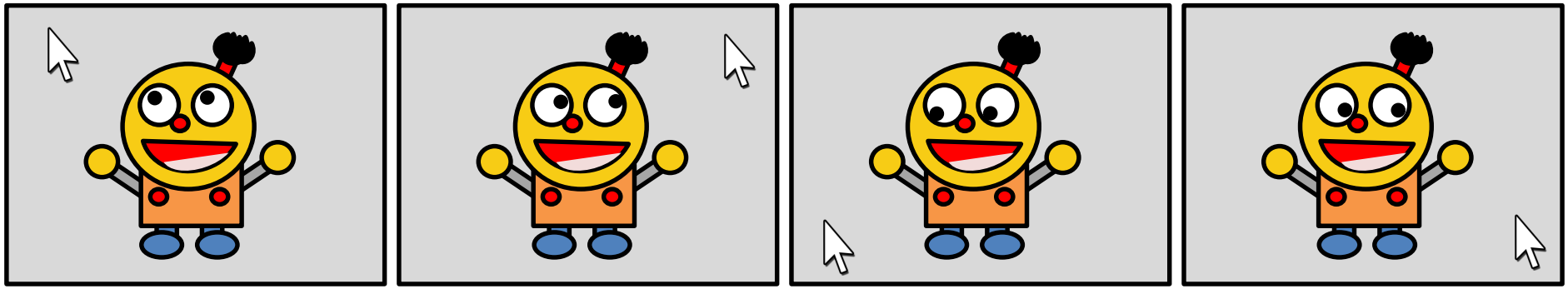
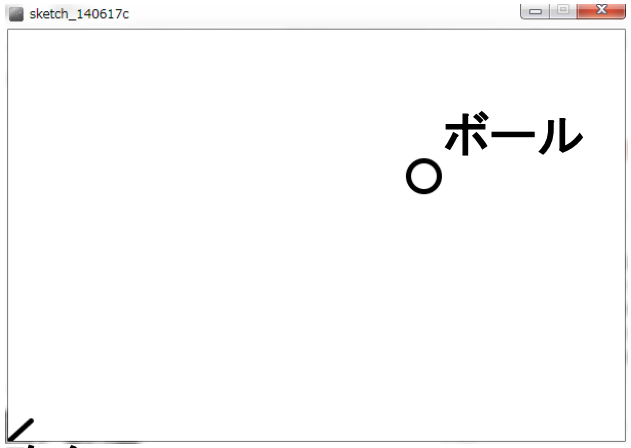
プログラミング演習I（第5回）課題

- **基本課題①　スケッチ名：eye2**
  - カーソルの位置によってキャラクタの目の向きが変わるプログラムを作ってください。
  - ただし、カーソルがキャラクタの顔に対して【左上にある時】【右上にある時】【左下にある時】【右下にある時】の4パターンで表現すること。



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：launch2**
  - 600x400のウインドウ左下から砲台（砲台の長さはだいたい40ピクセル）の向きにボール（半径15m）を発射するプログラムを作成せよ。
  - ただし、砲台は**上下キーによって角度を1度ずつ変更**（初期角度は45度）し、**右キーで投射**されるようにすること。また、初速を100m/sとせよ。

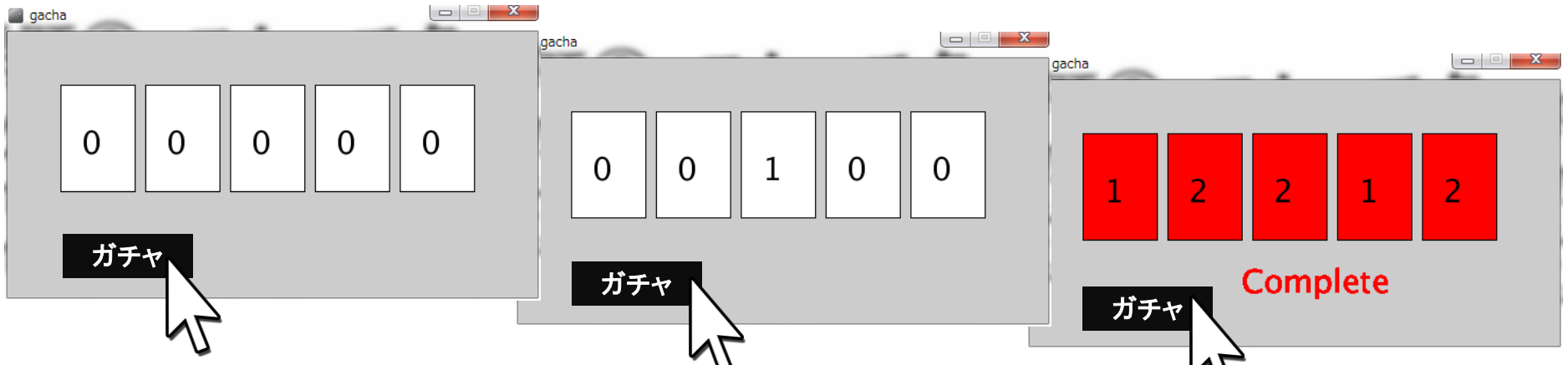


$$x = v_0 t \cos\theta$$

$$y = v_0 t \sin\theta - \frac{1}{2} g t^2$$

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（ただしボタン以外のクリック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」と表示し，カードの色が赤色になるとともに，画面を華やかにせよ

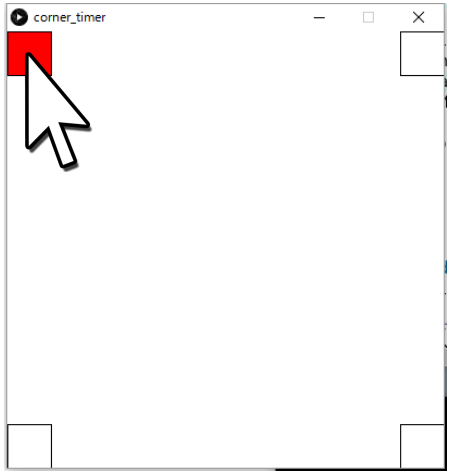


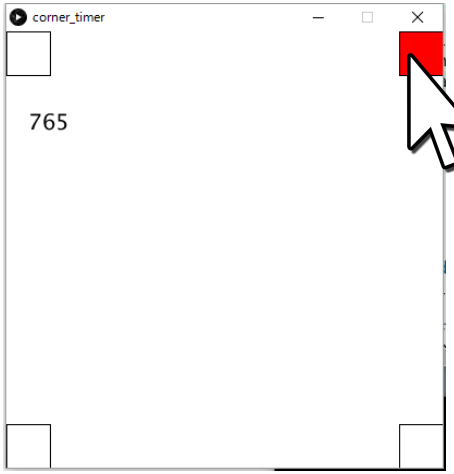
ヒント

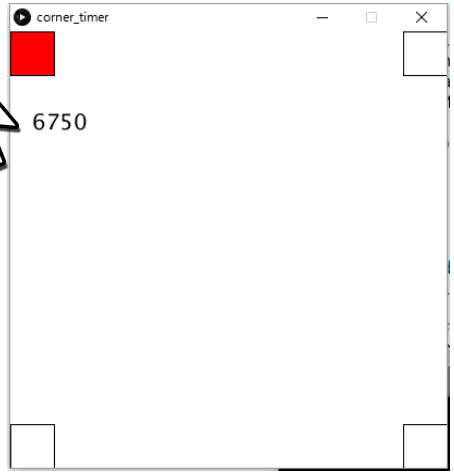
- 基本課題1
  - カーソルの場所に応じて条件分岐を行い，目の描画を切り替えるだけ！
- 基本課題2
  - キー操作を取得する方法を予習資料でチェック！
  - 角度を管理する変数を用意しておいてキー操作で値を変化させていく．で，発射するだけ
- 基本課題3
  - ボタンを描画する（日本語のためにはフォントを指定）
  - すべてが1枚以上というのはどういう条件なのかを整理！

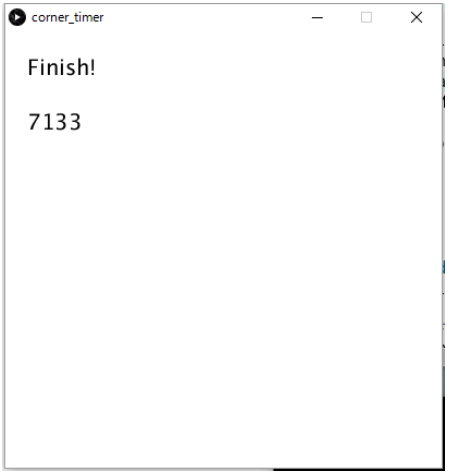
プログラミング演習I（第5回）課題

- **発展課題①　スケッチ名：corner_time**
  - 400x400のウインドウ（背景白色）の四隅に縦横50ピクセルの四角形を描画し，その内の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



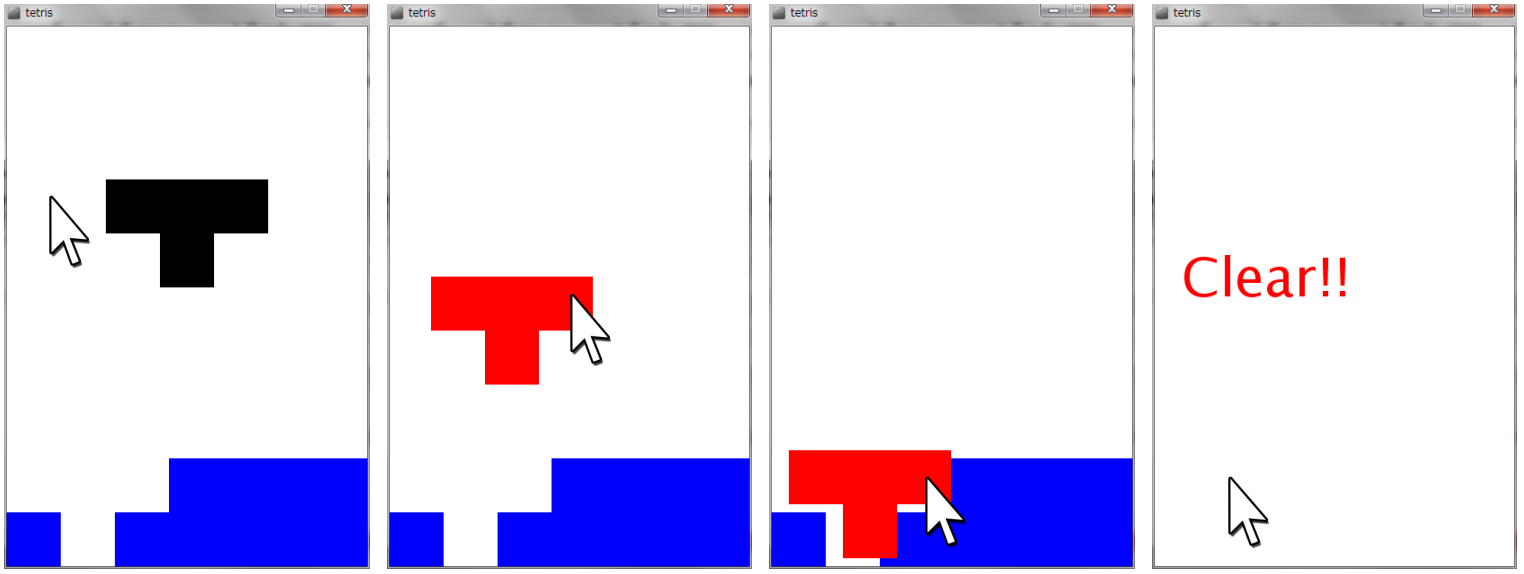




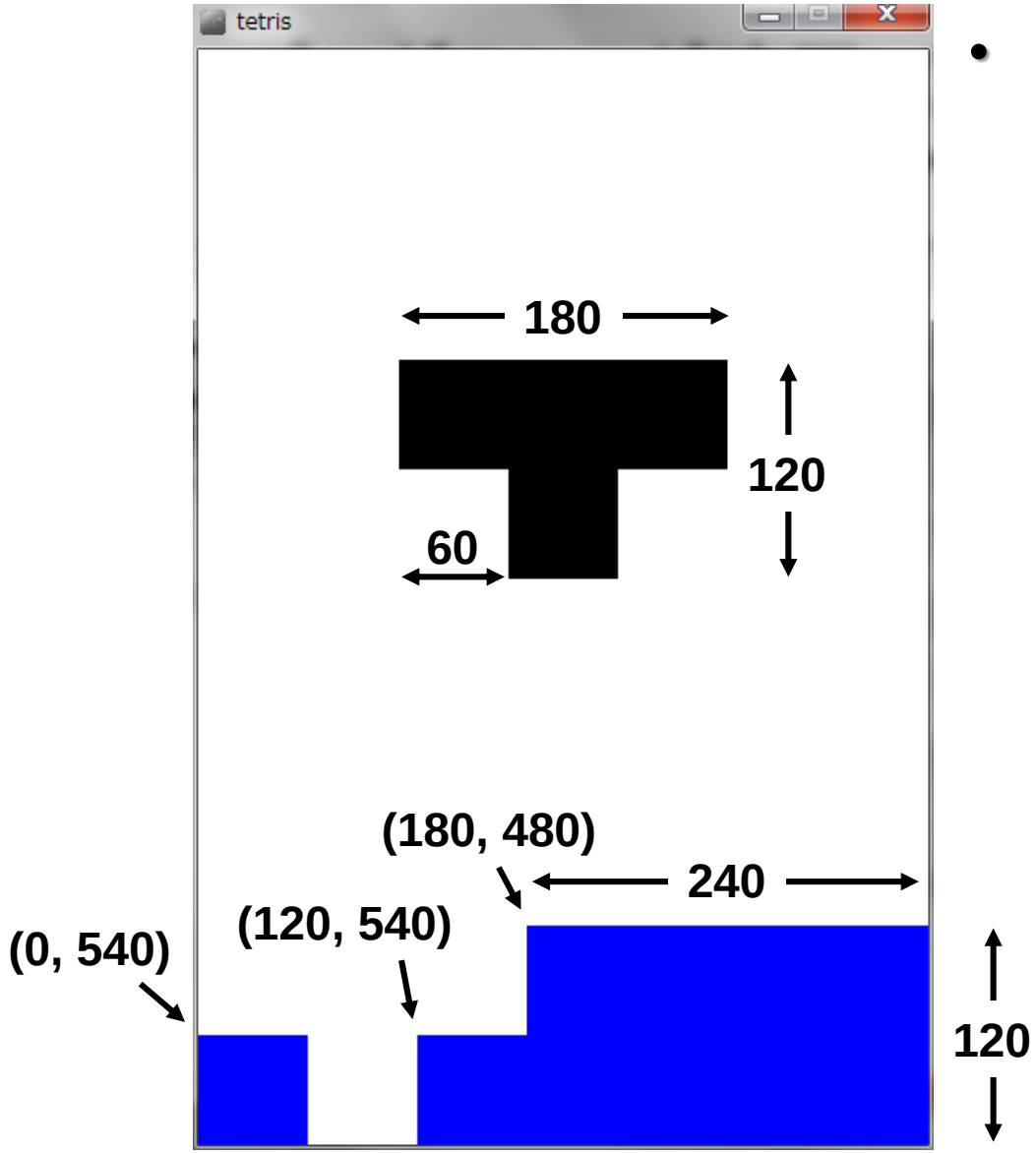


プログラミング演習I（第5回）課題

- **発展課題②　スケッチ名：Tetris**
  - 420x600のウインドウ内に、下図に示す横180ピクセル縦12ピクセルの凸型の図形を描き（黒色）、図形内でマウスボタンを押すと選択状態（赤色）になり、ボタンを離すまで移動できるようにせよ。
  - なお、移動においては相対位置をキープするようにして下さい。また、ドラッグ中は図形が赤色になるようにせよ。
  - また、下図のように他のブロックを用意し、その隙間にピッタリハマった場合に「Clear!!」と表示するようにせよ（音を鳴らしても良い）



ヒント



- 必要な変数は何か？ 右に変数名を書き込もう！

  マウスの座標（　　，　　）

  ブロックの基本座標（　　，　　）

  x方向の相対座標:

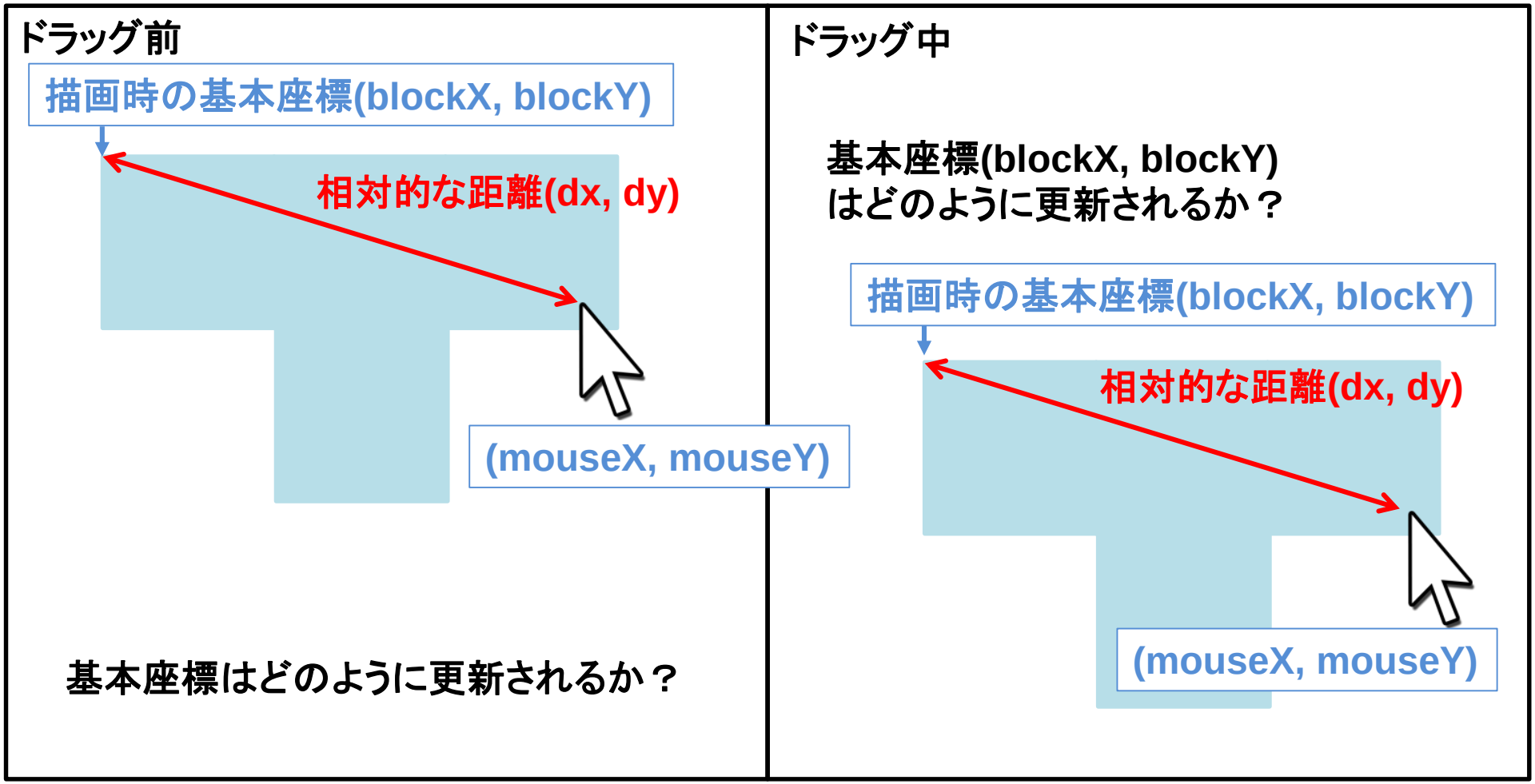
  y方向の相対座標:

  選択の有無:

  クリアしたかどうか:

ヒント

- 相対座標はどのようにして保持し，その時の座標にどのようにして反映するか整理！



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

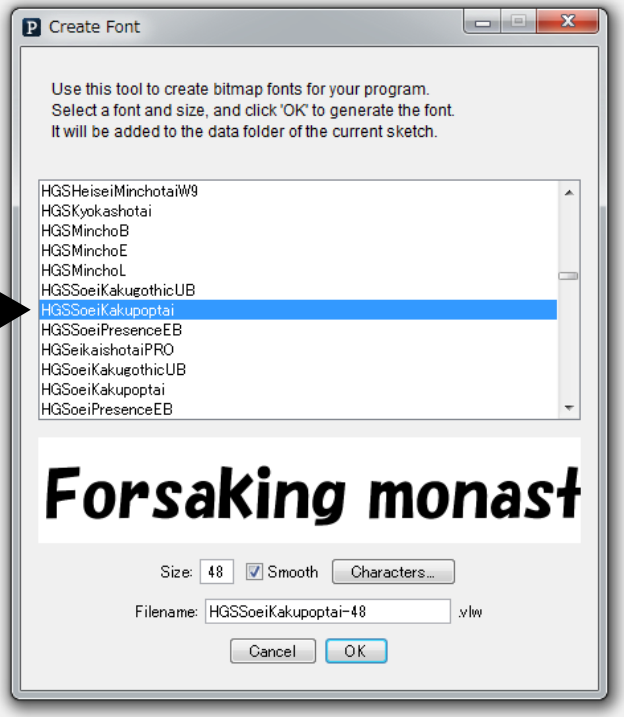
```
PFont myFont; // フォント

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
}
```