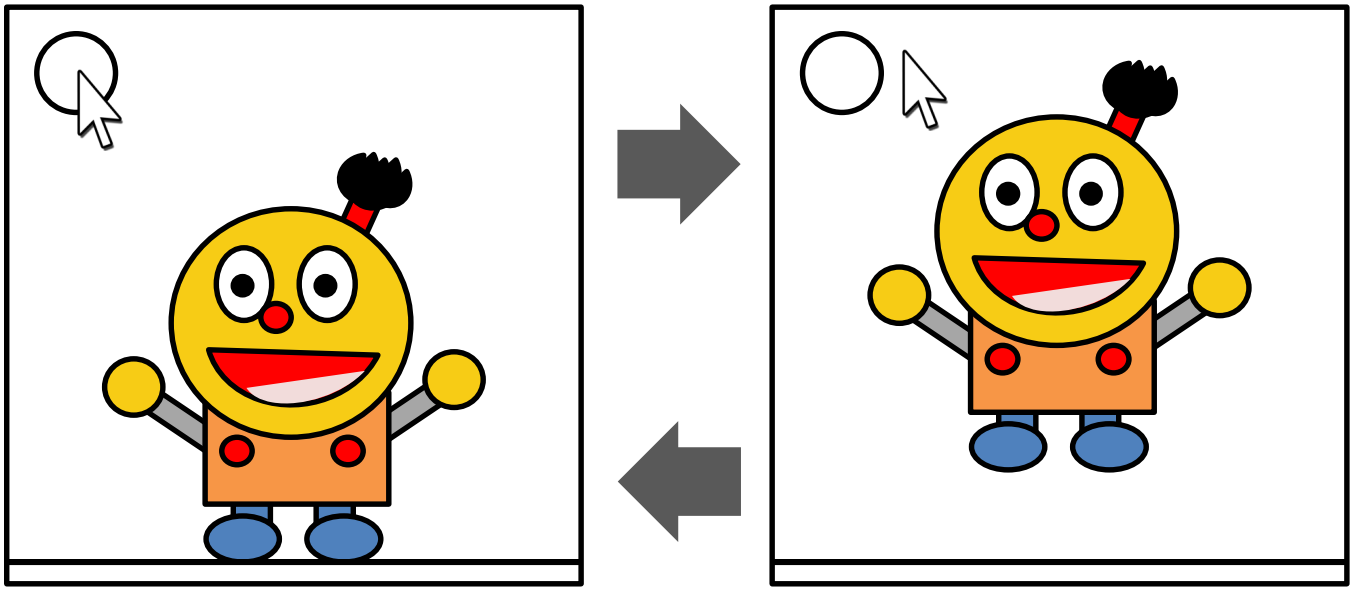
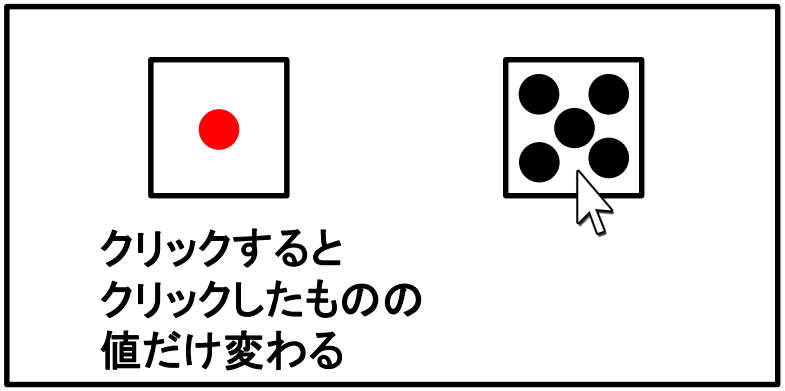
プログラミング演習I（第5回）課題

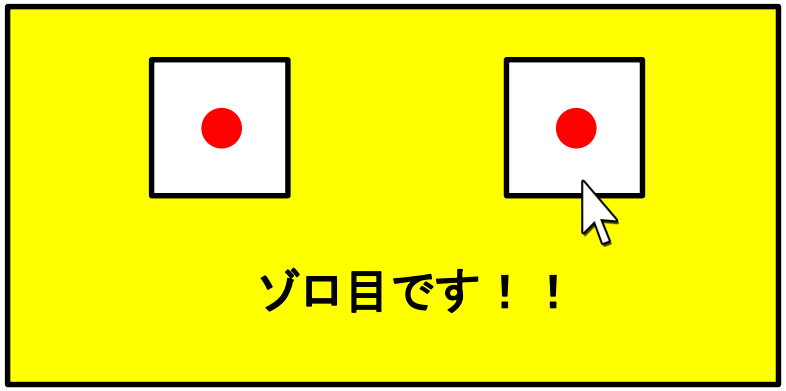
- **基本課題①　スケッチ名：charajump2**
  - キャラクタと地面となる面を表示し、画面左上に用意した直径50以下の丸ボタンをクリックすることでキャラクタがジャンプするようなプログラムを作成せよ（丸ボタン以外ではジャンプしないこと）
  - ジャンプについては初速をわかる程度に設定しつつ鉛直上方投射するものとし，地面にくると止まるようにせよ
  - 丸ボタンを押すと何度でもジャンプできるようにせよ
    - ジャンプ中の挙動については，何も起こらないようにしても，そこからジャンプしても，初期値に戻ってもどちらでもよい



プログラミング演習I（第5回）課題

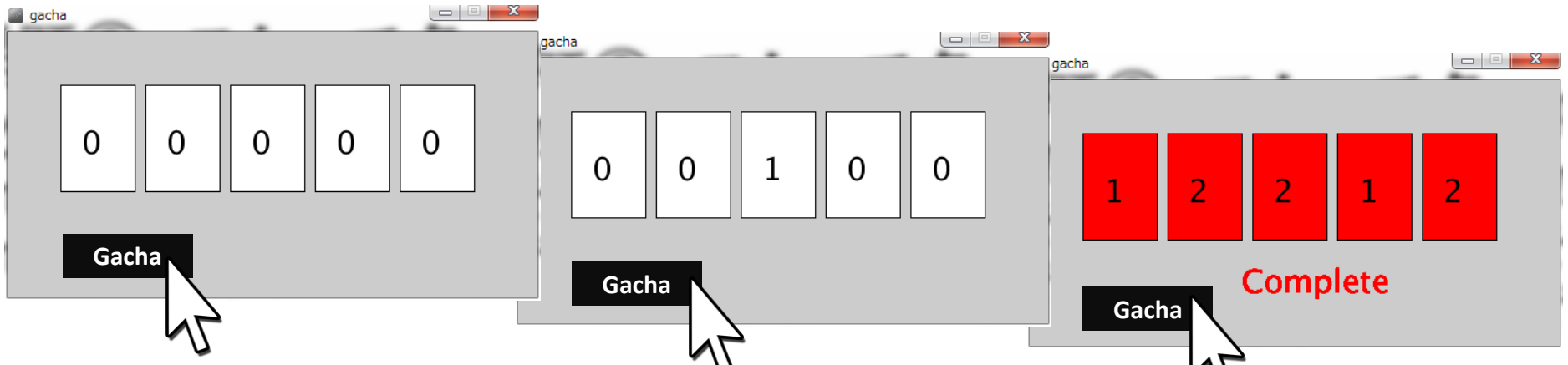
- **基本課題②　スケッチ名： dice**
  - 2つの四角形を表示し、その四角形内をクリックするとそのサイコロの目が変化するプログラムを作成せよ
    - 初期の値は「１」と「５」にせよ。
    - クリックされたサイコロの値だけランダムに変更するようにせよ。
    - サイコロの表示は、数がわかれば、単純に数字で表示しても、●で表現しても良い
  - また、2つのサイコロの目が一致したときに（ゾロ目になったとき）、画面を華やかにせよ（どう華やかにするかは任せます）





プログラミング演習I（第5回）課題

- **基本課題③　スケッチ名：complete_gacha**
  - ウインドウ下部のガチャボタンをクリックする度にクリックする度に5種類のカードの1種類がランダムに選ばれ，枚数が１加算されるプログラムを作成し，それぞれのカードが選ばれた枚数を表示するプログラムを作成せよ（ただしボタン以外では反応しないようにせよ）
  - また，すべてのカードが1枚以上になったら，Completeとウインドウ内にtextを用いて表示し，カードの色を赤色にせよ

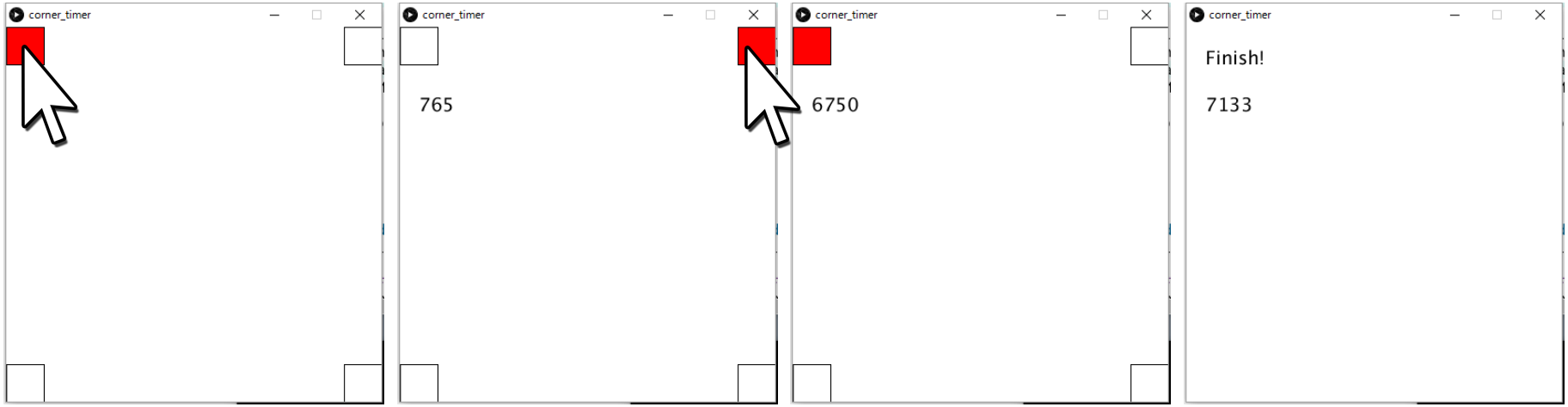


ヒント

- 基本課題1
  - 月曜日の発展課題のcharajumpに挑戦し，そのプログラムを改良しよう！
- 基本課題2
  - それぞれのサイコロについて，情報をもつようにしましょう．また，クリックされたらそのサイコロの目だけ変更するようにしましょう．
  - どのタイミングで何を描画したらいいのかということを整理しましょう
- 基本課題3
  - ボタンを描画する（日本語のためにはフォントを指定）
  - すべてが1枚以上というのはどういう条件なのかを整理

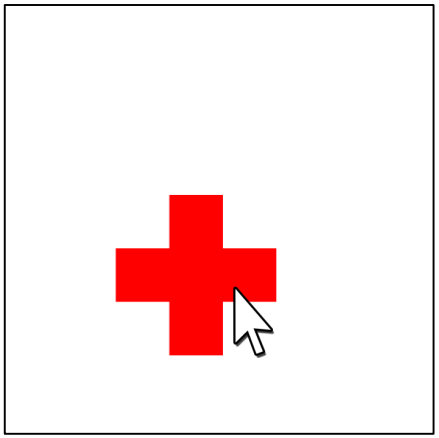
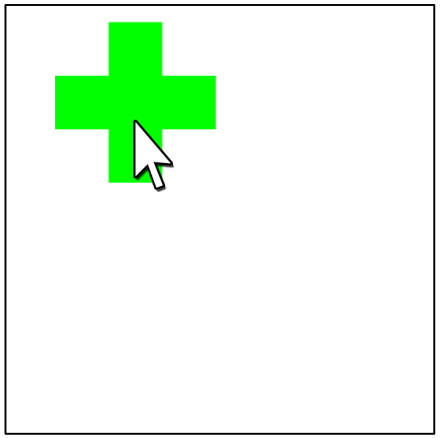
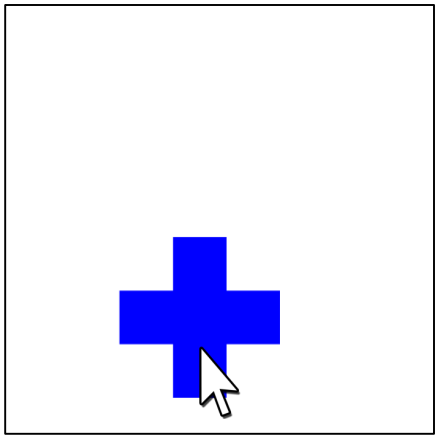
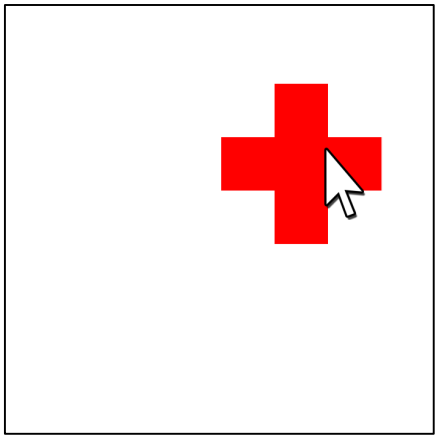
プログラミング演習I（第5回）課題

- **発展課題①　スケッチ名：corner_time**
  - 400x400のウインドウ（背景白色）内に4つの四角形を配置し，その四角形の内の1つを赤色で塗りつぶせ．また，その赤色の四角形内をクリックすると次の場所（左上→右上→左下→右下→左上の順で変更）に赤色の表示位置を切り替えよ
  - ただし，赤色の四角形外をクリックした時には，場所が変更されないようにせよ（つまり反応しないようにせよ）
  - また，2周（9回分クリック）したら終了するようにし，最初にクリックされてから，9回目がクリックされるまでの時間をミリ秒でリアルタイムに提示し，その9回終わると結果を提示するようにせよ



プログラミング演習I（第5回）課題

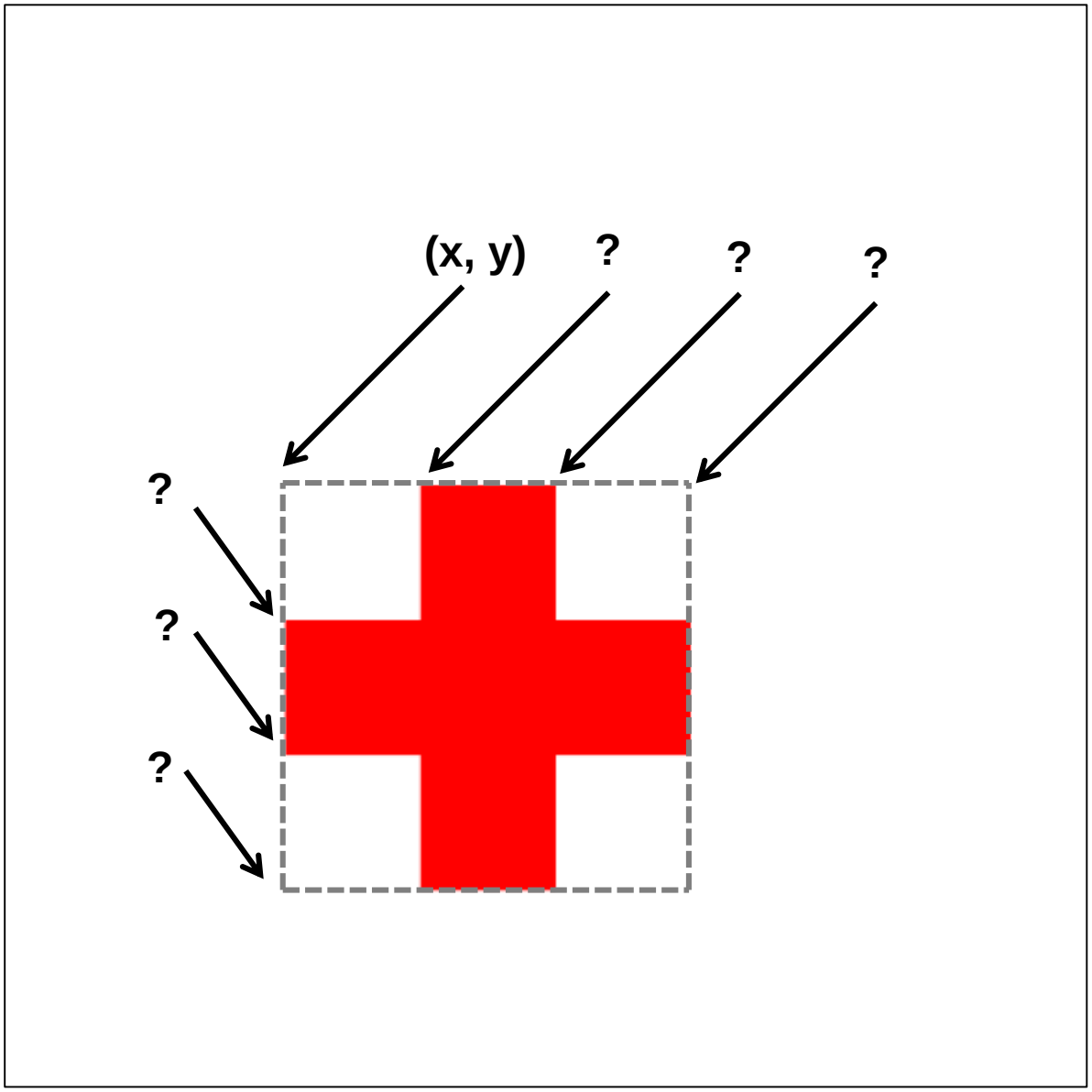
- **発展課題②　スケッチ名：randomCross**
  - 800x800の大きさのウインドウの中に、縦横300ピクセルの十字型の図形（太さを100ピクセルとする）を提示し、その図形内をクリックすると、別の位置に十字型の図形が移動するようにせよ。
  - なお、クリックするたびに赤→緑→青→赤（以降繰り返し）と色が変わるようにせよ。
  - また、十字の図形外をクリックした場合は反応しないようにせよ。
  - また、十字の図形は画面外に飛び出さないようにせよ。

プログラミング演習I（第5回）課題

- ヒント
  - まずは十字を左上の座標をベースとして描画できるようにしよう。どうしたらよいか？
    - xとyという変数を使うと、各座標はどう表現できる？
  - 次に、その十字との当たり判定を作ろう
  - 十字とクリックしたら座標を変更するようにしよう
  - 変更の際に、外に飛び出ないようにするためrandomを考慮しよう
  - 十字の色塗りをクリックの度に変更するようにしよう
    - 色を設定する変数を用意する

プログラミング演習I（第5回）課題



今日使うテクニック

① text()で表示する文字の大きさを変える方法

- 文字の大きさを変えるには　textSize(文字サイズ) を使う。

```
void setup() {
  size(300, 150);
}

void draw() {
  fill(0);
  textSize(50); // 文字の大きさを設定
  text( "Processing", 20, 90 ); // 文字を表示
}
```

- textSize()は、fill()やstroke()と同様に何回でもパラメータを変えて指定できるので、大きさの違う文字を混在させることができる。

今日使うテクニック

② text() で表示する文字の書体(フォント)を変える方法

- フォントを変えるには、PFont、createFont()、textFont() を使う。
- **日本語を使いたいときは日本語フォントの指定が必要**
- 以下はHGS創英角ポップ体で「Processing」と書く例

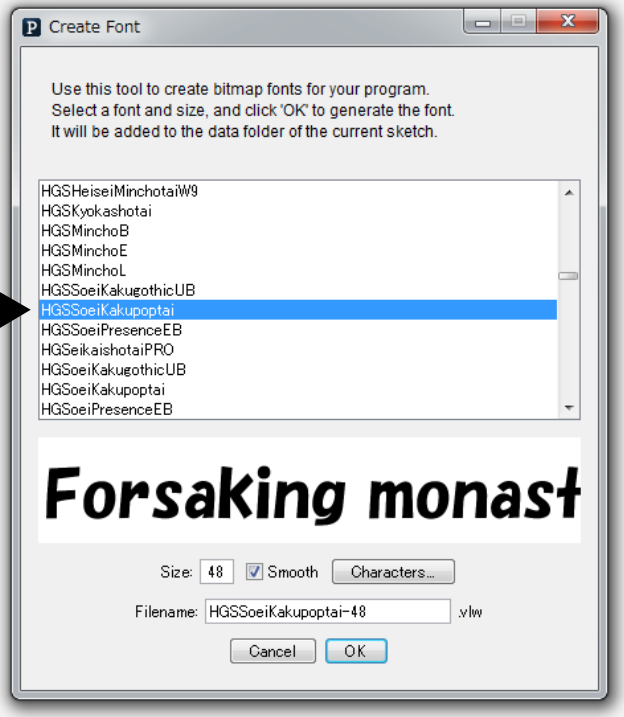
```
PFont myFont; // フォント

void setup() {
  size(300, 150);
  myFont = createFont("HGSSoeiKakupoptai",10);  // フォントを準備
  textFont(myFont);  // フォントを設定
  textSize(50);          // 文字サイズを改めて変更することもできる
}

void draw() {
  fill(0);
  text( "Processing", 20, 90 ); // 文字を表示
}
```

今日使うテクニック

- PFont はフォントを格納する変数につかうデータ型です。
  int や float などと同じような扱い。

- **createFont(フォント名, 文字サイズ)** でフォントを準備する。
  フォント名は、Processingのメニューの
  Tools -> Create Font...
  で出てくるパネルで確認できる。

**このリストにプログラム中で使える
フォント名が表示される。**



- 最後に、**textFont(フォント)** で
  フォントを設定する。

今日使うテクニック

③ millis()でミリ秒単位の経過時間を取得する

- アプリケーションが起動されてからの時間は millis() で取得することが可能

```
int iStartMillis;
boolean bFlagStart = false; // スタートしたかどうかのフラグ
void setup() {
  size(300, 150);
  fill( 0 ); // 文字色を黒色に設定
}
void draw() {
  background( 255 );
  if( bFlagStart ){ // スタートしていたら～
    text( millis()-iStartMillis, 20, 90 ); // 差分で経過時間を表示
  }
}
void mousePressed(){
  bFlagStart = true; // クリックされたらスタートフラグを立てる
  iStartMillis = millis(); // スタートの経過時間をセット
}
```