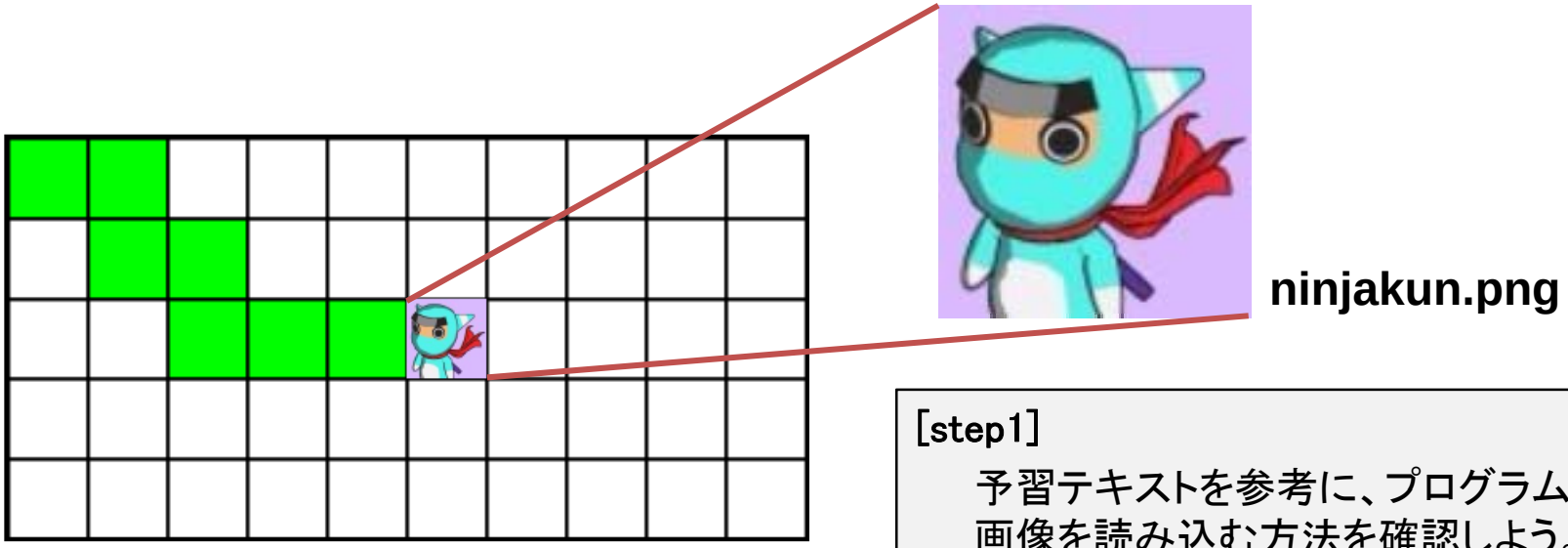
プログラミング演習I（第11回）課題

【3組】

- **基本①　スケッチ名：boardgame2**
  - 以前作ったキーボード操作でマス目を移動するプログラム（boardgame）において、円で描いていた駒の部分を好きな画像に差し替えてください。



ninjakun.png

[step1]

予習テキストを参考に、プログラムで画像を読み込む方法を確認しよう。

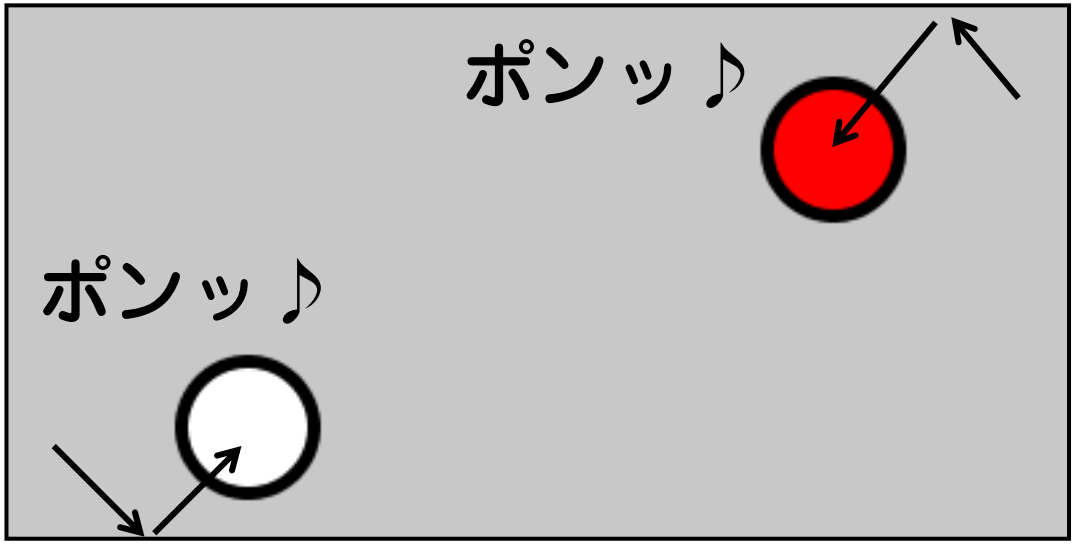
[step2]

以前ellipse()命令で円を描いていた場所を、image()命令に差し替えて画像を表示させよう。サイズも調整しよう。

プログラミング演習I（第11回）課題

【3組】

- **基本②　スケッチ名：bound2**
  - 画面内を2つの円が移動し、壁に当たると跳ね返るプログラムを作成してください。なお、壁に当った時に適当な効果音が鳴るようにしてください。



[step1]

まずは以前作ったboundというプログラムを思い出して、それをベースに2個の円を動かせるようにしよう。配列を使うとよいかも？

[step2]

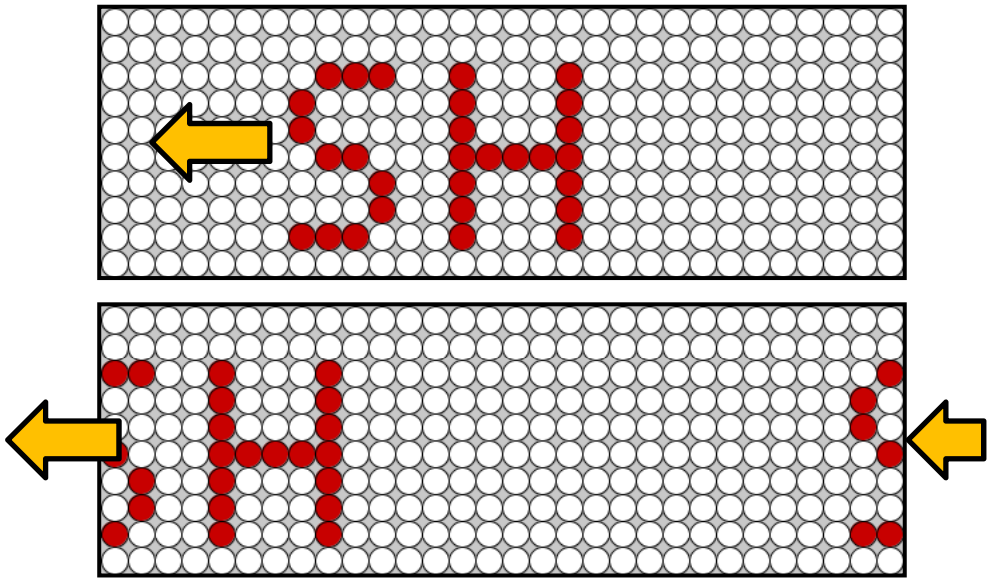
衝突したときに音が鳴る処理を加えてみよう。

2つの円で効果音は同じでOK。余裕のある人は効果音を2つ使ってみよう。

プログラミング演習I（第11回）課題

【3組】

- **基本③　スケッチ名：keijiban3**
  - 電光掲示板のプログラム（keijiban2）を改造して、左端から右端へのループと、自動スクロールを実現してください。
  - プログラム実行直後に自分のイニシャル2文字が流れるようにすること。
  - 文字が読める速度にすること。



[step1]

まずは実行直後に文字が表示されるようにしよう。setup()の中で配列にあらかじめデータを入れておけばOK。

[step2]

左に文字が流れる処理の部分を改良して、ループするようにしよう。両端をどう処理するかがカギだ。

[step3]

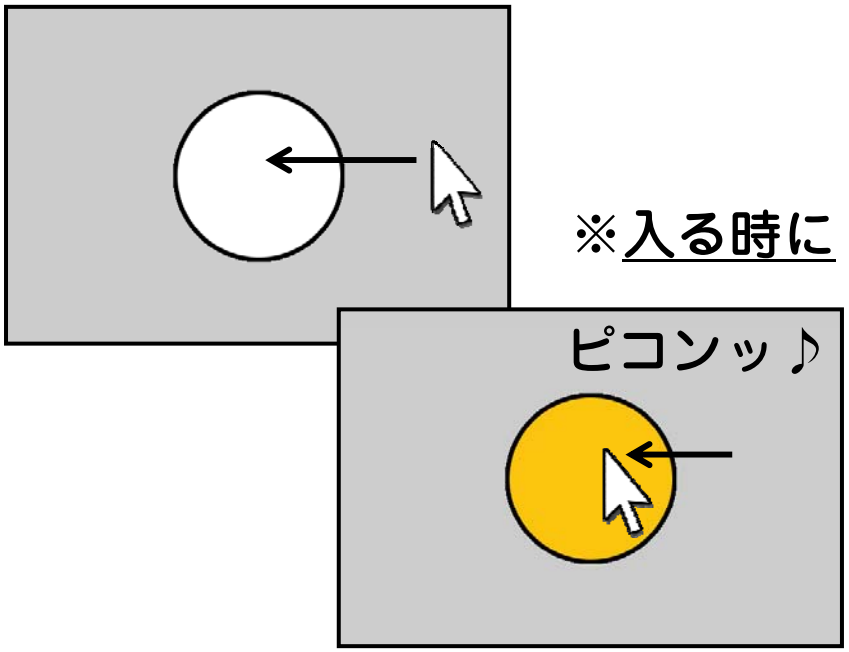
最後に自動ループさせよう。流れる速度を調整するにはどうしたらいいだろうか？

ヒント：　println(“light[” + i + “][” + j + “] = 1;”);　という1行をカーソルの当り判定の中に入れよう。

プログラミング演習I（第11回）課題

【3組】

- **発展①　スケッチ名：circlebutton**
  - 画面の中央に円を置き、その中にカーソルが入った時に効果音が鳴るようにしてください。
  - カーソルが円の中に入る時のみ音が鳴るようにし、中に居続けている間は音が鳴らないようにしてください。



[step1]
まずは円とマウスカーソルの当り判定のプログラムを作ろう。

[step2]
音を鳴らすための処理を追加しよう。

[step3]
カーソルが入る時のみ音を出すにはどうしたらいいだろうか？　入ったという情報を保持しておいて、入ったことを判定するタイミングで「既に入っているかどうか」を判定すればいけそうだ。

プログラミング演習I（第11回）課題

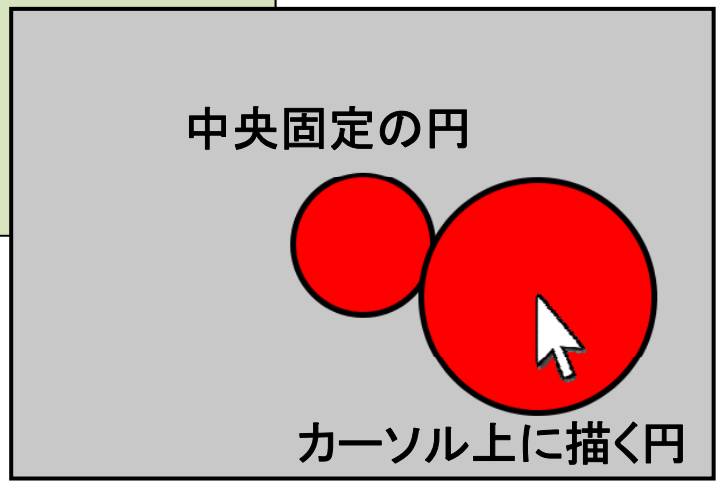
【3組】

- **発展②　スケッチ名：collision**
  - 画面中央に固定された円と、マウスカーソル上に表示される円の2つがある。互いに接触または重なっているときに色が赤くなるプログラムを作成してください。
  - なお、円同士の接触判定をする以下の関数を必ず作ること。

```
int collision( 円1のx座標, 円1のy座標, 円1の半径,
               円2のx座標, 円2のy座標, 円2の半径 ) {

    // 接触していないとき flag = 0
    // 接触しているとき flag = 1

    return flag;
}
```



描画速度について知ろう

- 描画処理は draw() が繰り返し実行されることによって行われています。これはちょうど、スケッチブックのページをめくって描いてを何度も繰り返していくイメージです。

- draw()で描画される１画面のことを「１フレーム」と呼びます。これは映画のフィルムの１コマと同じ概念です。また、描画速度のことをフレームレート (frame rate) と呼びます。

- フレームレートは「1秒間に何フレーム表示するか？」という値で定義されます。単位は FPS (Frame Per Second : フレーム/秒) です。例えば、30FPS ならば １秒間に30フレーム表示していることになります。

- だいたい 30～60 FPS あれば、人の目には滑らかに動いているように見えます。

描画速度を変えるには？

- Processingの描画速度は、デフォルトで 60FPS になっています。任意の速度を指定したいときは、 frameRate() という命令を使います。 setup() の中で指定してください。

```
void setup() {
    size( 400, 300 );
    frameRate( 120 );  // 120FPSに設定
}
```

- ここで指定した通りの速度で必ず動いてくれるわけではなく、コンピュータの処理能力や、描画処理の内容によって、実際はこれより遅くなることがあります（無茶ぶりはできない）。

処理にウェイトを入れたいときは？

- 物体の移動速度を遅くしたり、処理と処理の間に待ち時間を入れたいときは、delay() という命令を使うとよいです。指定した時間だけ停止させられます。値は ミリ秒 で指定します。

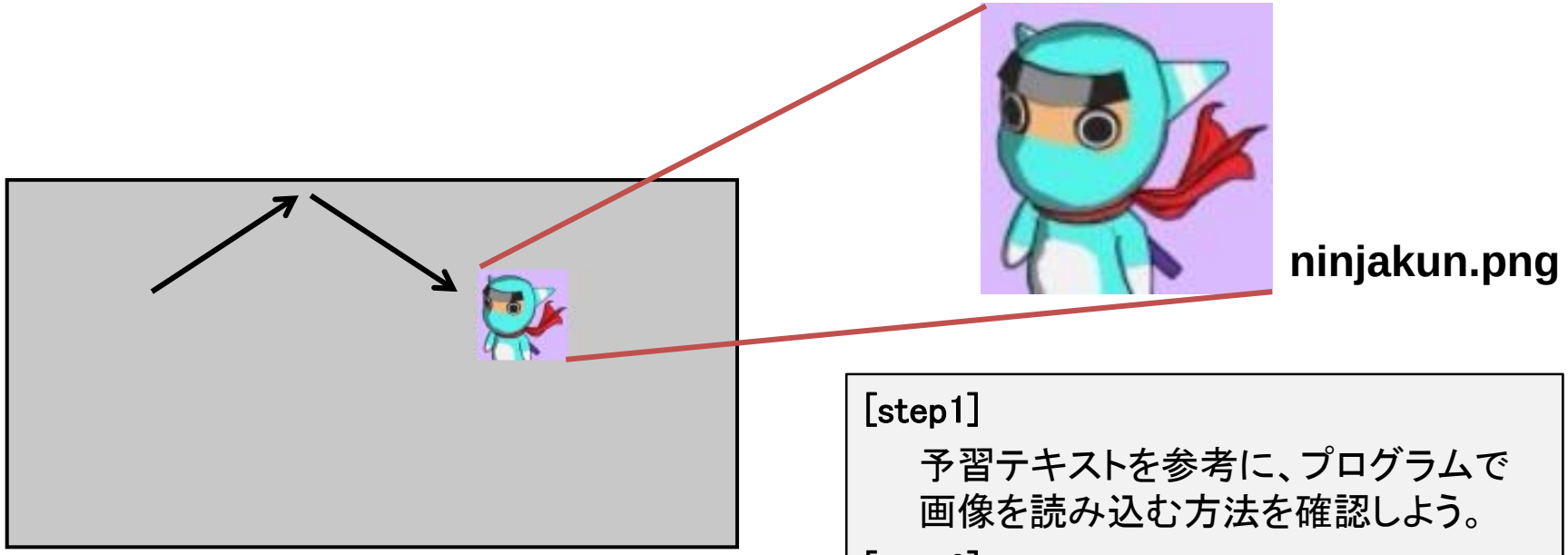
```
int pos = 0;
void draw() {
    background(255);
    pos++;
    if (pos > width) {
      pos = 0;
    }
    ellipse( pos, 60, 50, 50 );

    delay(100);  // 100ms待つ
}
```

プログラミング演習I（第11回）課題

【4組】

- **基本①　スケッチ名：boundimage**
  - 以前作った、ボールが跳ね返るプログラム（bound）において、円を表示する部分を、好きな画像に差し替えてください。



ninjakun.png

[step1]

予習テキストを参考に、プログラムで画像を読み込む方法を確認しよう。

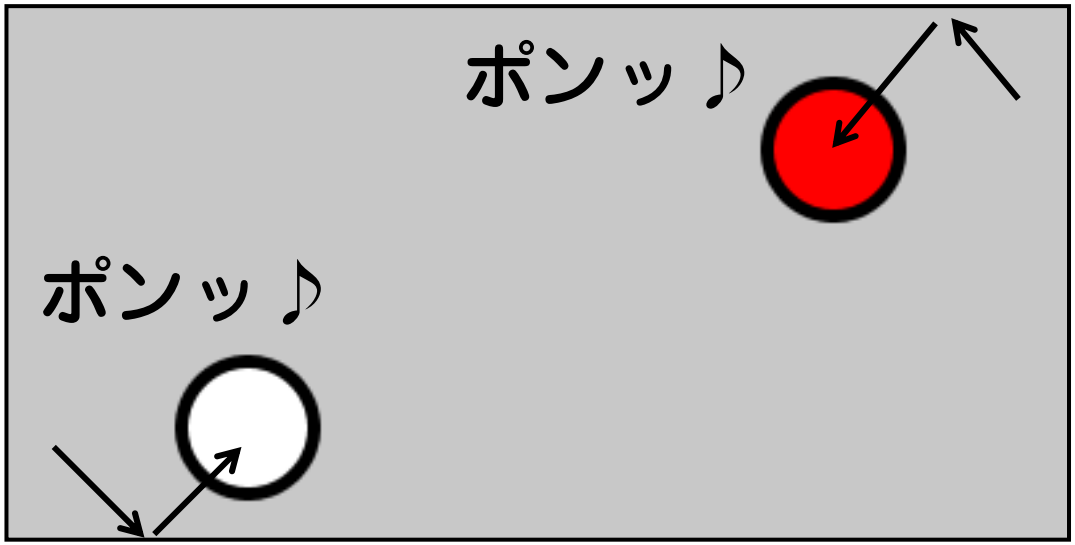
[step2]

以前ellipse()命令で円を描いていた場所を、image()命令に差し替えて画像を表示させよう。サイズも調整しよう。

プログラミング演習I（第11回）課題

【4組】

- **基本②　スケッチ名：bound2**
  - 画面内を2つの円が移動し、壁に当たると跳ね返るプログラムを作成してください。なお、壁に当った時に適当な効果音が鳴るようにしてください。



[step1]

まずは以前作ったboundというプログラムを思い出して、それをベースに2個の円を動かせるようにしよう。配列を使うとよいかも？

[step2]

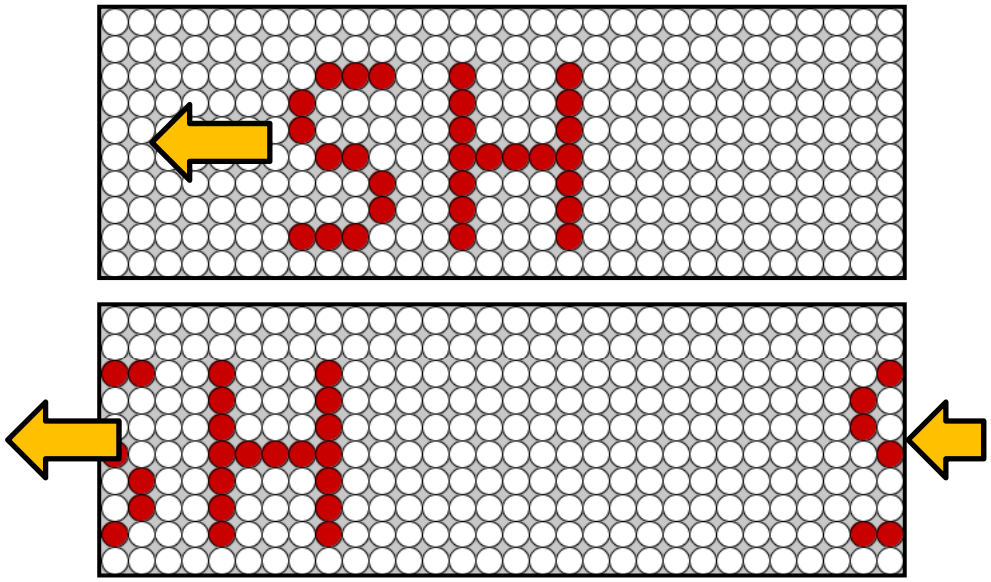
衝突したときに音が鳴る処理を加えてみよう。

2つの円で効果音は同じでOK。余裕のある人は効果音を2つ使ってみよう。

プログラミング演習I（第11回）課題

【4組】

- **基本③　スケッチ名：keijiban3**
  - 電光掲示板のプログラム（keijiban2）を改造して、左端から右端へのループと、自動スクロールを実現してください。
  - プログラム実行直後に自分のイニシャル2文字が流れるようにすること。
  - 文字が読める速度にすること。



[step1]

まずは実行直後に文字が表示されるようにしよう。setup()の中で配列にあらかじめデータを入れておけばOK。

[step2]

左に文字が流れる処理の部分を改良して、ループするようにしよう。両端をどう処理するかがカギだ。

[step3]

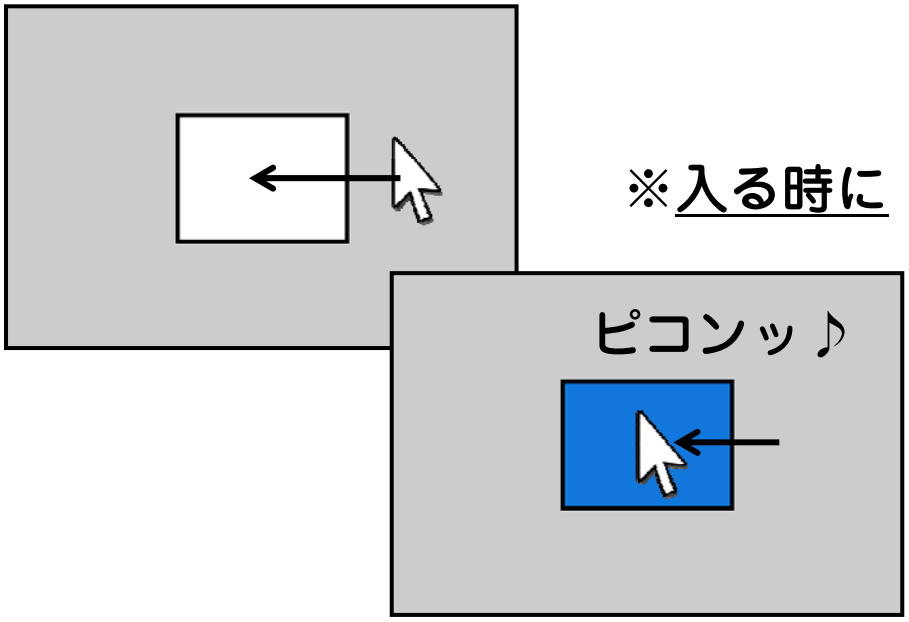
最後に自動ループさせよう。流れる速度を調整するにはどうしたらいいだろうか？

ヒント：　println(“light[” + i + “][” + j + “] = 1;”);　という1行をカーソルの当り判定の中に入れよう。

プログラミング演習I（第11回）課題

【4組】

- **発展①　スケッチ名：rectbutton**
  - 画面の中央に四角形を置き、その中にカーソルが入った時に効果音が鳴るようにしてください。
  - カーソルが四角形の中に入る時のみ音が鳴るようにし、中に居続けている間は音が鳴らないようにしてください。



[step1]
まずは四角形とマウスカーソルの当り判定のプログラムを作ろう。

[step2]
音を鳴らすための処理を追加しよう。

[step3]
カーソルが入る時のみ音を出すにはどうしたらいいだろうか？　入ったという情報を保持しておいて、入ったことを判定するタイミングで「既に入っているかどうか」を判定すればいけそうだ。

プログラミング演習I（第11回）課題

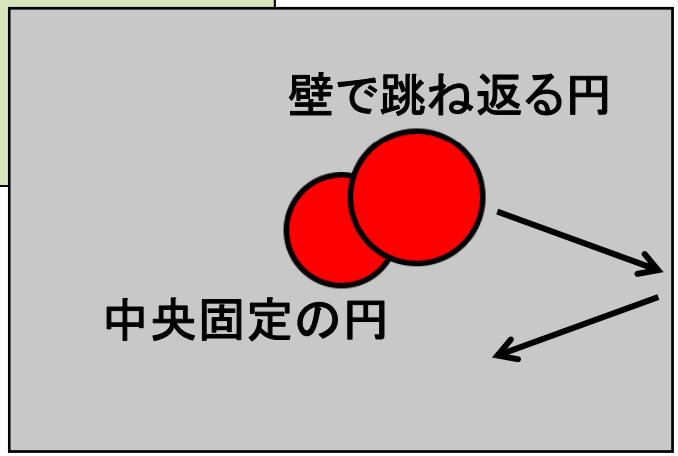
【4組】

- **発展②　スケッチ名：detect**
  - 画面中央に固定された円と、移動して壁で跳ね返る円の2つがある。互いに接触または重なっているときに色が赤くなるプログラムを作成してください。
  - なお、円同士の接触判定をする以下の関数を必ず作ること。

```
int detect( 円1のx座標, 円1のy座標, 円1の半径,
            円2のx座標, 円2のy座標, 円2の半径 ) {

    // 接触していないとき flag = 0
    // 接触しているとき flag = 1

    return flag;
}
```



描画速度について知ろう

- 描画処理は draw() が繰り返し実行されることによって行われています。これはちょうど、スケッチブックのページをめくって描いてを何度も繰り返していくイメージです。

- draw()で描画される１画面のことを「１フレーム」と呼びます。これは映画のフィルムの１コマと同じ概念です。また、描画速度のことをフレームレート (frame rate) と呼びます。

- フレームレートは「1秒間に何フレーム表示するか？」という値で定義されます。単位は FPS (Frame Per Second : フレーム/秒) です。例えば、30FPS ならば １秒間に30フレーム表示していることになります。

- だいたい 30～60 FPS あれば、人の目には滑らかに動いているように見えます。

描画速度を変えるには？

- Processingの描画速度は、デフォルトで 60FPS になっています。任意の速度を指定したいときは、 frameRate() という命令を使います。 setup() の中で指定してください。

```
void setup() {
    size( 400, 300 );
    frameRate( 120 );  // 120FPSに設定
}
```

- ここで指定した通りの速度で必ず動いてくれるわけではなく、コンピュータの処理能力や、描画処理の内容によって、実際はこれより遅くなることがあります（無茶ぶりはできない）。

処理にウェイトを入れたいときは？

- 物体の移動速度を遅くしたり、処理と処理の間に待ち時間を入れたいときは、delay() という命令を使うとよいです。指定した時間だけ停止させられます。値は ミリ秒 で指定します。

```
int pos = 0;
void draw() {
    background(255);
    pos++;
    if (pos > width) {
      pos = 0;
    }
    ellipse( pos, 60, 50, 50 );

    delay(100);  // 100ms待つ
}
```