

計算機室体験へようこそ！

情報科学類のオープンキャンパスへお越しくださった皆さんに、学生スタッフが計算機室の体験会を用意いたしました。今皆さんがいる**計算機室**は、情報科学類の授業で実際に使用されている部屋です。この計算機室体験を通して少しでも雰囲気を感じ取ってくだされば幸いです。体験の中で何かわからないことがあった場合、また具合が悪くなってしまった場合は遠慮なく学生スタッフにお申し付けください。

■お願い 計算機室（この教室）内は**飲食厳禁**です。高価な機材もあり、飲み物などをこぼしてしまうと機材が故障してしまう可能性があります。ご協力をよろしくお願いいたします。

STEP1 円を描こう

今回の計算機室体験では **Processing** というプログラミング言語を使います。この言語では、簡単な命令で図形を描くことができます。

早速 Processing を起動してみましょう。画面の一番下側に次の画像のようなもの（**Dock** と呼びます）が見えると思います。マウスのポインタ（矢印）をこの Dock に近づけてみると、各アイコンの上に名前が表示されます。四角で囲まれた「アプリケーション」という名前のアイコンをクリックしましょう。

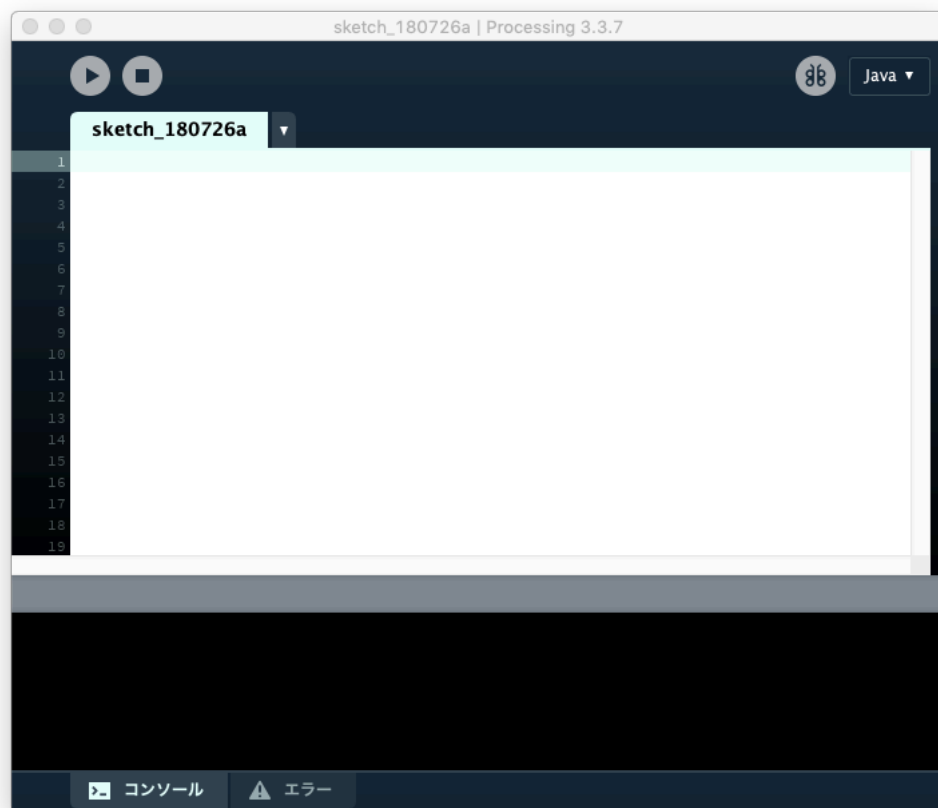


以下のようにアプリケーションの一覧が開きます。四角で囲まれた「Processing」のアイコンをクリックしましょう。




以下のような画面が表示されましたか？ これで Processing が起動しました。


試しに円を描いてみましょう。以下のコードをこのウィンドウに書き写してみてください。括弧（・）は **shift** キーを押しながら 8 や 9 のキーを押すと入力できます。

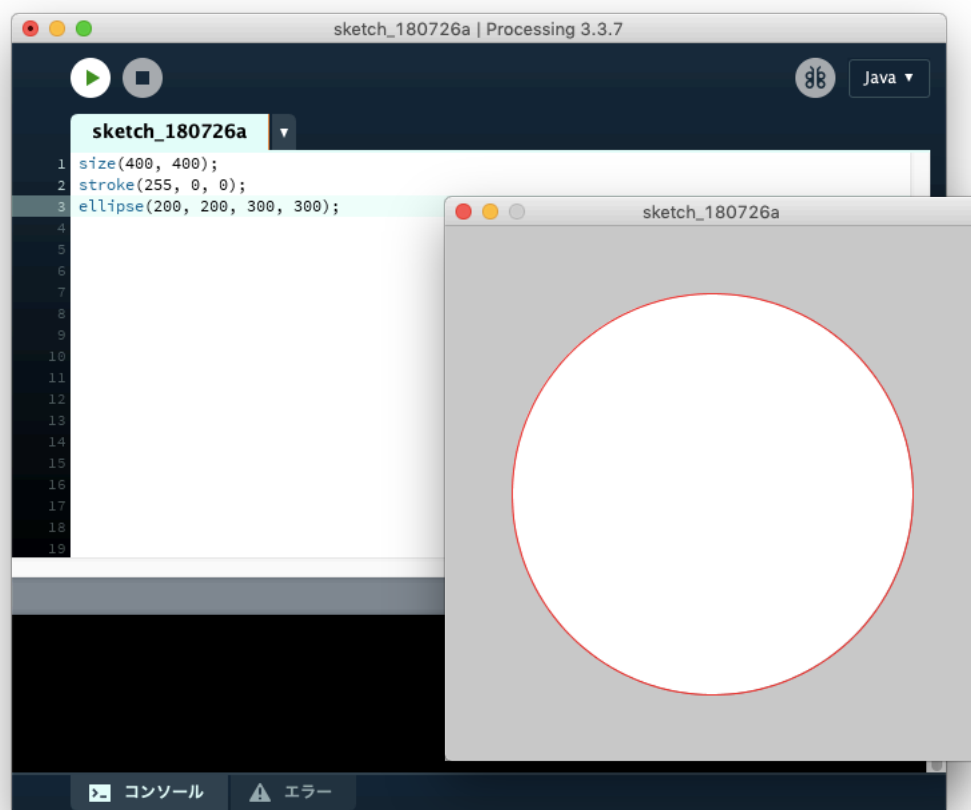


```
1 size(400, 400);  
2 stroke(255, 0, 0);  
3 ellipse(200, 200, 300, 300);
```

書き終わったら実行ボタン  をクリックします。すると今入力したコードを元にして円が描かれます。

うまくいかないときは…（次のことを確認してみよう）

- **スペルミス**がないか確認してみましょう。
- アルファベットは**すべて小文字**で入力されているか確認してみましょう。もしアルファベットが大文字で入力されてしまう場合は、キーボードの **caps キー** のランプが点灯しているかもしれません。点灯していたら押すと消えます。ランプが消えたらもう一度入力してみましょう。
- **セミコロン「;」**を入力し忘れていないか確認してみましょう。セミコロン「;」のキーは **L キー** の右隣りにあります。
- 入力したコードに波線  が引かれていれば、その箇所がエラーの原因かもしれません。



各行の命令について詳しく説明します。

1 行目 `size` 命令

これはキャンバスの大きさ指定する命令です。以下のように使います。

```
size( 横の長さ, 縦の長さ );
```

キャンバスとは、実行したときに出てくる四角のウィンドウのことです。

2 行目 `stroke` 命令

これは線の色を指定する命令です。以下のように使います。

```
stroke( R, G, B );
```

赤, 緑, 青の強さを 0 ~ 255 の範囲で指定することで色を表現します。**RGB** とは三原色のことであり, **R**ed, **G**reen, **B**lue の頭文字を取った呼び方となっています。

■例 赤 : (255, 0, 0), 緑 : (0, 255, 0), 青 : (0, 0, 255),
黄 : (255, 255, 0), 黒 : (0, 0, 0), 白 : (255, 255, 255) など

3 行目 `ellipse` 命令

これは描く円の位置と形を指定する命令です。正円だけでなく楕円を描くこともできます。以下のように使います。

```
ellipse( 中心の x 座標, 中心の y 座標, 縦の直径, 横の直径 );
```

数学とは違って軸の取り方が異なるので注意しましょう。軸の中心はキャンバスの左上端です。

Processing でよく使われる命令

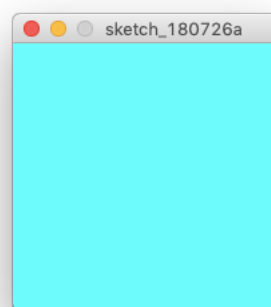
background 命令

キャンバスを指定した色で塗りつぶします。`stroke` と同様に RGB で指定します。

```
background( R,  G,  B);
```

実行例

```
1 size(200, 200);  
2 background(0, 255, 255);
```



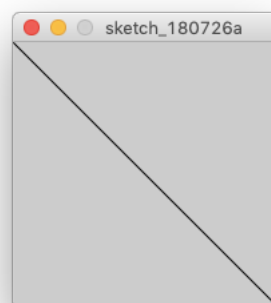
line 命令

指定した点から点に線を引きます。

```
line( 始点の x 座標,  始点の y 座標,  終点の x 座標,  終点の y 座標);
```

実行例

```
1 size(200, 200);  
2 line(0, 0, 200, 200);
```



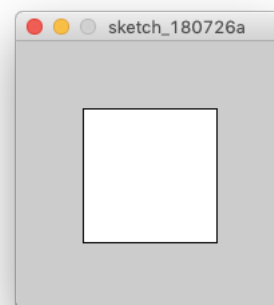
rect 命令

長方形を描きます。

```
rect( 左上の x 座標, 左上の y 座標, 幅, 高さ );
```

実行例

```
1 size(200, 200);  
2 rect(50, 50, 100, 100);
```



STEP2 イベントを使おう

Processing には、イベント駆動プログラミングという考え方が採用されています。この考え方では、ユーザーからの操作（これをイベントと呼びます）によって、それぞれ決められた処理を実行します。すなわち、マウスやキーボードなどの使い方で結果を変えることができるということです。

このステップでは、イベントを用いて Processing を動かしてみましょう。

主なイベント

setup()

プログラムを実行したときに最初に一度だけ実行されます。STEP1 で紹介した `size()` や `background()` などは、通常ここで最初に宣言されます。

draw()

画面を描画するたびに実行されます。Processing は画面を 1 秒間に何度も描画するため、`draw` も繰り返し実行されます。

mouseDragged()

マウスドラッグを検出して処理を行います。マウスドラッグとはマウスを動かすことです。つまり、マウスが動かされている間のみ実行されます。このイベントは `draw()` を呼び出すと自動的に実行されます。

例 1

以下のコードは、ドラッグしている間キャンバスが黒色になるプログラムです。コメントアウトされているもの（`//○○○`，`/*○○○*/`）はプログラム処理として読み込まれないため、写す必要はありません。特定の記号や文字（コメント）をプログラムの処理から外す（アウト）といった意味で覚えておいてください。

```
1 void setup() {
2     size(400, 400);
3 }
4 void draw() {
5     background(255, 255, 255); // 常に背景を白にする
6 }
7 void mouseDragged() {
8     background(0, 0, 0); // マウスを動かしている間のみ黒にする
9 }
```

なお、7 行目 `mouseDragged` の `D` は大文字で入力する必要があります。大文字の `D` を入力するには、**shift** キーを押しながら `D` キーを押します。

例 2

以下のプログラムは、ドラッグしている間マウスポインタの座標に合わせて線を引くというものです。実際に入力して動かしてみてください。

```
1 void setup() {
2     size(400, 400);
3     background(255, 255, 255);
4 }
5 void draw() {
6 }
7 void mouseDragged() {
8     line(pmouseX, pmouseY, mouseX, mouseY);
9 }
```

Processing にはマウスの位置に点を打つ `point` という命令がありますが、10 行目ではその `point()` ではなく `line()` を使っています。これは、`point` 命令では滑らかな線を引くことが出来ないためです。

ここで `line()` の使い方を思い出してみましょう。前半に代入されているのは始点の x 座標、次に始点の y 座標です。後半に代入されているのは終点の x 座標、次に終点の y 座標です。すなわち、`pmouseX`、`pmouseY` は前のマウスの x, y 座標を保存し、`mouseX`、`mouseY` は現在のマウスの x, y 座標を保存しています。そのため、切れ目のない線を描くことが出来るのです。

また、`draw()` の中身は何も書いていませんが、`draw()` 自体をきちんとプログラムに書かないと臨む結果になりません。先述のとおり、`mouseDragged()` は `draw()` が呼び出されることによって実行されるためです。大事なポイントですので押さえておきましょう。

STEP3 条件分岐による処理をしよう

プログラミングをしていると、「条件 A が成り立てば X を実行し、成り立たなければ Y を実行する」というような、条件によって動作を変えたい場面に遭遇します。今回の計算機室体験で使用している Processing やその他の多くの言語では、この条件分岐を行うために `if` というキーワードを用います。先ほど説明したような条件分岐を行うためには、以下のように書きます。

```
1 if (条件 A) {  
2   X の処理  
3 } else {  
4   Y の処理  
5 }
```

X, Y の処理はいつもと同じように命令を書きますが、A の部分には条件式と呼ばれるものを書く必要があります。条件式は表 1 のようなもので、S や T の部分には変数や計算式、数値などを書くことが出来ます。

表 1 代表的な条件式

条件式	意味	条件式	意味
<code>S == T</code>	S と T が等しい	<code>S >= T</code>	S は T 以上である
<code>S != T</code>	S と T が異なる	<code>S < T</code>	S は T より小さい
<code>S <= T</code>	S は T 以下である	<code>S > T</code>	S は T より大きい

ステップ 2 で使った `mouseX`, `mouseY` で簡単な条件分岐をしてみましょう。`mouseX` が 200 より小さいとき、すなわちキャンパス中央より左にマウスのポインタ（矢印）がある場合に処理 X を実行するようなプログラムを考えると、次のように書くことが出来ます。

```
1 if (mouseX < 200) {  
2   stroke(255, 0, 0); // 線の色を赤にする  
3 }
```

このプログラムでは `else { Y の処理 }` の部分を書いていません。実は、Y の処理が必要ない場合には `else { }` の部分を省略できるのです。

処理 X は `mouseX` が 200 未満のときに線の色を赤にするという処理でしたが、逆に `mouseX` が 200 以上のときに線の色を青にする処理 Y を加える場合、次のように `else { Y の処理 }` を追加することになります。

```
1 if (mouseX < 200) {  
2   // マウスポインタがキャンパス中央より左にあるとき  
3   stroke(255, 0, 0); // 線の色を赤にする  
4 } else {  
5   // 上記以外（キャンパス中央より右）のとき  
6   stroke(0, 0, 255); // 線の色を青にする  
7 }
```


では、ここで次の練習問題に挑戦してみましょう。

練習問題

`mouseX` の他に `mouseY` を用いることでキャンバスを 4 分割するように条件式を指定し、それぞれの条件で出力される線の色を変えてみよう。

■ヒント `mouseX` はキャンバス x 座標、`mouseY` はキャンバス y 座標に関する変数です。そして、`size()` で指定されたキャンバスの大きさは 400 ですから、200 がキャンバスの中央を示していることになります。

なお、次のようにすると「条件 A も条件 B も成り立つ (A かつ B)」や「条件 C か条件 D のうち少なくともどちらかが成り立つ (C または D)」という条件式を書くことができます。

```
1 if ( 条件 A && 条件 B ) { // A かつ B
2   X の処理
3 } else if ( 条件 C || 条件 D ) { // C または D
4   Y の処理
5 } else {
6   Z の処理
7 }
```

STEP4 キーボード入力を利用しよう

条件分岐が出来るようになったので、条件の幅を増やすためにキーボードからの入力によって処理を変えてみましょう。キーボード入力についての変数やイベントを以下で紹介します。プログラミングでの変数とは、値を保存しておくための文字のことを指します。

主な変数

key

押されているキーの文字を取得します。例えば、キーボードの C を押しているとき `key` の値は 'c' になります。

keyPressed

キーボードのキーが押されているときに値が `true`（条件式が成り立っている状況・^{しん}真）になります。逆に押されていないときの値は `false`（条件式が成り立っていない状態・^{まぎ}偽）になります。この変数は次のように使います。

```
1 if (keyPressed) {  
2   X の処理  
3 }
```

`if` では条件式が `true` のときに中身が実行されるため、`keyPressed` を使うと何らかのキーを押している間に処理 X が実行されるということになります。

主なイベント

keyPressed()

キーボードのキーが押されたときに呼び出されるイベントです。何らかのキーを押すことで中身を実行してくれます。

例

キーボードの C キーを押したときに画面を白で塗りつぶすプログラムを書いてみましょう。STEP2 で作成した絵を描くプログラムに次のコードを付け足します。

```
1 void keyPressed () {  
2   if (key == 'c') { // キーボードのCキーが押されたとき  
3     background(255, 255, 255);  
4   }  
5 }
```

`key == 'c'` は変数 `key` が `c` という文字に等しいという条件式を表しています。したがって、押されたキーが C であるときだけ `background(255, 255, 255);` で画面を塗りつぶすように条件分岐します。

先ほど変数について触れましたが、Processing には変数の宣言が必要なものと不要なものがあります。少しだけ頭に入れておきましょう。

宣言が必要なもの

変数には型というものがあります（表 2）。変数はしばしばプログラムの最初に宣言されて用いられます。宣言された変数のみプログラム内で扱うことが出来ます。

表 2 代表的な型

型	意味	例
<code>int</code>	整数である	<code>2019, -3</code>
<code>float</code>	小数点以下を含む	<code>3.14, -0.1</code>
<code>String</code>	文字列である	<code>"Tsukuba", "coins"</code>
<code>boolean</code>	真偽を問う	<code>true, false</code>

宣言が不要なもの

この STEP の冒頭で紹介した `key` や `keyPressed`, 他にも `width` (幅) や `height` (高さ) は Processing があらかじめ宣言していた変数です。このため、これらの変数は自分で宣言することなく使うことが出来ます。なお、あらかじめ宣言されていた変数にも型があります。どの型に当てはまるか、表 2 を参考にして考えてみましょう。

では、ここで練習問題に挑戦してみましょう。

練習問題

A キーが押されているとき、円が左から右に進むプログラムを作成してみよう。次のプログラムを写し、`if` の条件式のコメントアウトされている部分を変えて実行してみましょう。

■ヒント 変数 `keyPressed` と `key` を上手に使ってみよう。

```
1  /* 変数の宣言 */
2  float x = 0.0;
3  float speed = 1.0;
4  /* イベントの宣言 */
5  void setup() {
6      size(400, 400);
7  }
8  void draw() {
9      background(200, 200, 200);
10     if (/* この部分の条件式を考えよう */) {
11         x += speed;
12     }
13     ellipse(x, height/2, 60, 60);
14 }
```

STEP5 複雑なプログラムを書いてみよう（発展）

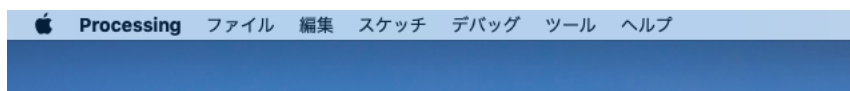
ここまでの STEP で、命令・イベント（マウス入力）・条件分岐・キーボード入力を学びました。様々な命令やイベントを紹介しましたが、Processing には他にも様々なリファレンス（命令やイベント）があります。新しい変数やイベントを用いたり、それらを組み合わせることで、より複雑な処理をするプログラムを書くことが出来ます。

では早速ですが、ここまで学んできた知識を生かして、以下の問題に挑戦してみてください。新しく追加されている変数やイベントがあるため、これらを確認しながらプログラムを完成させてみましょう。

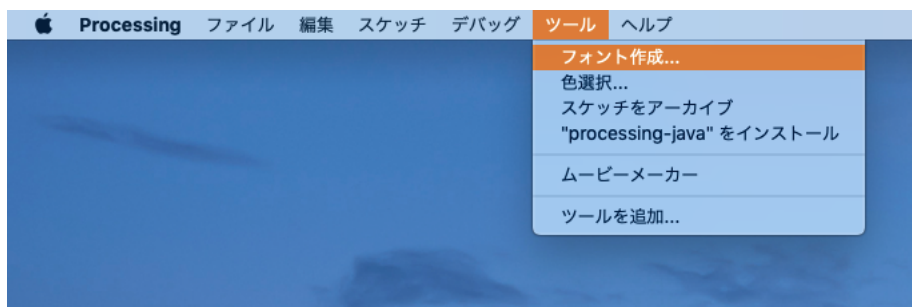
問題 1 入力文字の表示

ここでは、キーボード入力によってキャンバス内に好きな文字を表示するプログラムを作成します。好きな文字を表示させるためには**フォント**の知識が必要になります。まずはフォントファイルの作成をしましょう。

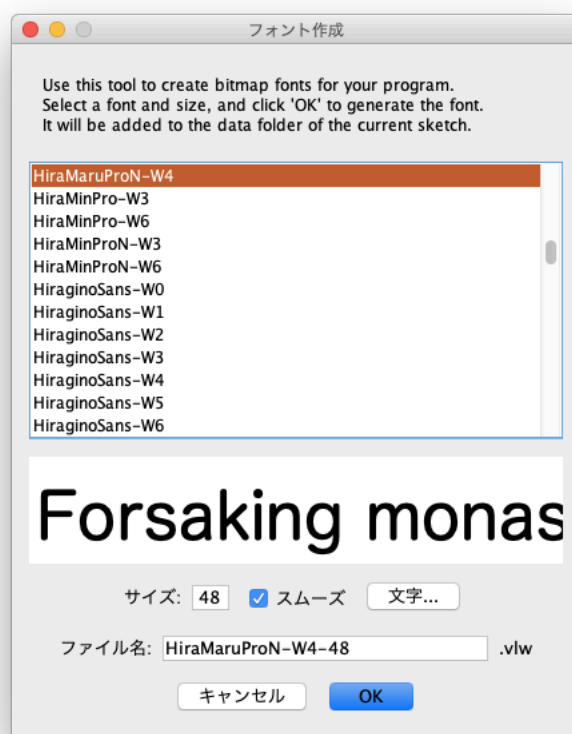
Processing のウィンドウが一番前になっている状態にすると、ディスプレイの一番上に見える**タスクバー**が以下のように表示されるようになります。



この状態でタスクバーの「ツール」をクリックすると以下のようにメニューが現れるので、その中の「フォント作成...」をクリックします。



すると以下のようなウィンドウが表示されます。フォントの種類の一覧の画面です。適当なフォントの名前をクリックすると下の方にプレビューが表示されるので、好きなフォントを探してみてください。



フォントが決まったら、「OK」をクリックする前にファイル名をコピーしておきましょう。プログラム内で指定するのにファイル名が間違っていると正しく読み込んでくれません。ファイル名をコピーするには、以下のファイル名の欄をクリックして ⌘ + A を押し、そのあと ⌘ + C キーを押します。コピーしたものを貼り付けるときは ⌘ + V キーを押します。

なお、「⌘」は「command」と書いてあるキー（コマンドキー）のことです。また「⌘ + A」は「コマンドキーを押しながら A キーを押す」という意味です。



ここまでの方法で作成したフォントファイル（.vlw）をプログラム内で指定することで、そのフォントを使用することができます。(1) ～ (3) の部分を各自で書き換えて動かしてみましょう。

```
1  /* 変数の宣言 */
2  int x = 0;
3  int y = 50;
4  PFont font;
5
6  /* イベントの宣言 */
7  void setup() {
8      size(400, 400);
9      background(0, 0, 0);
10     fill(255, 128); // オブジェクトを塗りつぶす命令
11
12     font = loadFont( "(1) 作成したフォントファイル名を入力しよう.vlw");
13     textFont(font, 48);
14 }
15
16 void draw() {
17 }
18
19 void keyPressed() {
20     (2) text という命令を用いて、ここで宣言してみよう
21 }
22
23 void keyReleased() {
24     x += 35;
25     if ( (3) 条件式 A) { // 文字がキャンバス外に出ってしまったとき
26         x = 0;
27         y += 50;
28     }
29 }
```

問題2 3D オブジェクトの作成

Processing では、二次元モデルだけではなく三次元モデルについても作成して動かすことができます。X キー・Y キー・Z キーを押すことで回転している箱の大きさを変えるプログラムを作成してみましょう。

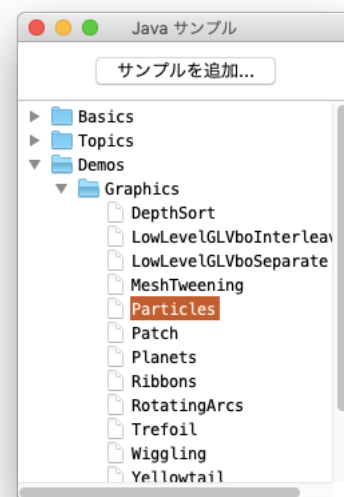
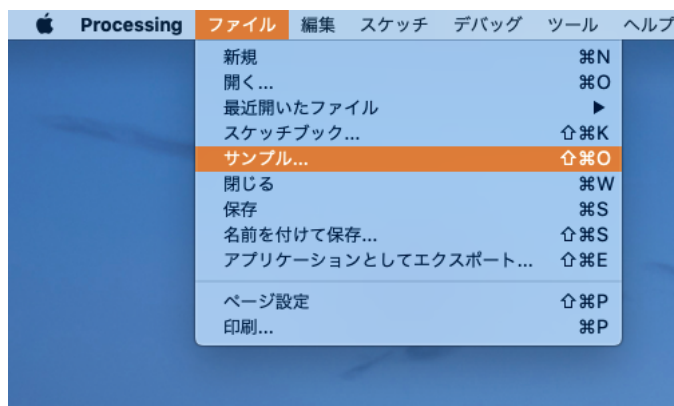
変数やイベントが多いため、それぞれの役割を確認しておくことでエラーや間違いを防ぐことができます。

```
1  /* 変数の宣言 */
2  // それぞれの軸に関して箱を変化させるための変数
3  int x = 0;
4  int y = 0;
5  int z = 0;
6
7  // それぞれの軸に関して箱をどのくらいの速度で変化させるか決定する
   変数
8  int speedX = 5;
9  int speedY = 5;
10 int speedZ = 5;
11
12 int angle = 0; // 観測する角度を決定する変数
13
14 /* イベントの宣言 */
15 void setup() {
16     size(400, 400, P3D);
17 }
18 void draw() {
19     background(0, 0, 0);
20     translate(width/2, height/2); // 立体の中心を画面中央に移動
21     rotateX(radians(angle)); // x 軸について angle 度回転する
22     // rotateY(radians(angle));
23     // rotateZ(radians(angle));
24     box(150+x, 150+y, 150+z);
25     angle += 3;
26     if (angle >= 360) angle = 0;
27     delay(25); // ループを遅らせるための命令, 中の数字はミリ秒 (1000
                   = 1 秒)
28 }
29 void keyPressed() {
30     if ( (1) 条件 A ) {
31         (2) x についての処理
32     }
33     if ( (3) 条件 B ) {
34         (4) y についての処理
35     }
36     if ( (5) 条件 C ) {
37         (6) z についての処理
38     }
39 }
```

STEP6 サンプルプログラムを使ってみよう（補足）

Processing にはサンプルのプログラムが用意されています。サンプルプログラムには様々なものがあります。色々試してみると Processing で出来ることが意外と多いことに気づくでしょう。

タスクバーの「ファイル」→「サンプル...」を覗いてみましょう。



以下は例としてサンプル「Particles」を実行してみた結果です。

