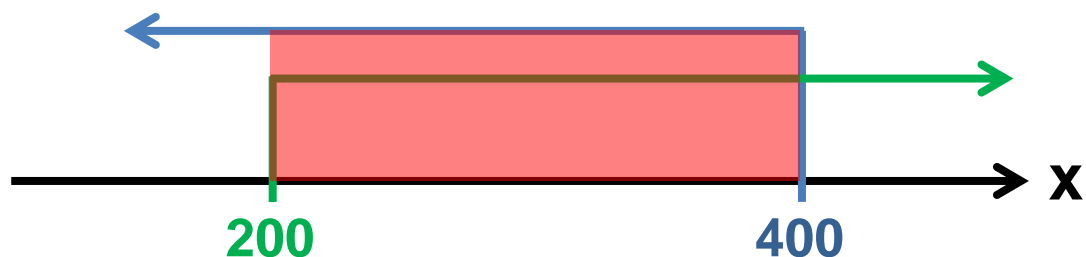
# ～かつ～はどうするのか？



```
if ( x > 200 && x < 400 ) {
```

x が200より大きく, 400より小さい時はここに来る

```
}
```



詳しく説明すると...

- x = 100 のときは, 「x > 200 が false」で「x < 400 が true」となるため, 「false && true」となり, 「false」となる
- x = 300 のときは, 「x > 200 が true」で「x < 400 が true」となるため, 「true && true」となり, 「true」となる
- x = 500 のときは, 「x > 200 が true」で「x < 400 が false」となるため, 「true && false」となり, 「false」となる



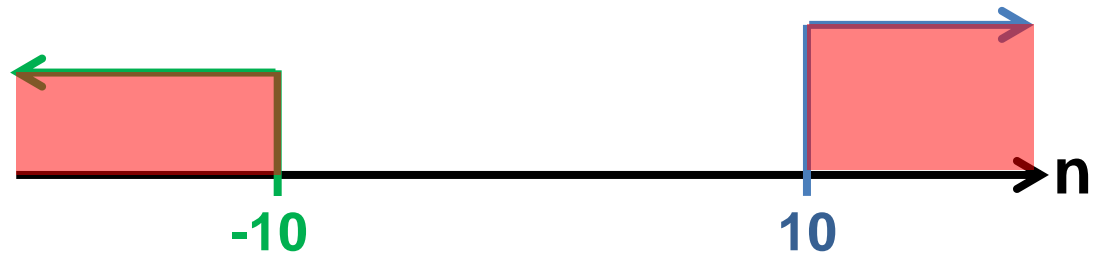
# ～かつ～はどうするのか？



```
if ( n < -10 || n > 10 ) {
```

nが-10より小さいか、10より大きければここに来る

```
}
```



詳しく説明すると...

- $n = -20$  のときは、「 $n < -10$ がtrue」で「 $n > 10$ がfalse」となるため、「true || false」となり、「true」となる
- $n = 0$  のときは、「 $n < -10$ がfalse」で「 $n > 10$ がfalse」となるため、「false || false」となり、「false」となる
- $n = 20$  のときは、「 $n < -10$ がfalse」で「 $n > 10$ がtrue」となるため、「false || true」となり、「true」となる

# 色々つまづくポイント

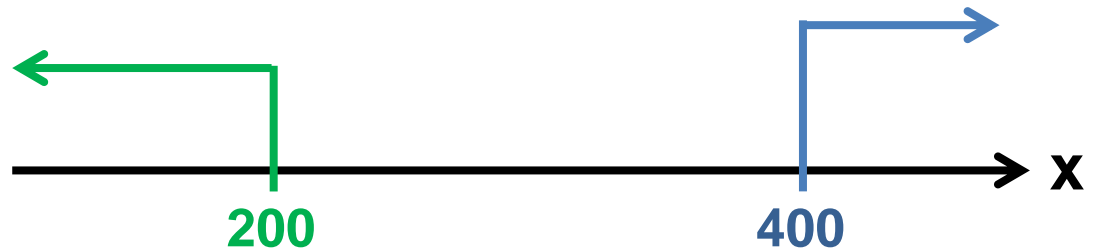


true に何時まで経ってもならない例

```
if( x < 200 && x > 400 ){
```

```
    // だめ
```

```
}
```



ずっと true になってしまう例

```
if( n > -10 || n < 10 ){
```

```
    // だめ
```

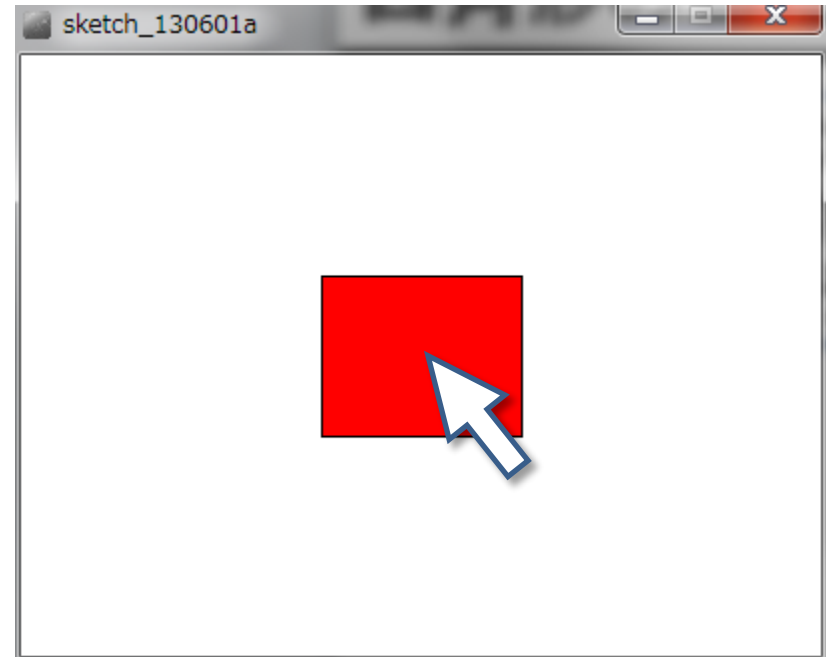
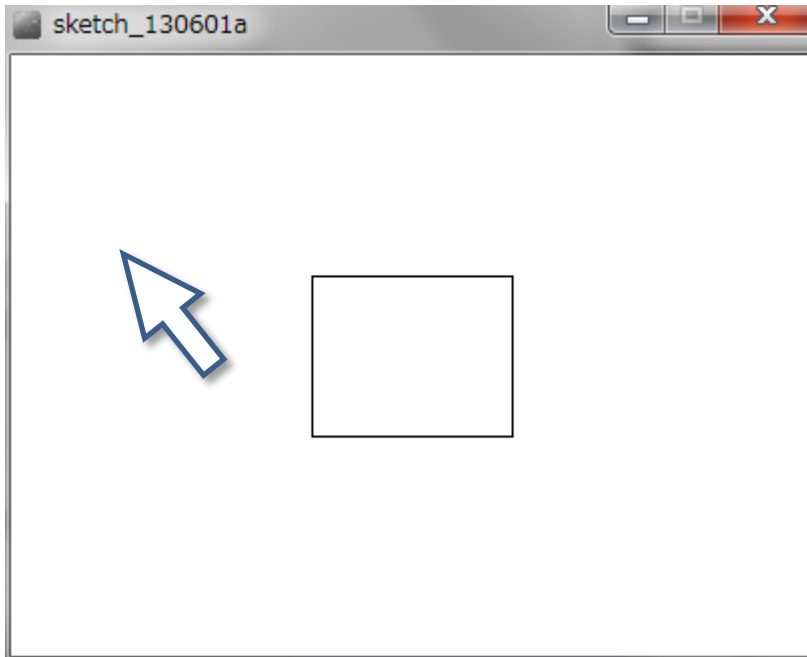
```
}
```



# 四角形の当たり判定



(Q) 400x300のウィンドウの中央に表示された横100, 縦80の長方形の中にカーソルがあると長方形を赤色に, そうでなければ白色にするには?

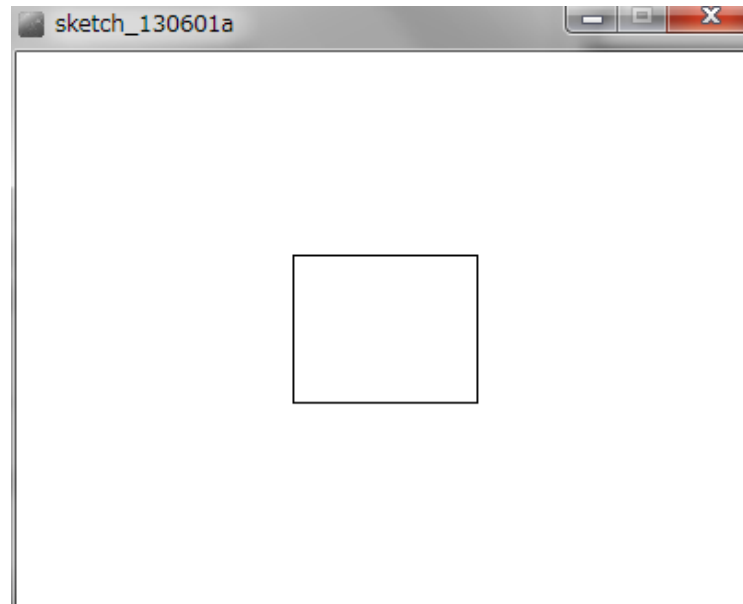


# 四角形の当たり判定



- 考え方

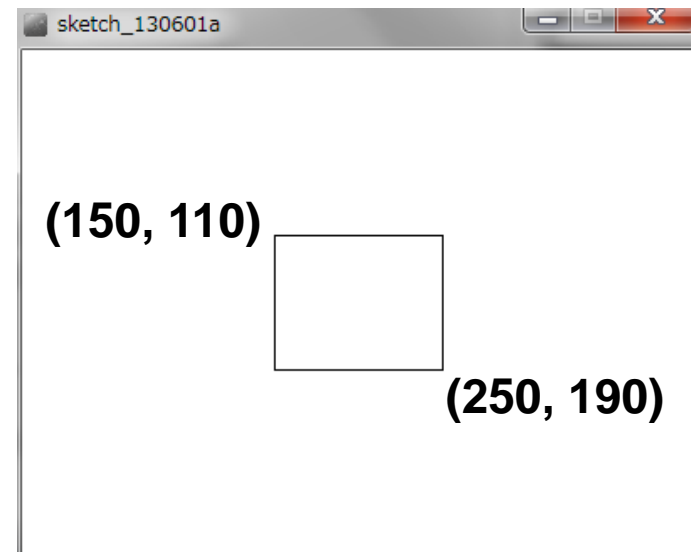
- 画面の中央は (200, 150)
- 長方形の左上と右下の座標は??
- 長方形は `rect( 左上x, 左上y, 横幅, 縦幅 );` で描画
- マウスカーソルの座標は (mouseX, mouseY)



# 四角形の当たり判定



- 中央(200, 150)で, 横幅100, 縦幅80なので
  - 左上の座標は $(200-100/2, 150-80/2) = (150, 110)$
  - 右下の座標は $(200+100/2, 150+80/2) = (250, 190)$
- $150 \leq \text{mouseX} \leq 250$  かつ  $110 \leq \text{mouseY} \leq 190$ なら赤色, そうでなければ白色で塗りつぶす
  - つまり,  $\text{mouseX} \geq 150$  かつ  $\text{mouseX} \leq 250$  かつ  $\text{mouseY} \geq 110$  かつ  $\text{mouseY} \leq 190$  の時!
- 「かつ」は, 「&&」で表現する



# 四角形の当たり判定



(注意)  $150 \leq \text{mouseX} \leq 250$  はダメ!

$150 \leq \text{mouseX} \ \&\& \ \text{mouseX} \leq 250$  に分解

```
void setup(){
  size( 400, 300 );
}

void draw(){
  background( 255, 255, 255 );
  if( mouseX >= 150 && mouseX <= 250 && mouseY >= 100 && mouseY <= 190 ){
    fill( 255, 0, 0 );
  } else {
    fill( 255, 255, 255 );
  }
  rect( 150, 110, 100, 80 );
}
```

# 四角形の当たり判定



if や else if, else の {} の中に if を入れてもOK!!  
(多段階の条件分岐)

```
void draw(){  
    background( 255, 255, 255 );  
    if( mouseX >= 150 && mouseX <= 250 ){  
        if( mouseY >= 100 && mouseY <= 190 ){  
            fill( 255, 0, 0 );  
        } else {  
            fill( 255, 255, 255 );  
        }  
    } else {  
        fill( 255, 255, 255 );  
    }  
    rect( 150, 110, 100, 80 );  
}
```

さらに $110 \leq \text{mouseY} \leq 190$   
ならここに入ってくる

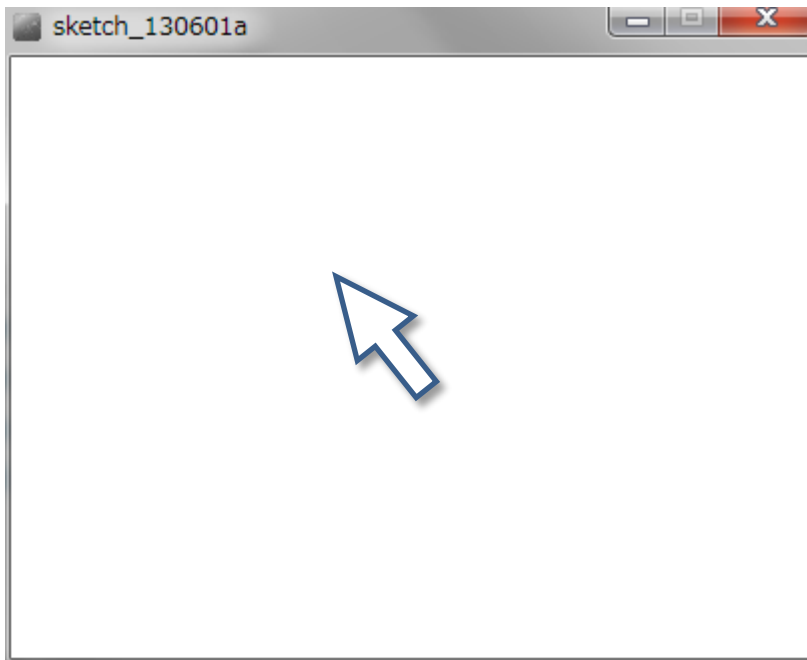
$150 \leq \text{mouseX} \leq 250$ なら  
ここに入ってくる

さらに $110 \leq \text{mouseY} \leq 190$   
でない場合はここに入ってくる

# 四角形の当たり判定(2)



(Q) ランダムな場所に、縦幅30x横幅20の大きさに配置された四角形を探す(カーソルがその上にあるときだけ表示する)プログラムはどう作るか？





# 四角形の当たり判定(2)



## • 考え方

– 四角形の左上の座標を実数の変数に

- `float leftTopX;` // 他の変数名でもOK

- `float leftTopY;` // `hidariueX`, `hidariueY` でもOK

– 四角形の左上の座標をランダムに決める

- `random( 最小値, 最大値 );` で, 最小値から最大値までの値を求めることができる

- `leftTopX = random( 50, 350 );` // 50~350のランダムな値などのように...

# 四角形の当たり判定(2)



- 考え方(続き)

- mouseX がどういった条件を満たせば表示する？

- $\text{leftTopX} \leq \text{mouseX} \leq \text{leftTopX} + 30$

- mouseY がどういった条件を満たせば表示する？

- $\text{leftTopY} \leq \text{mouseY} \leq \text{leftTopY} + 20$

- 2つの条件を満たした時に長方形を表示する！

- 長方形は `rect( 左上x, 左上y, 横幅, 縦幅 );` で描画

# 四角形の当たり判定(2)



```
float leftTopX;  
float leftTopY;
```

```
void setup(){  
    size( 400, 300 );  
    leftTopX = random(0, 390); // 0~390までの任意の値  
    leftTopY = random(0, 290); // 0~290までの任意の値  
}
```

400までにしちゃうとはみ出て見つからない

```
void draw(){  
    background(255,255,255);  
    if( mouseX >= leftTopX && mouseX <= leftTopX+30 ){  
        if( mouseY >= leftTopY && mouseY <= leftTopY+20 ){  
            rect( leftTopX, leftTopY, 30, 20 );  
        }  
    }  
}
```

# 四角形の当たり判定(2)



```
float leftTopX;  
float leftTopY;  
  
void setup(){  
    size( 400, 300 );  
    leftTopX = random(width-10);  
    leftTopY = random(height-10);  
}
```

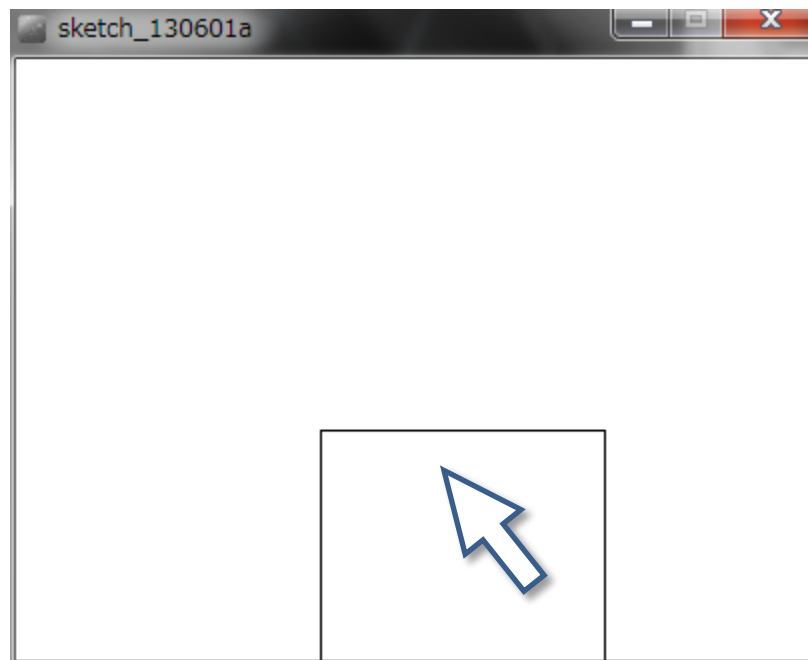
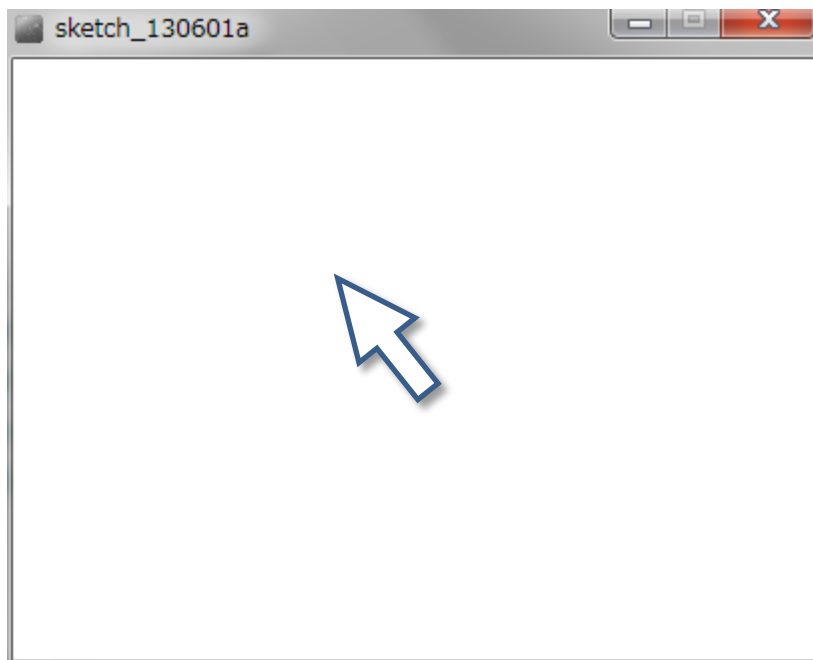
```
void draw(){  
    background(255,255,255);  
    if( mouseX >= leftTopX && mouseX <= leftTopX+30  
        && mouseY >= leftTopY && mouseY <= leftTopY+20 ){  
        rect( leftTopX, leftTopY, 30, 20 );  
    }  
}
```

もちろん全部&&でつないでもOK

2行で書いてもOK!



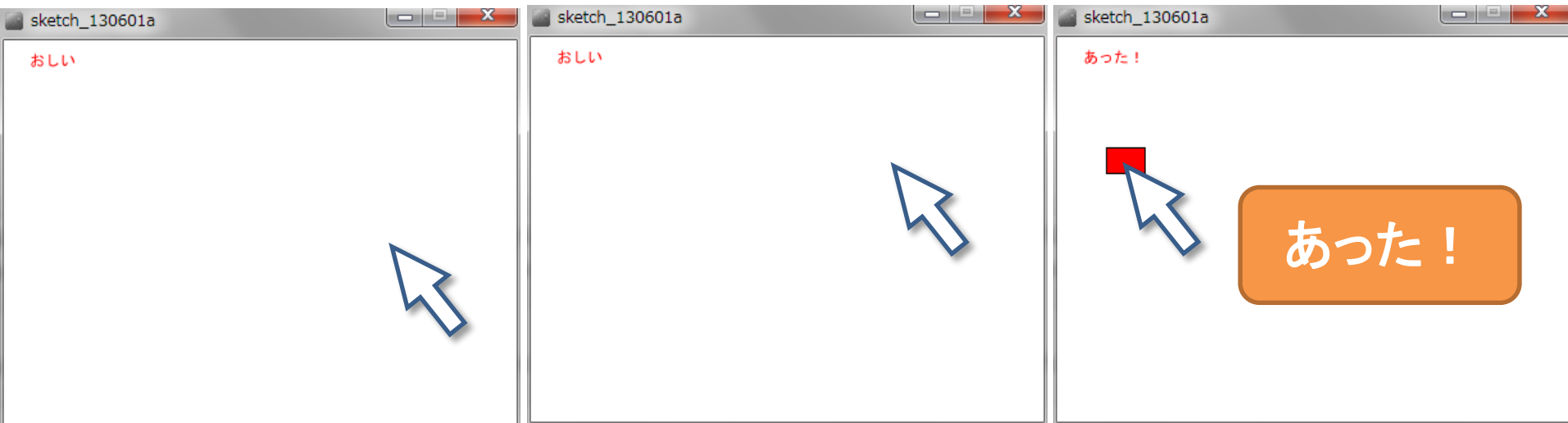
- ランダムな場所に、ランダムな大きさに配置された四角形を探す(カーソルがその上にあるときだけ表示する)プログラムを作成しよう！
  - 四角形の左上の座標, 縦幅横幅を実数の変数に (leftTopX, leftTopY, rectWidth, rectHeightなど)



# 四角形の当たり判定(3)



(Q) ランダムな場所に、縦幅30x横幅20の大きさに配置された四角形を探すプログラムでx座標またはy座標があつたらヒントを出そう！



# 四角形の当たり判定(3)



- カーソルのX座標のみ, Y座標のみを見た時に, 四角形の中に含まれていればおいしいと表示する



マウスカーソルがオレンジのエリアにある時  
おいしいと表示する！

- つまり,  $\text{leftTopX} \leq \text{mouseX} \leq \text{leftTopX} + 30$  または,  $\text{leftTopY} \leq \text{mouseY} \leq \text{leftTopY} + 20$  のときに表示！

# 四角形の当たり判定(3)



- 「AかつB」または「CかつD」はどう表現？
  - 「かつ」は「&&」
  - 「または」は「||」
  - まず、「AかつB」と「CかつD」を判定してしまっても、その後「AかつB」または「CかつD」を判定する
  - 先に判定する場合は「()」で囲む(算数と一緒に！)

```
if( (A && B) || (C && D) ){ ... }
```



# 四角形の当たり判定(3)



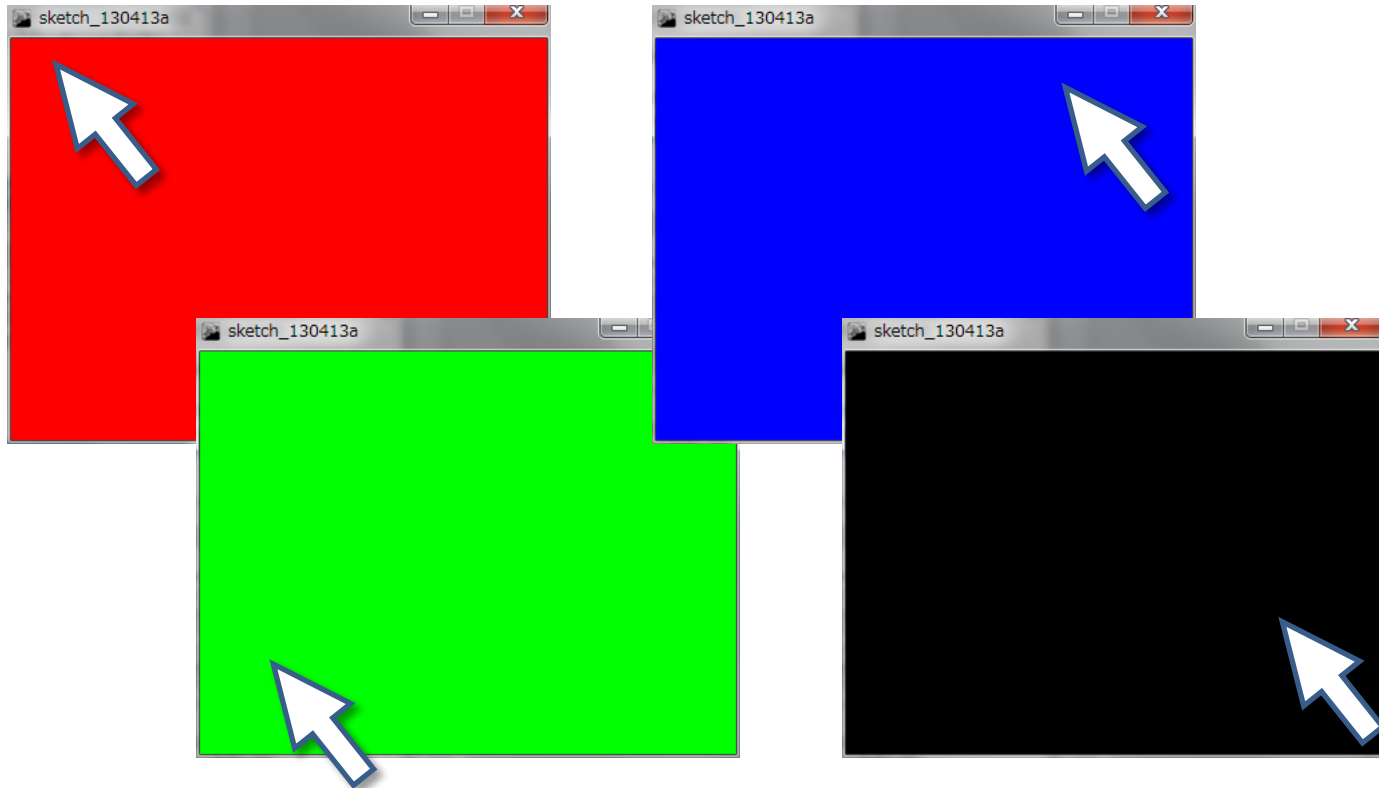
```
float leftTopX, leftTopY;
void setup(){
    size( 400, 300 );
    leftTopX = random(10, 390);
    leftTopY = random(10, 290);
    fill( 255, 0, 0 ); // 文字と四角形の塗り色は赤色にする
}
void draw(){
    background(255,255,255); // 背景は白色
    if( mouseX >= leftTopX && mouseX <= leftTopX+30
        && mouseY >= leftTopY && mouseY <= leftTopY+20 ){
        rect( leftTopX, leftTopY, 30, 20 );
        text( "あった！", 20, 20 ); // (20,20)に「あった！」と表示！
    } else {
        if( (mouseX >= leftTopX && mouseX <= leftTopX+30)
            || (mouseY >= leftTopY && mouseY <= leftTopY+20) ){
            text( "おいしい", 20, 20 ); // (20,20)に「おいしい」と表示！
        }
    }
}
```

**$\text{leftTopX} \leq \text{mouseX} \leq \text{leftTopX} + 30$  または、  
 $\text{leftTopY} \leq \text{mouseY} \leq \text{leftTopY} + 20$**

# 色々な条件に挑戦



(Q) 400x300のウィンドウでマウスが画面の左上で赤背景, 右上で青背景, 左下で緑背景, 右下で黒背景にするには？



# 色々な条件に挑戦



- 条件(「<」にするか「<=」にするかは人次第)
  - 左上は  $\text{mouseX} < 200$  かつ  $\text{mouseY} < 150$  → 赤色
  - 右上は  $\text{mouseX} \geq 200$  かつ  $\text{mouseY} < 150$  → 青色
  - 左下は  $\text{mouseX} < 200$  かつ  $\text{mouseY} \geq 150$  → 緑色
  - 右下は  $\text{mouseX} \geq 200$  かつ  $\text{mouseY} \geq 150$  → 黒色

```
void draw(){
    if( mouseX < 200 && mouseY < 150){
        background(255,0,0);
    } else if( mouseX >= 200 && mouseY < 150 ){
        background(0,0,255);
    } else if( mouseX < 200 && mouseY >= 150 ){
        background(0,255,0);
    } else {
        background(0,0,0);
    }
}
```



- if の処理内容が1つだけのときは {} を省略可能

```
if( mouseX < 200 && mouseY < 150){  
    background(255,0,0);  
}
```



```
if( mouseX < 200 && mouseY < 150)  
    background(255,0,0);
```

**ただ、慣れるまでは省略しない！**

# 値で沢山分岐する場合は？



- キーボードからの入力は「void keyPressed()」で取得，入力キーは変数の「key」を調べるだけ！
  - 沢山 if – else if – else if – else if – else if - ... と繋げて  
も良いが，見通しがやや悪くなる

```
void keyPressed(){  
    if( key == 'a' ){  
        println( "Aが押されました" );  
    } else if( key == 'b' ){  
        println( "Bが押されました" );  
    } else if( key == 'c' ){  
        println( "Cが押されました" );  
    } else {  
        println( "それ以外のキー" );  
    }  
}
```

```
void keyPressed(){  
    switch( key ){  
        case 'a':  
            println( "Aが押されました" );  
            break;  
        case 'b':  
            println( "Bが押されました" );  
            break;  
        default:  
            println( "それ以外のキー" );  
            break;  
    }  
}
```

# switch – case – break



```
switch( 分岐させる変数 ){  
  case  値1:  
    値1の時の処理  
    break; // 値1の時の処理ここまで  
  case  値2:  
    値2の時の処理  
    break; // 値2の時の処理ここまで  
  default:  
    // 値1でも2でもない場合の処理  
    break;  
}
```

# up/down/left/right



key ではなく  
keyCode という  
変数を利用

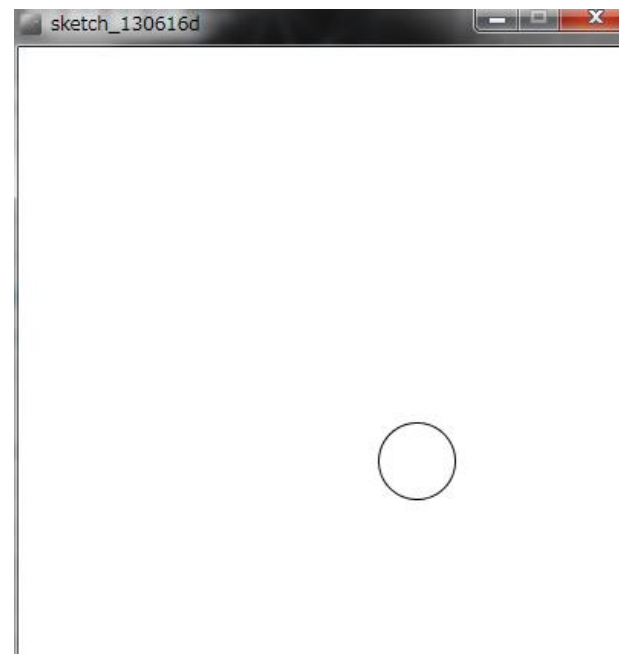
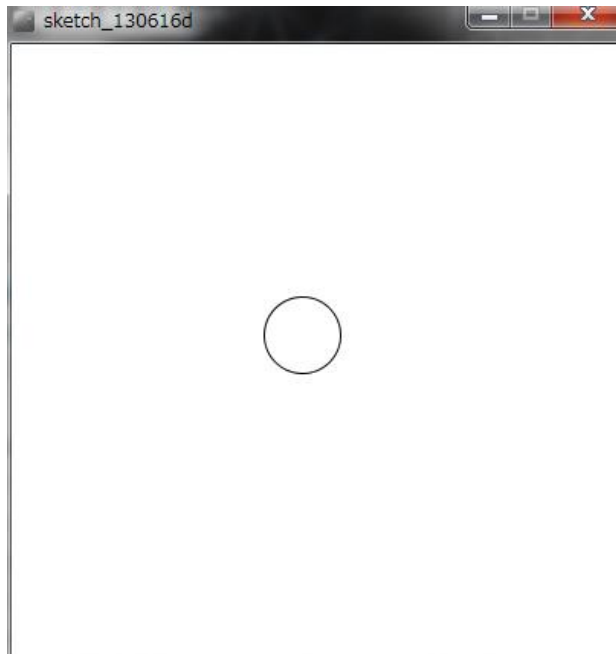
```
void keyPressed() {  
    switch( keyCode ) {  
        case UP:  
            println( "upが押されました" );  
            break;  
        case DOWN:  
            println( "downが押されました" );  
            break;  
        case LEFT:  
            println( "leftが押されました" );  
            break;  
        case RIGHT:  
            println( "rightが押されました" );  
            break;  
        default:  
            println( "それ以外のキー" );  
            break;  
    }  
}
```

# up/down/left/rightで移動



(Q) 上下左右キーで円を動かすにはどうするか？

- keyPressed() で入力を取得
- keyCode で上下左右ボタンを取得
- 座標を up/down/right/left で変更





# up/down/left/rightで移動



```
void setup(){
    size( 400, 400 );
}

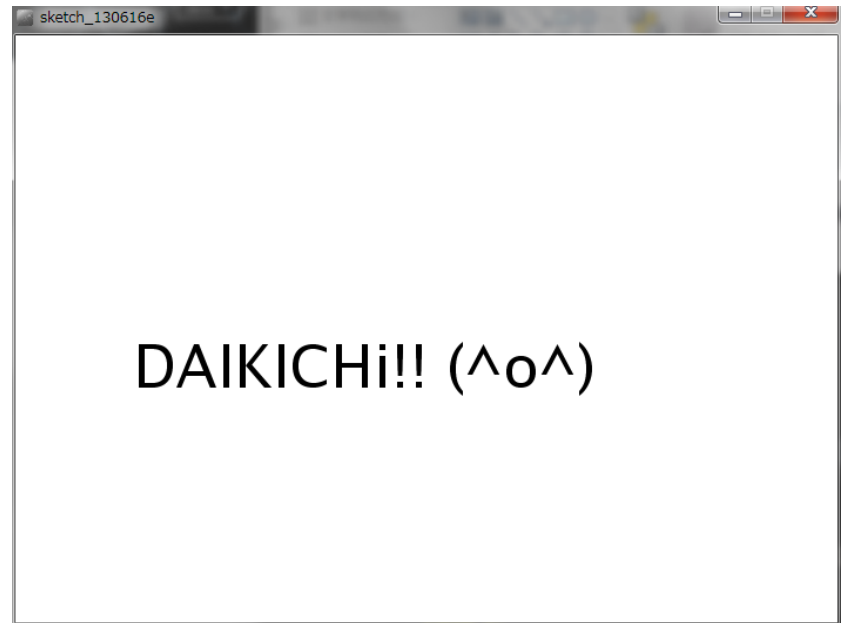
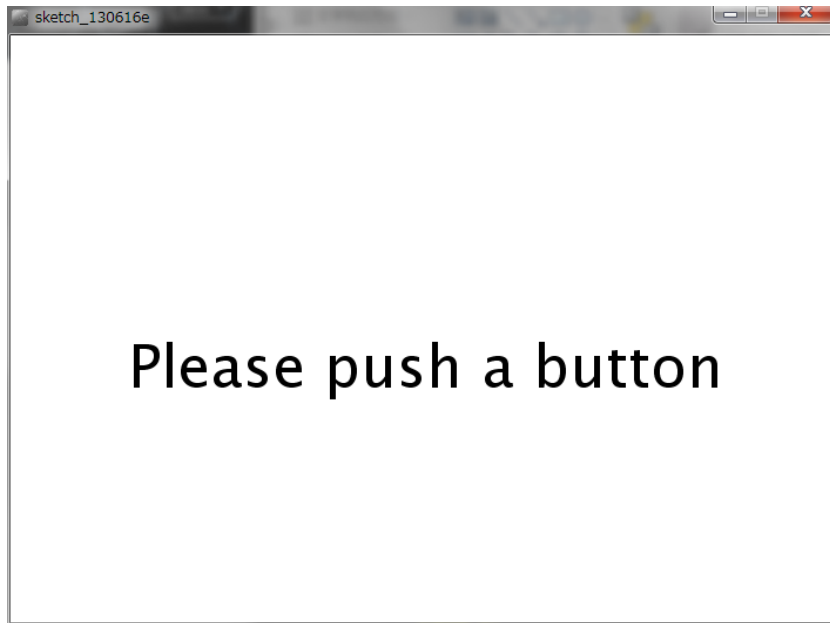
int x = 200;
int y = 200;
void draw(){
    background( 255, 255, 255 );
    ellipse( x, y, 50, 50 );
}
```

```
void keyPressed() {
    switch( keyCode ) {
        case UP:
            y = y - 10;
            break;
        case DOWN:
            y = y + 10;
            break;
        case LEFT:
            x = x - 10;
            break;
        case RIGHT:
            x = x + 10;
            break;
    }
}
```

# 占いを作ってみる



(Q) マウスクリックすると占い結果を表示するプログラムを作ってみよう！



# 占いを作ってみる



## • 考え方

- マウスクリックした際に 0~9 の乱数を発生
- 乱数を発生させるのは `random( ... );`
  - `random( 数字 );` 数字までのランダムな値を取得
  - `random( 開始, 終了 );` 開始~終了のランダムな値を取得
  - **注意点: 乱数は実数 (0.0000, 0.00001, 0.00002, ...) になるので整数と単純に比較してはダメ. 整数に変換してあげる必要がある! (実数から変数へ `(int)` で変換!)**
    - `num = (int) random( 10 );`
- 乱数の値を変数に格納
- 変数の値に応じて占い結果を表示する!

# 占いを作ってみる



```
void setup(){
  size( 700, 500 );
  fill( 0, 0, 0 );
  textSize( 50 );
}

int num = -1; // -1の場合占い無し
void mousePressed(){
  num = (int)random(0,10);
}
```

```
void draw() {
  background( 255, 255, 255 );
  switch( num ) {
  case 0:
    text( "DAIKICHi!! (^o^)", 100, 300 );
    break;
  case 1:
    text( "DAIKYO!! (X_X)", 100, 300 );
    break;
  case 2:
    text( "KICHI (-_-)", 100, 300 );
    break;
    // ～以降 3 -- 9 まで繰り返し
  default:
    text( "Please push a button", 100, 300 );
    break;
  }
}
```



- 左から右に動く白色の四角形をマウスでクリックすると赤色で塗りつぶすプログラムに挑戦
- 画面に表示されていない左から右に動く四角形を探すプログラムに挑戦
  - 惜しい場合はヒントを表示しましょう！
- 簡単なゲームを作ってみましょう
  - ただ表示されているボタンを全て押すだけのゲーム
  - 動いてくる敵を避けるゲームなど. なんでもOKです
- 画面のクリック場所によって占いの結果を変更するプログラムを作ってみましょう