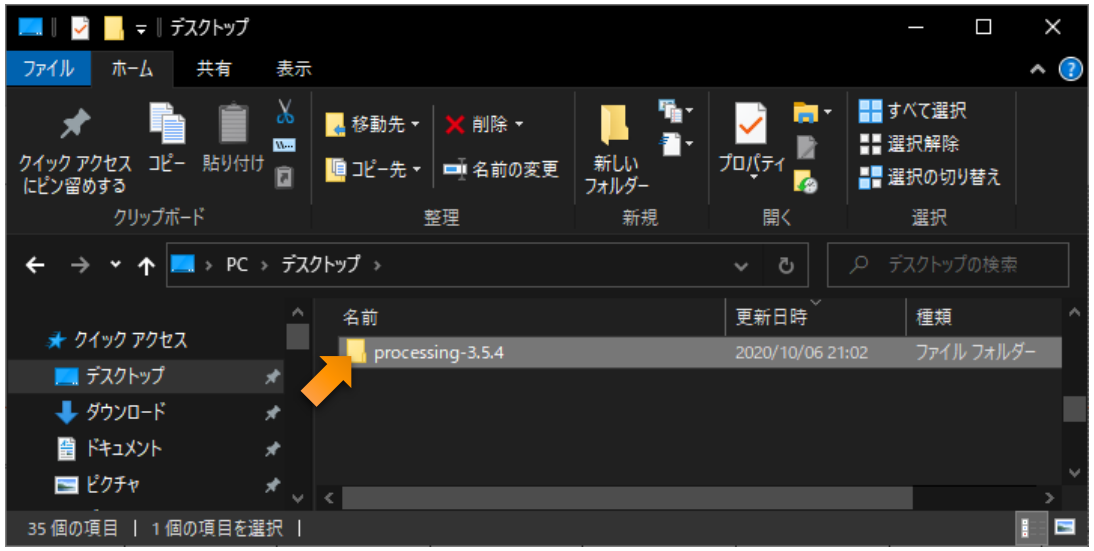
メディアデザインメジャー
メジャー体験演習

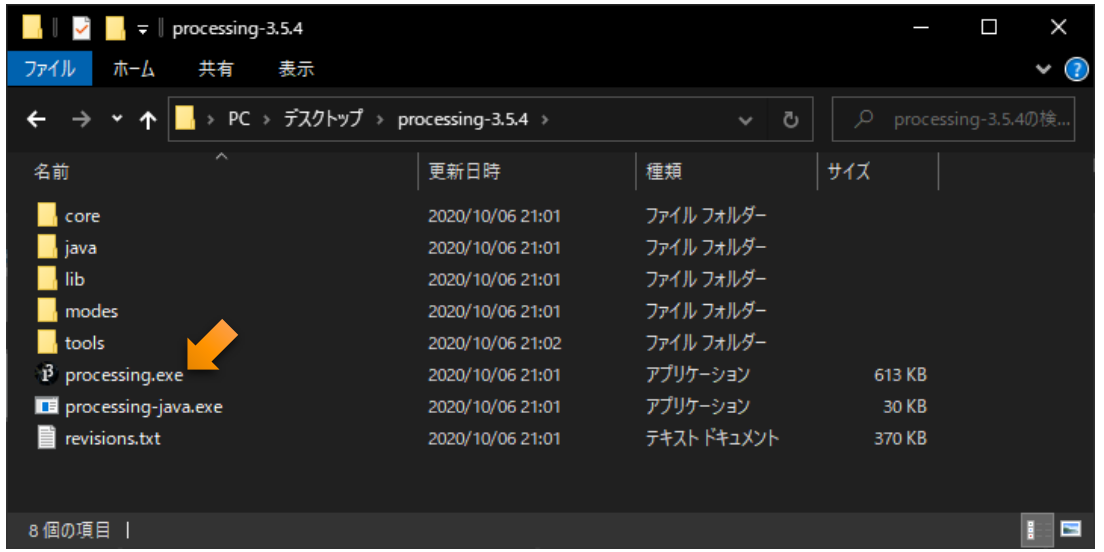
ビジュアルグループ２日目

Processing を起動する

フォルダを開く

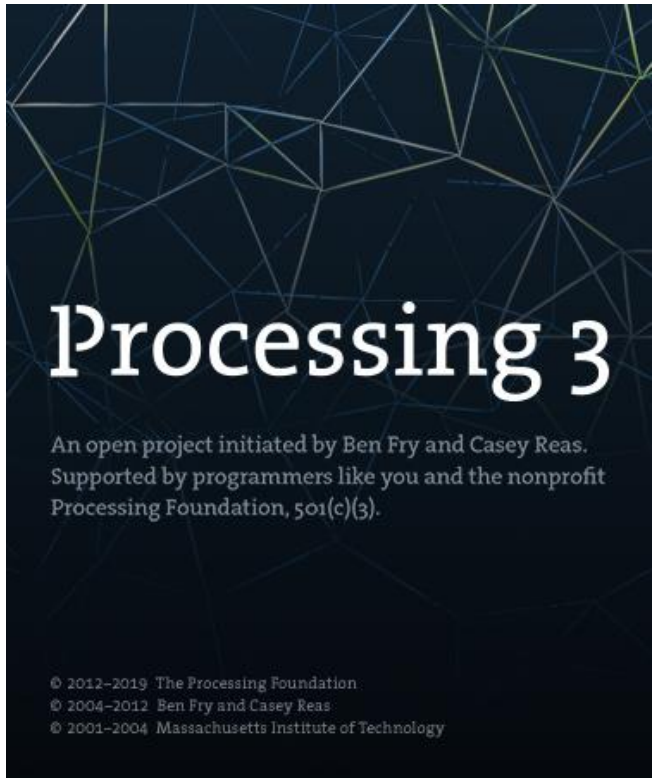


processing.exe をダブルクリックする

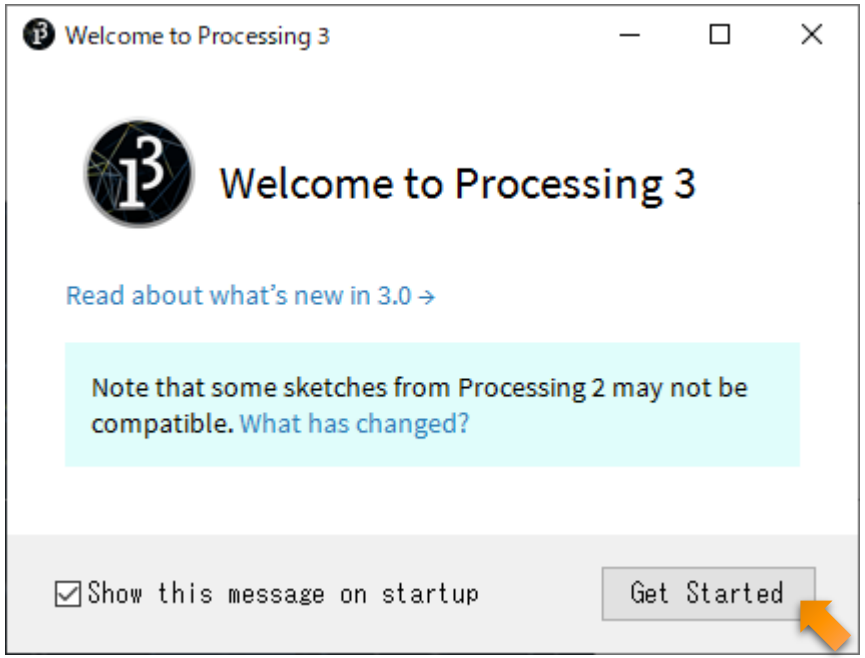


起動中

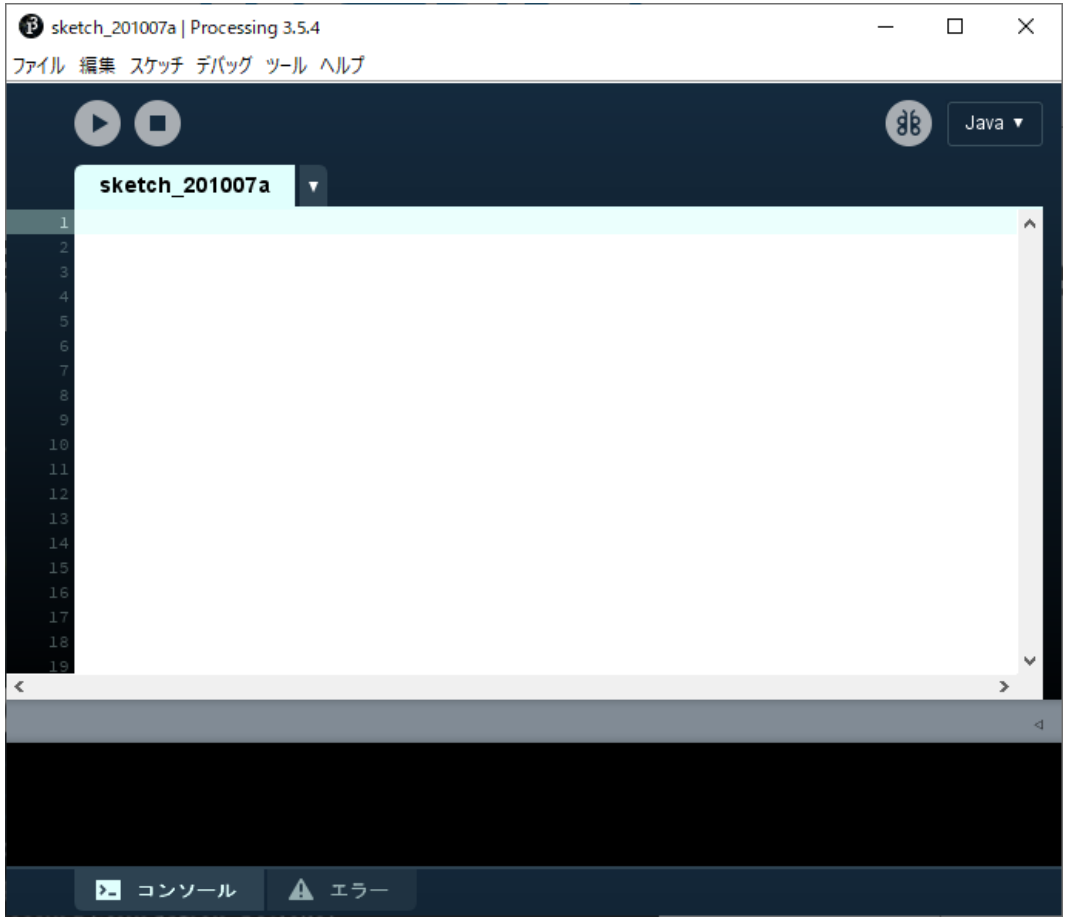
しばらく待つ



Get Started をクリックする



Processing のウィンドウ



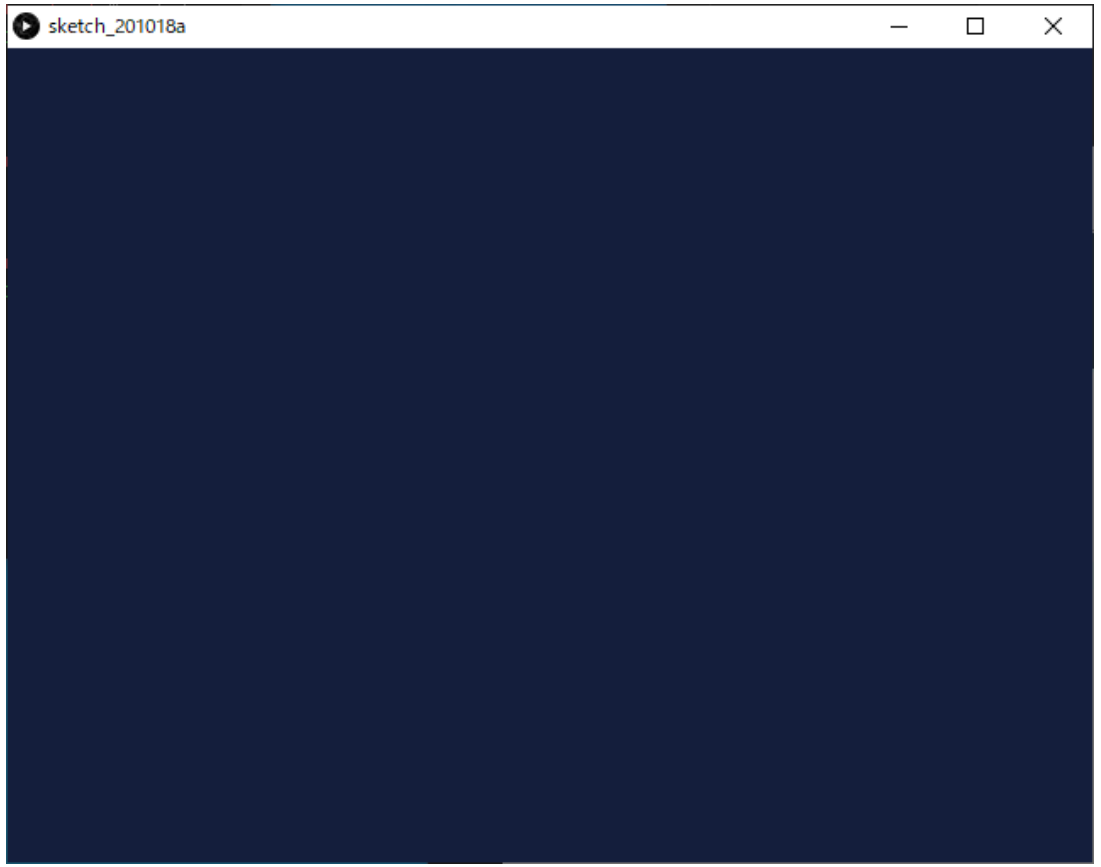
プログラムで3DCGを描く

“Processing was designed to be a flexible software sketchbook.”

(Processing のリファレンスマニュアル https://processing.org/reference/ より)

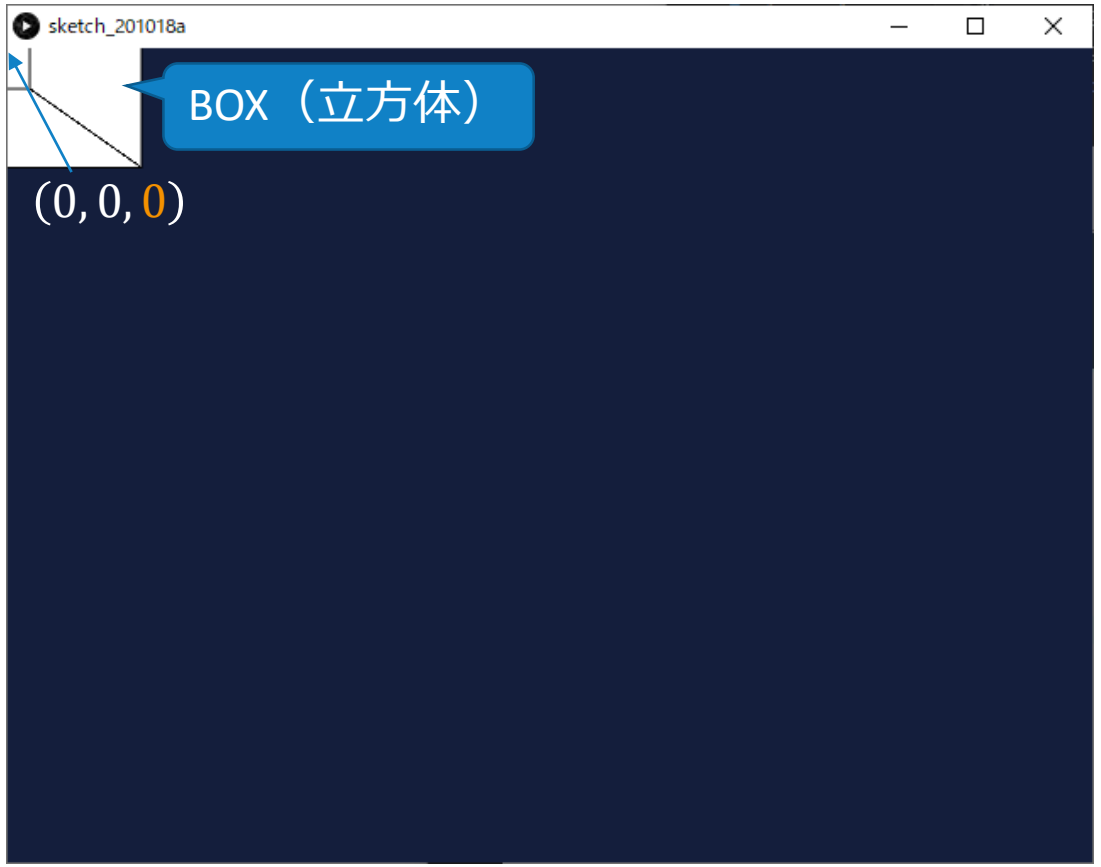
スケッチを３次元にする

```
void setup() {
  size(640, 480, P3D);
}

void draw() {
  background(20, 30, 60);
}
```



箱を一つ作って描く

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  shape(box);
}
```

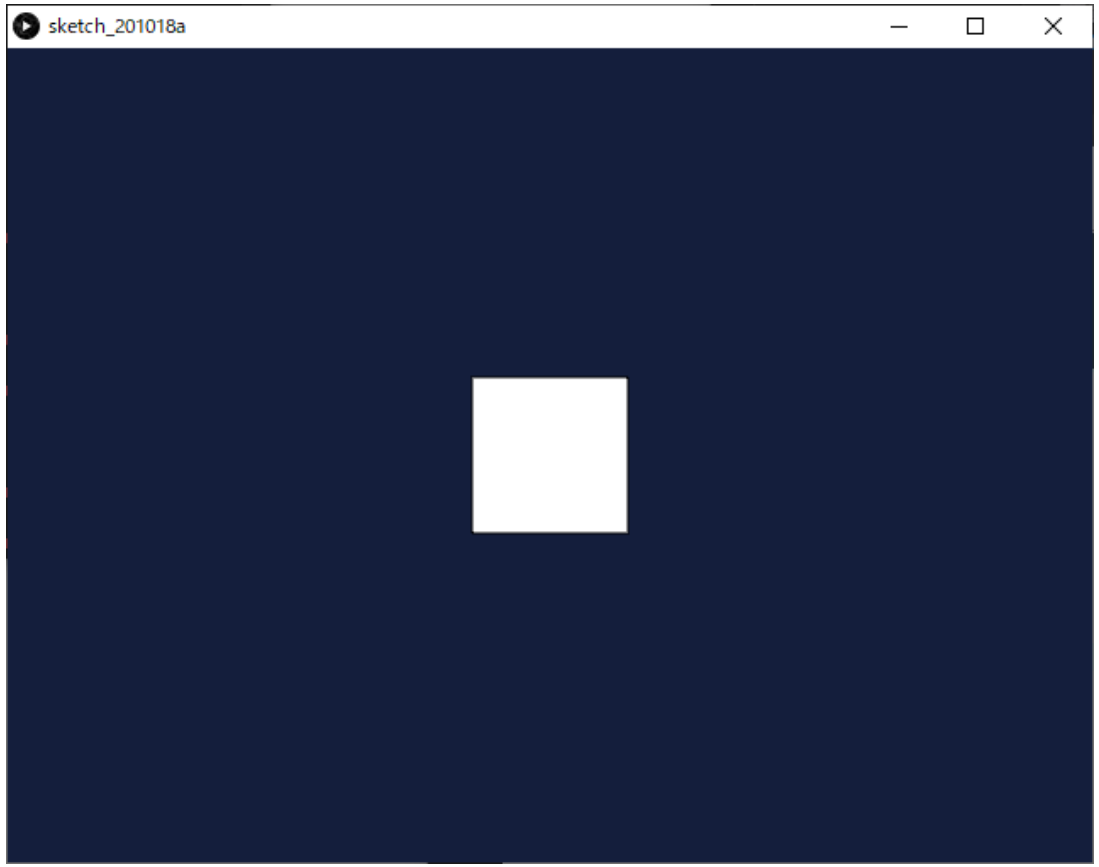


PShape

- PShape s;
  - 図形のクラスの変数 s を作る
- s = createShape(kind, p)
  - kind の図形のオブジェクトを一つ作って s に保持させる
  - kind : POINT, LINE, TRIANGLE, QUAD, RECT, ELLIPSE, ARC, BOX, SPHERE
    - それぞれ point(), line(), triangle(), quad(), rect(), ellipse(), arc(), box(), sphere() に対応
  - p : パラメータのリスト
    - createShape(BOX, 100) : 100 は立方体の一片の長さ
- shape(s)
  - s が保持している図形のオブジェクトを描く

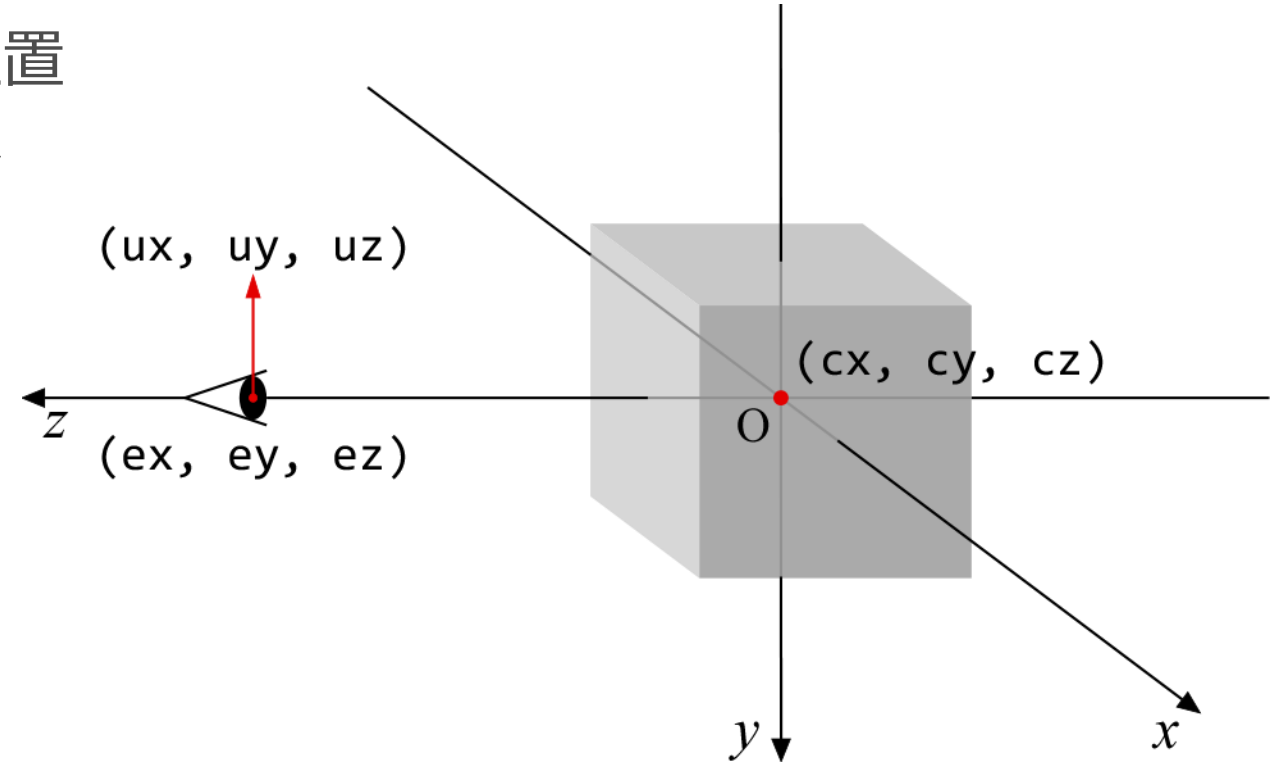
カメラを設定する

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  shape(box);
}
```



カメラの設定

- camera(ex, ey, ez, cx, cy, cz, ux, uy, uz)
  - ex, ey, ez : カメラ（視点）の位置
  - cx, cy, cz : 目標（中心）の位置
  - ux, cy, uz: 上方向のベクトル

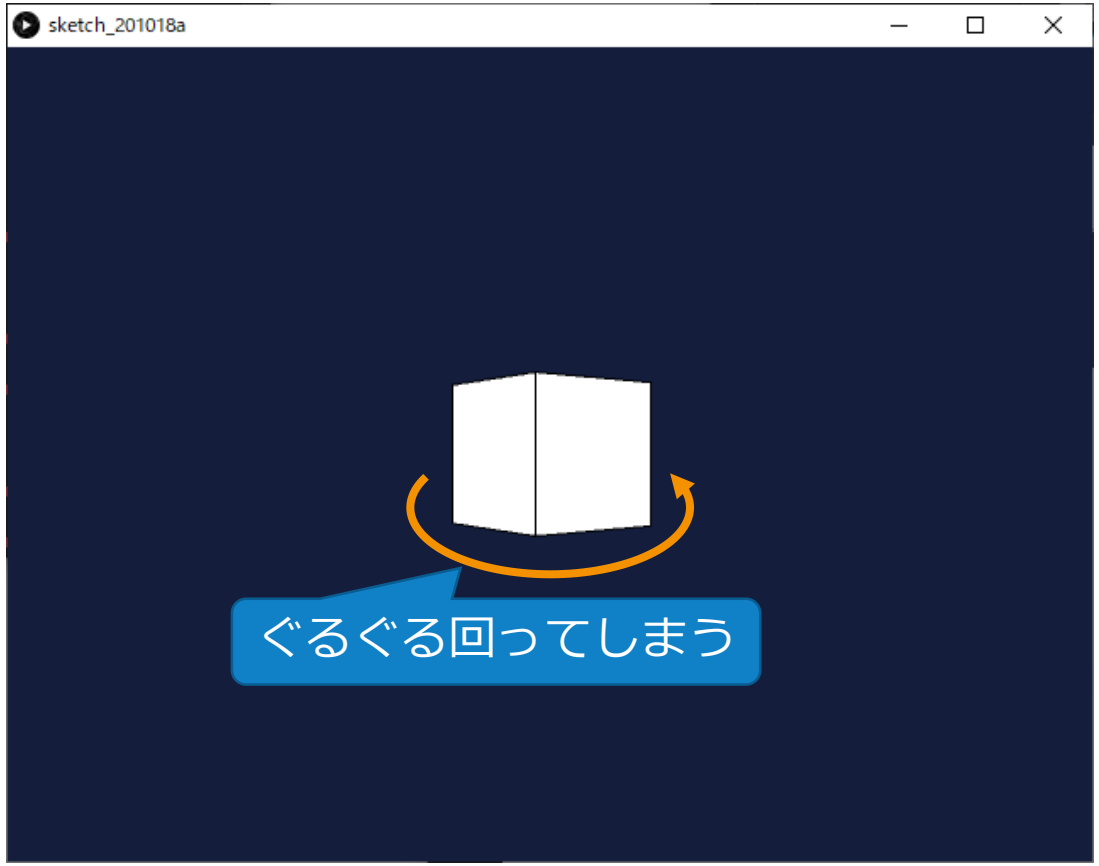


箱を回転する

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  box.rotateY(0.01);
  shape(box);
}
```

draw() は繰り返し実行される

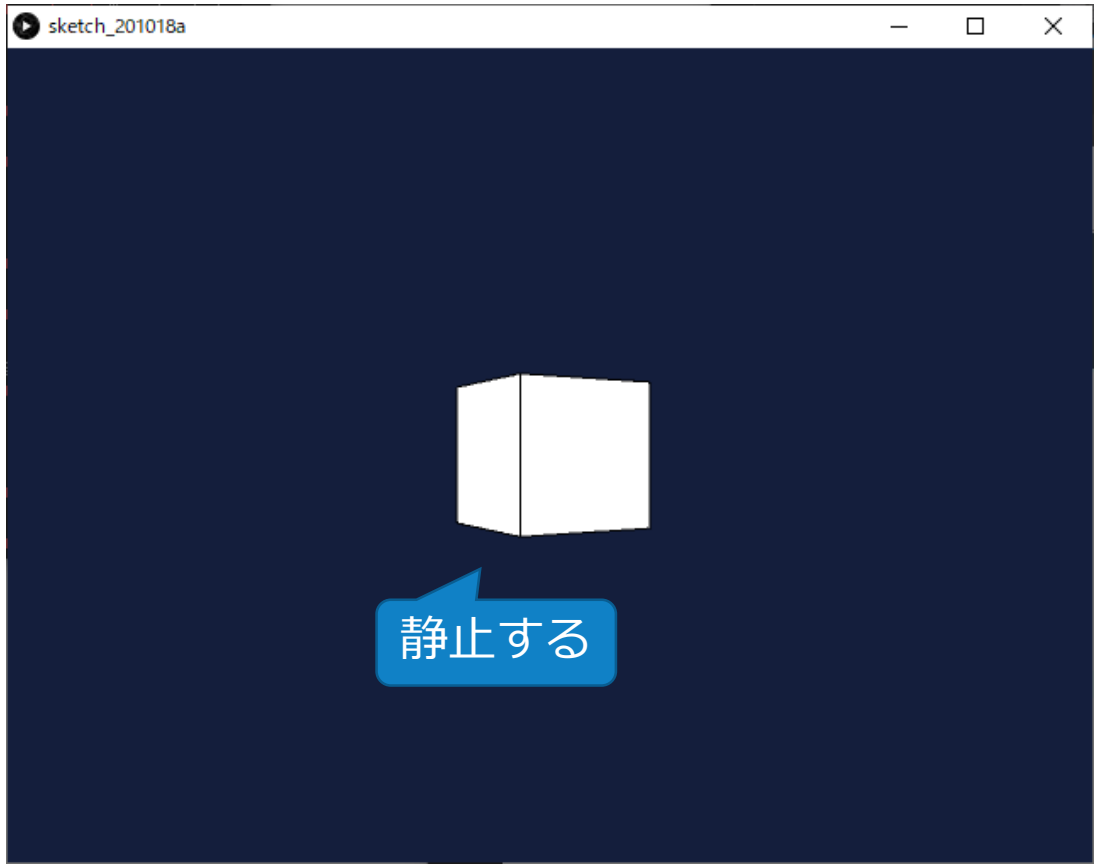
そのため回転角が累積される



毎回変換行列を単位行列で初期化する

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  box.resetMatrix();
  box.rotateY(PI / 6);
  shape(box);
}
```

PI は π、PI / 6 は π / 6 ⇒ 30°

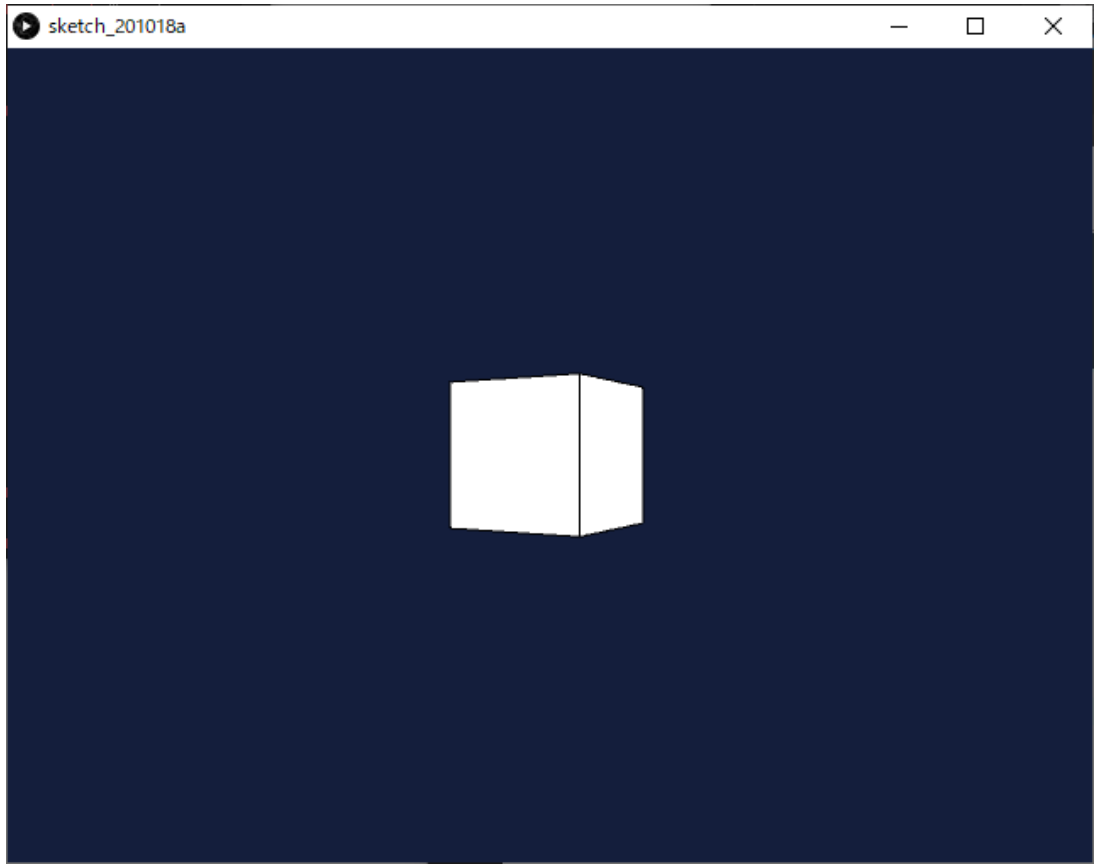


回転角を時刻で決める

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  float t = millis() * 0.001;
  box.resetMatrix();
  box.rotateY(PI * t);
  shape(box);
}
```

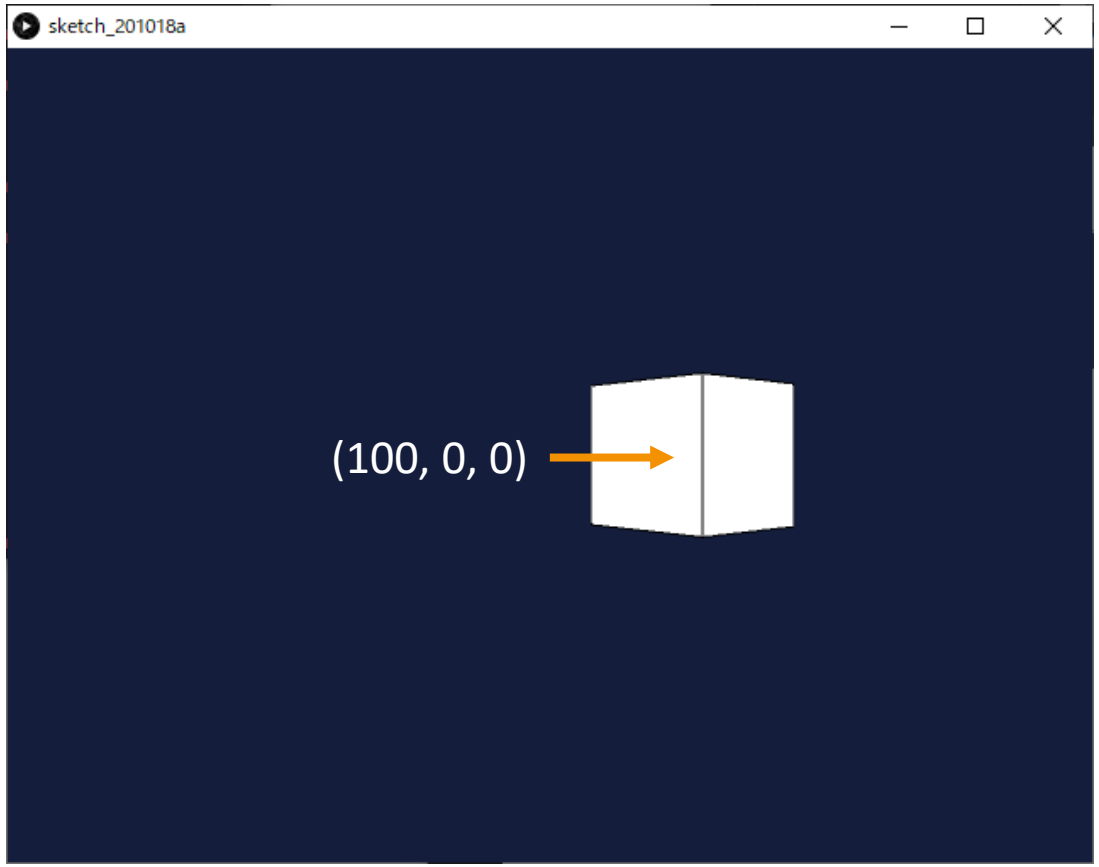
経過時間を秒に換算

1秒当たりπ回転する
（2秒で1回転）



回転後に平行移動する

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  float t = millis() * 0.001;
  box.resetMatrix();
  box.rotateY(PI * t);
  box.translate(100, 0, 0);
  shape(box);
}
```



座標変換

- PShape クラスのオブジェクト（以下では s）は自身の座標変換のための変換行列を保持している
- s.resetMatrix()
  - 変換行列を初期化（単位行列を設定）する
- s.translate(x, y, z)
  - x, y, z : s の現在位置からの平行移動量
- s.scale(x, y, z)
  - x, y, z : s の現在の大きさに対する X, Y, Z 軸方向の拡大縮小率
- s.rotateX(angle), s.rotateY(angle), s.rotateZ(angle)
  - angle : s の現在の姿勢からの、それぞれ X, Y, Z 軸中心の回転角

時間の計測

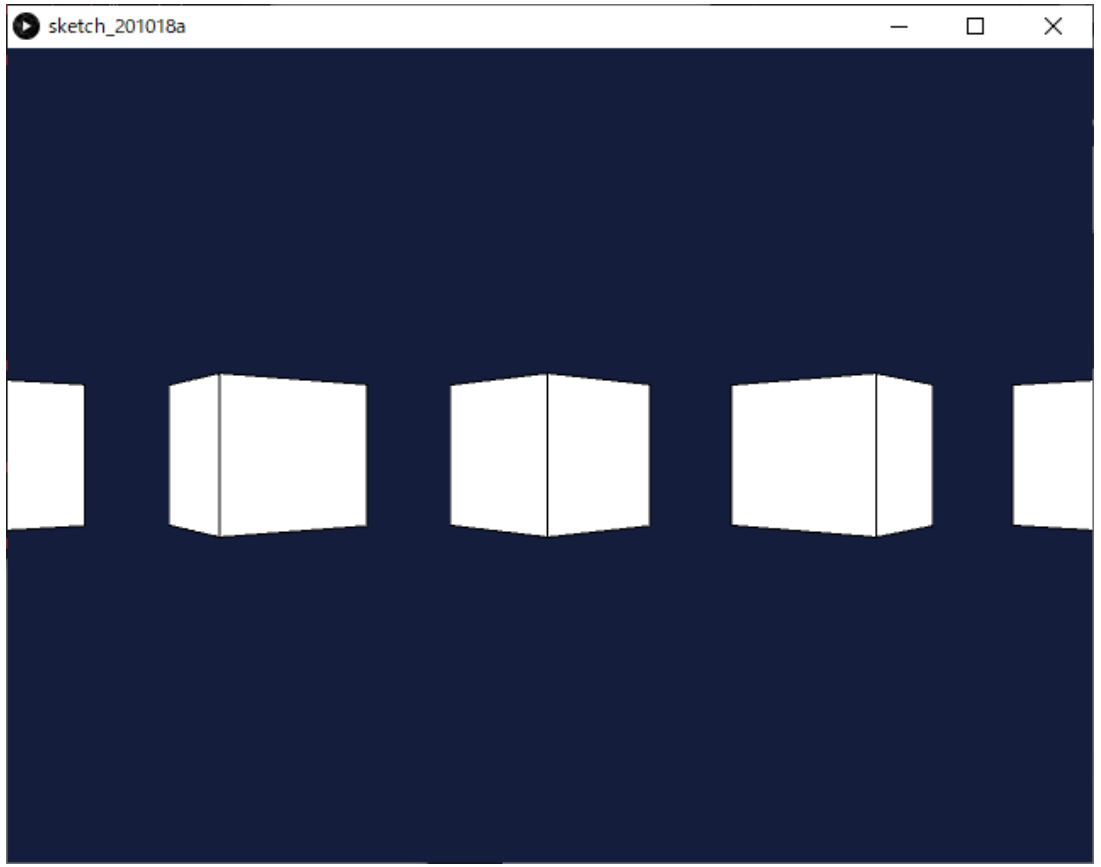
- millis()
  - プログラムの実行開始からの経過時間を単位ミリ秒の整数 (int) で返す
    - `float t = millis() * 0.001;  // 単位をミリ秒から秒に変換する`
- second(), minute(), hour(), day(), month(), year()
  - 現在時刻のそれぞれ秒、分、時、日、月、年を整数 (int) で返す

課題 5

複数の箱を横に並べて表示する

横に11個並べる

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
}

void draw() {
  background(20, 30, 60);
  camera(0, 0, 500, 0, 0, 0, 0, 1, 0);
  float t = millis() * 0.001;
  for (int x = -1000; x <= 1000; x += 200) {
    box.resetMatrix();
    box.rotateY(PI * t);
    box.translate(x, 0, 0);
    shape(box);
  }
}
```



位置を変えながら図形を描く

```
for (int x = -1000; x <= 1000; x += 200) {
```

- x = -1000 から始めて 200 おきに x = 1000 までを 100 回繰り返す

```
  box.resetMatrix();
```

- box が保持しているオブジェクトの変換行列を初期化する

```
  box.rotateY(PI * t);
```

- 変換行列に Y 軸中心に 1 秒あたり π 回転する回転の変換を乗じる

```
  box.translate(x, 0, 0);
```

- 変換行列に (x, 0, 0) に平行移動の変換行列を乗じる

```
  shape(box);
```

- box が保持しているオブジェクトを描画する

```
}
```

全部の箱が表示されるように設定する

- 11個の箱全部が表示されるようにカメラの位置を設定してください

スクリーンショットをアップロードする

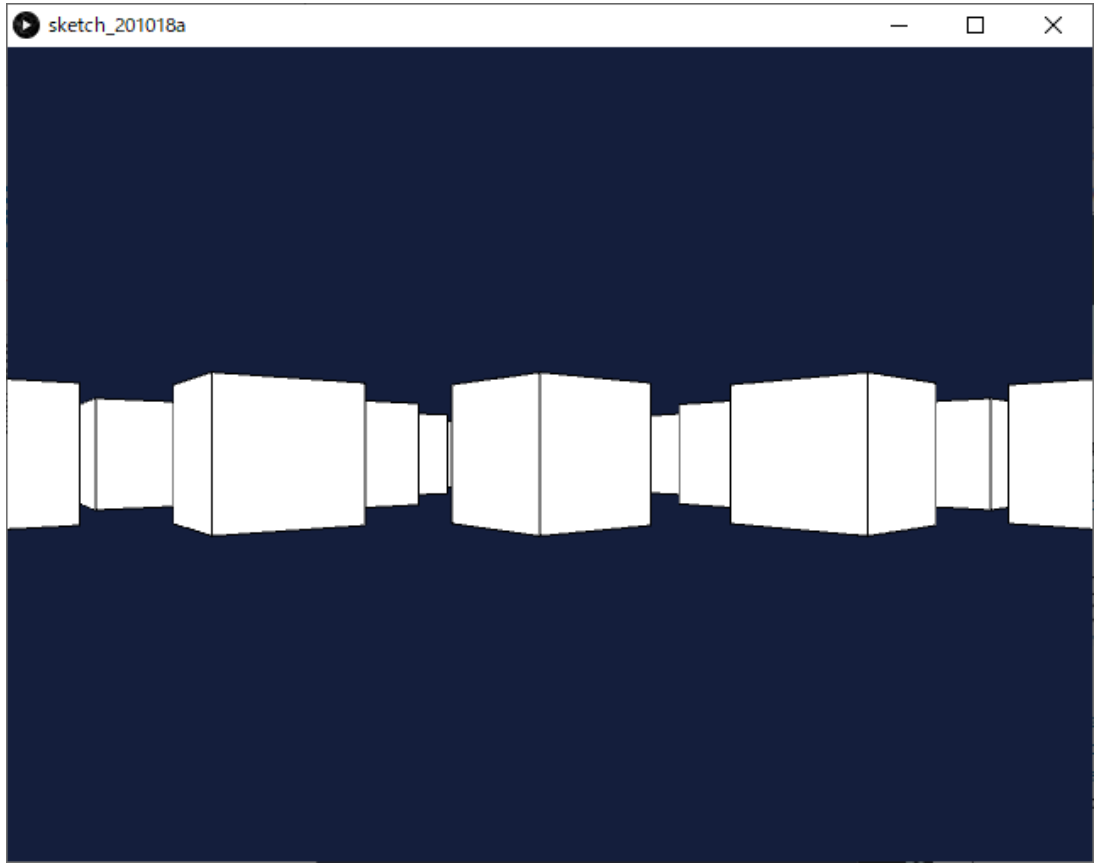
- 実行結果が表示されているウィンドウのスクリーンショットを **05.png** というファイル名で保存して、Moodle の「スクリーンショットのアップロード」からアップロードしてください。

課題 6

箱の列を前後に並べて表示する

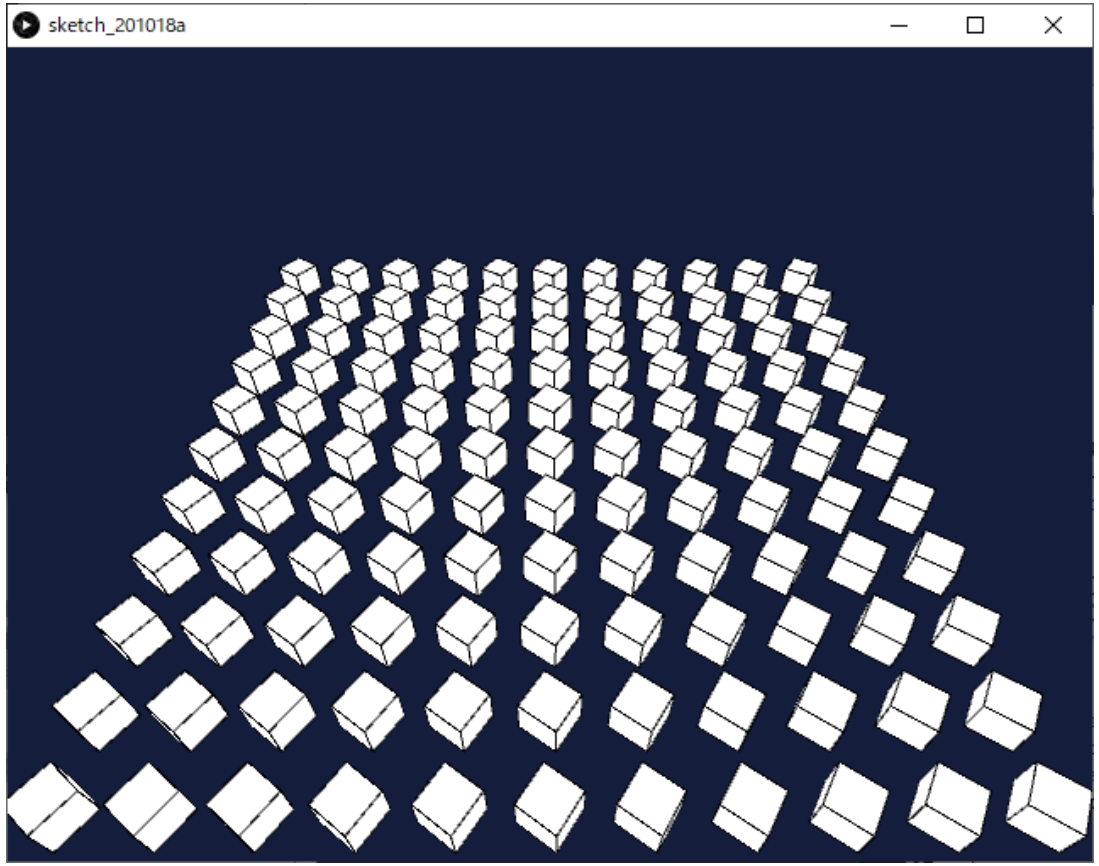
前後に11列並べる

```
PShape box;

// 途中省略

void draw() {
  // 途中省略
  float t = millis() * 0.001;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, 0, z);
      shape(box);
    }
  }
}
```

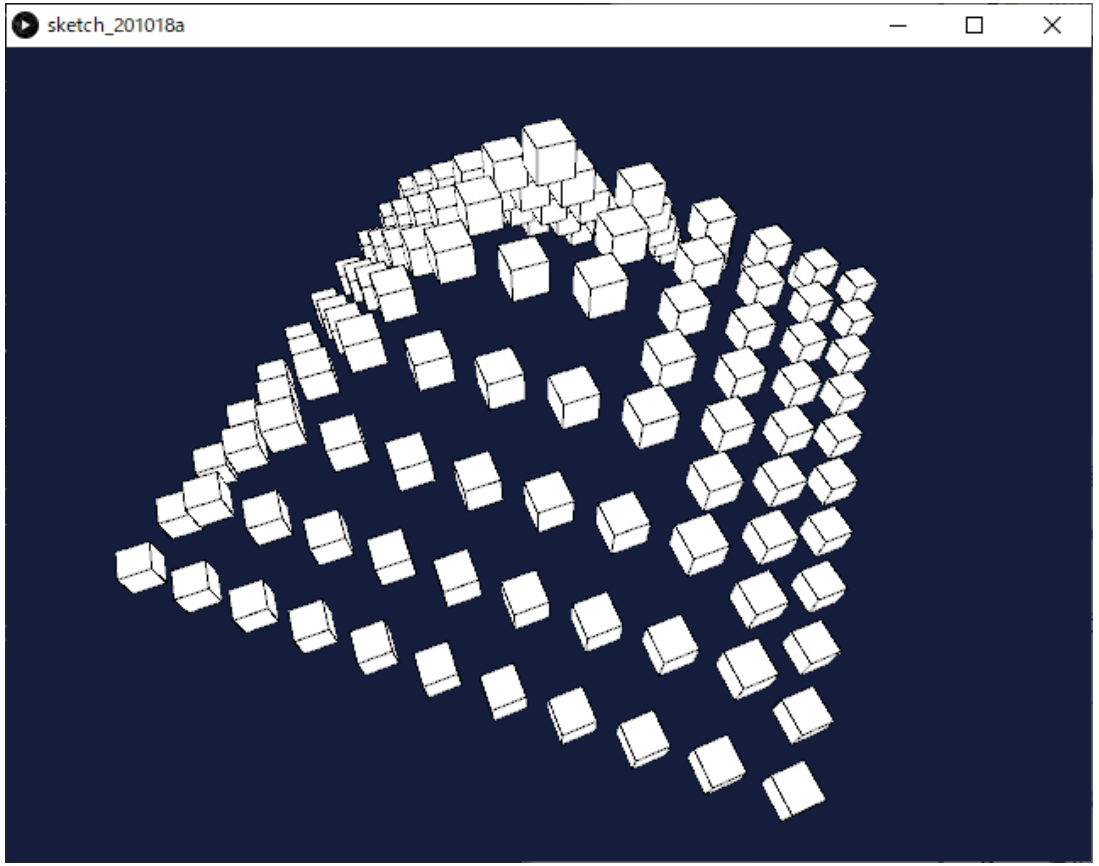


全部の箱が表示されるように設定する

- 11行11列の計121個の箱全部が表示されるようにカメラの位置や目標を設定してください
- 全体がはみ出ず小さすぎずに表示できれば視点や目標点の位置は自由です
- 余裕があれば箱の並べ方も変えてください（箱の数は 512 個以下）



並べ方を変えた例（努力課題）



スクリーンショットをアップロードする

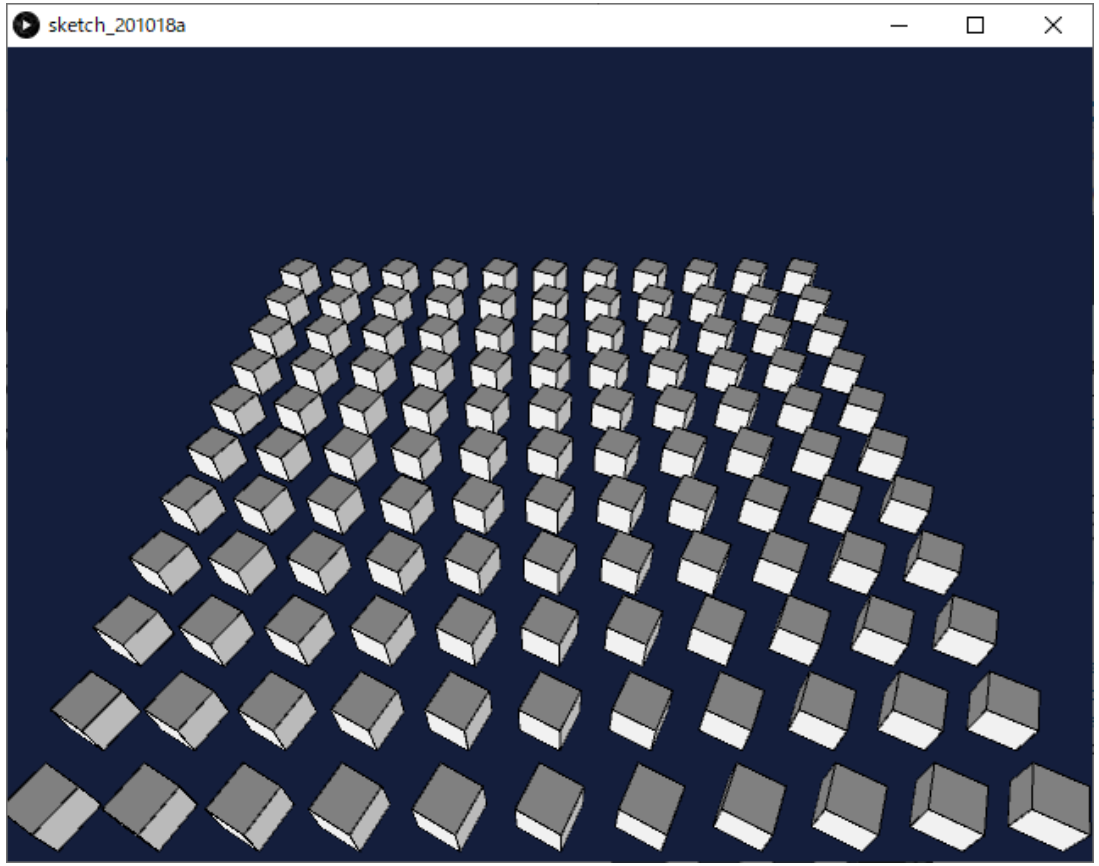
- 実行結果が表示されているウィンドウのスクリーンショットを **06.png** というファイル名で保存して、Moodle の「スクリーンショットのアップロード」からアップロードしてください。

陰影と色

箱に光を当てて色を付ける

陰影をつける

```
PShape box;

// 途中省略

void draw() {
  // 途中省略
  lights();
  float t = millis() * 0.001;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, 0, z);
      shape(box);
    }
  }
}
```

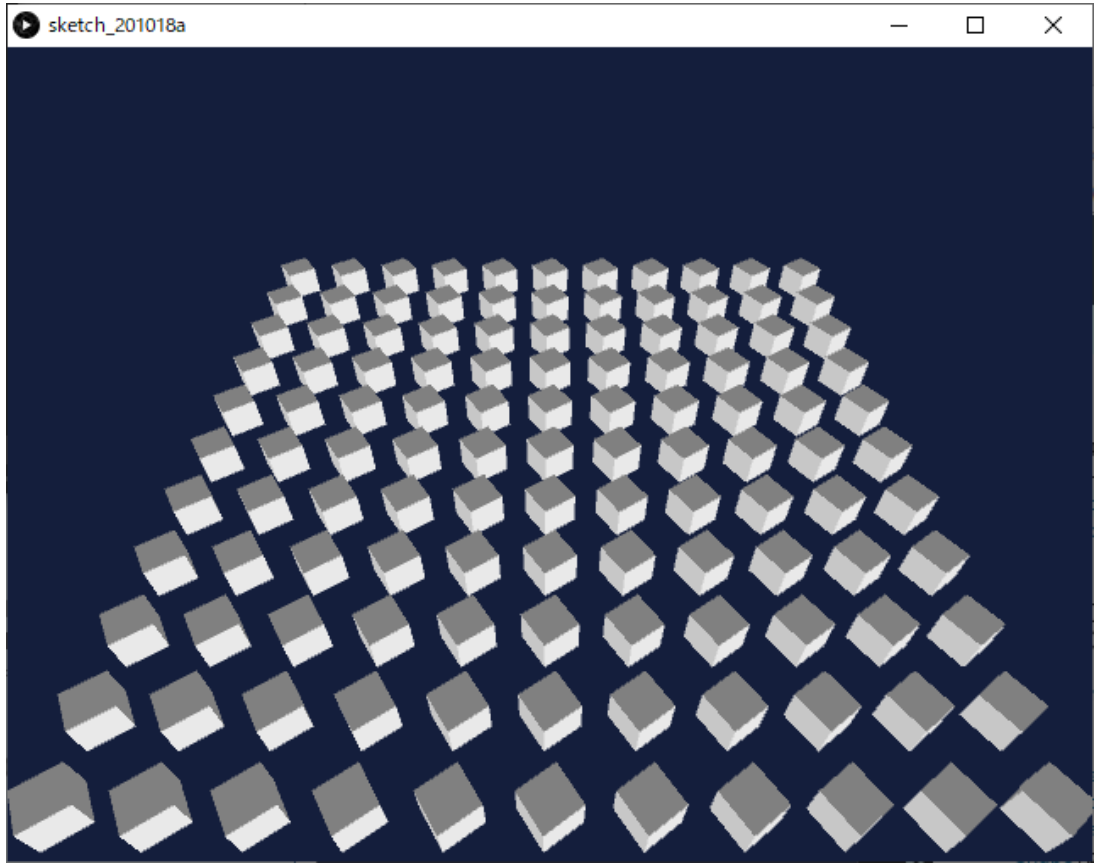


陰影付け処理

- lights()
  - デフォルトの環境光強度、平行光の設定、フォールオフの設定、光源強度の鏡面反射成分の設定を行います
    - デフォルト値は環境光強度 (128, 128, 128)、平行光源の強度 (128, 128, 128)、平行光源の方向 (0, 0, -1)、フォールオフ無し (1, 0, 0)、鏡面反射光なし (0, 0, 0)
    - この関数は draw() の中に置く必要があります
- ambientLight(r, g, b)
  - r, g, b : 環境光強度
- directionalLight(r, g, b, nx, ny, nz)
  - r, g, b : 平行光強度
  - nx, ny, nz : 平行光の方向

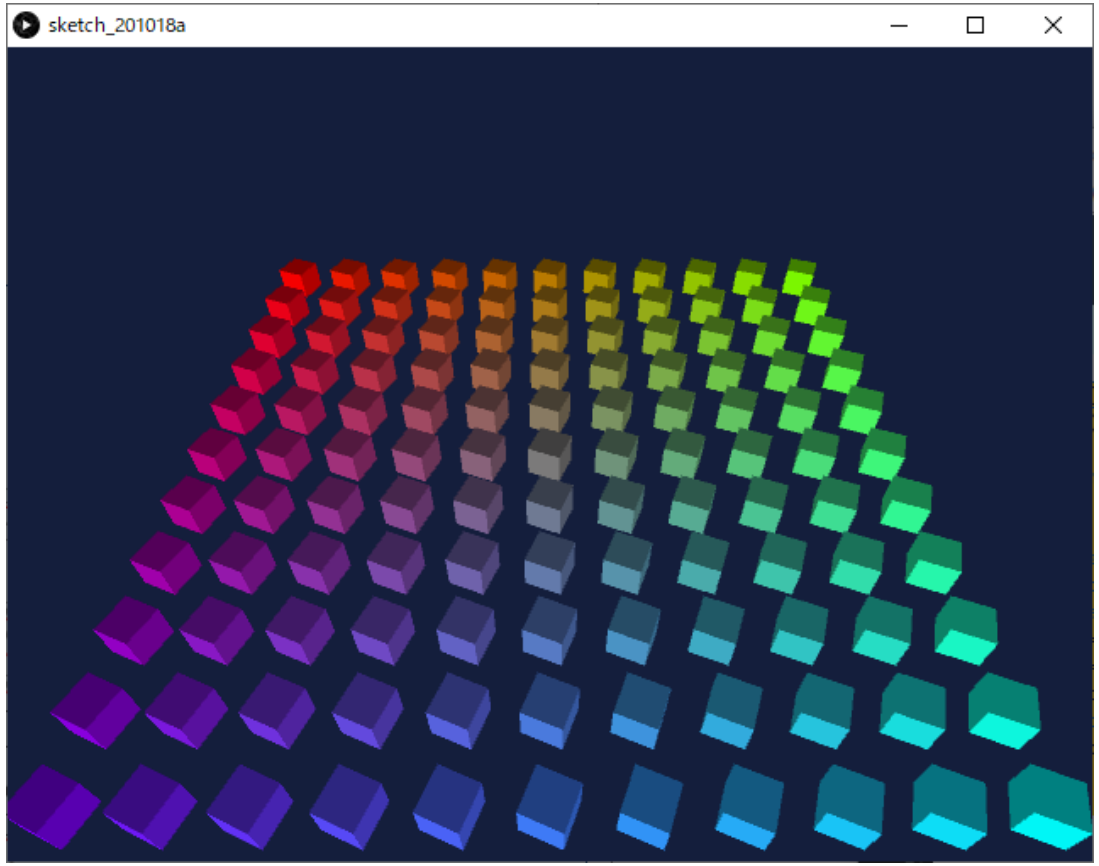
境界線を消す

```
PShape box;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
  box.setStroke(false);
}

// 以下省略
```



箱に色を付ける

```
// 途中省略

void draw() {
  // 途中省略
  lights();
  float t = millis() * 0.001;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, 0, z);
      int g = int((x + 1000) * 255 * 0.0005);
      int b = int((z + 1000) * 255 * 0.0005);
      color c = color(255 - (g + b) / 2, g, b);
      box.setFill(c);
      shape(box);
    }
  }
}
```



２次元グラデーションの作り方

```
int g = int((x + 1000) * 255 * 0.0005);
int b = int((z + 1000) * 255 * 0.0005);
```

- x, z は -1000 から 1000 まで 200 ずつ変化するから、1000 足せば 0→2000
- それに 255 × 0.0005 を掛ければ 0→255

```
color c = color(255 - (g + b) / 2, g, b);
```

- r = 255 - (g + b) / 2 なので g = b = 255 すなわちシアンのときだけ r = 0
- g 成分だけ、b 成分だけの色は出ない（シアン以外 r が 0 にならない）

PShape のその他の属性

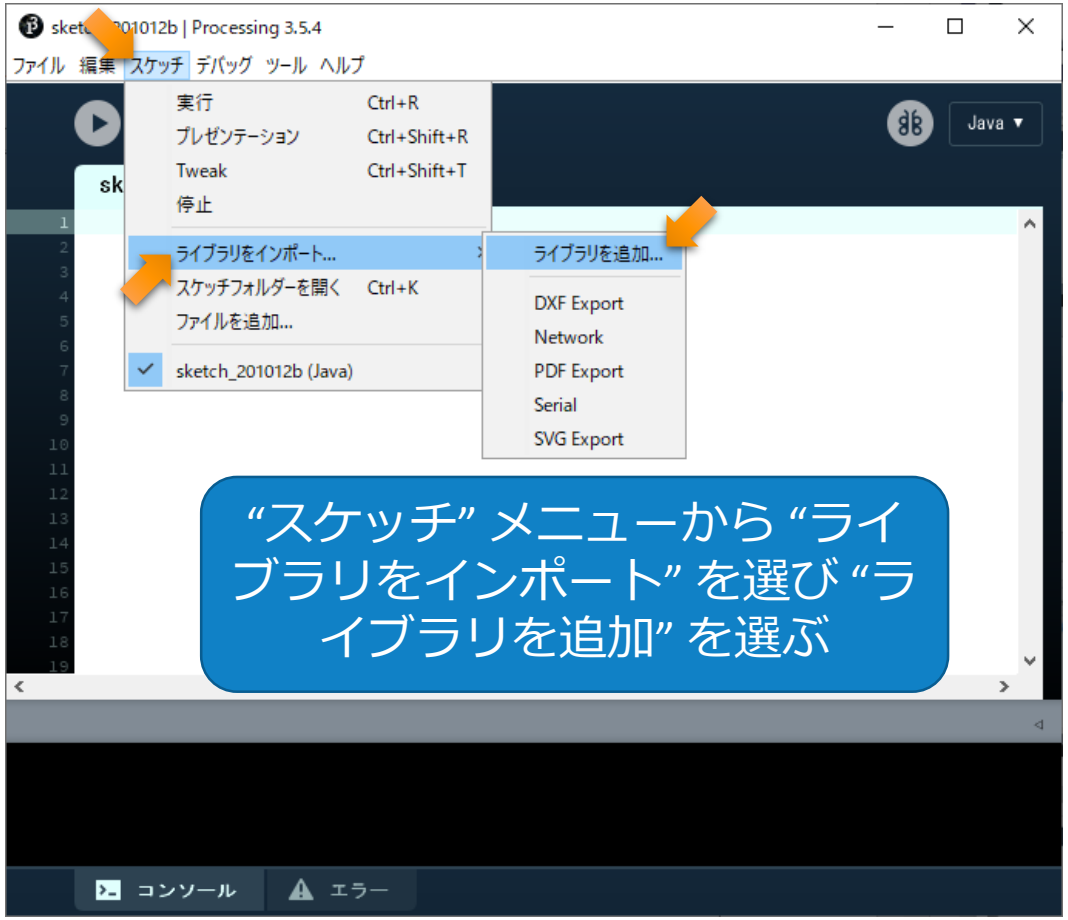
- PShape クラスのオブジェクト（以下では s）に対して
- s.setStroke(c)
  - c : color 型の線の色, false なら線を描かない
- s.setFill(c)
  - c : color 型の塗りつぶしの色, false なら塗りつぶさない

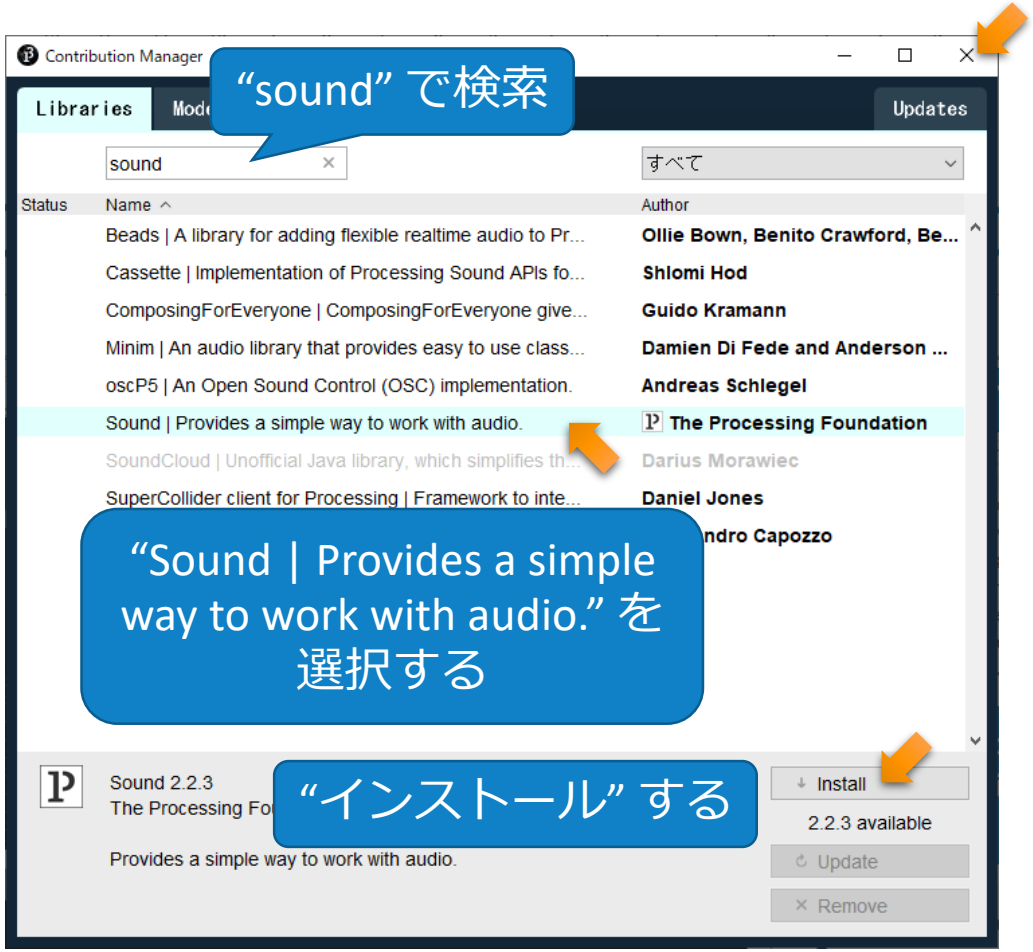
音声入力

サウンド入力ライブラリを追加した後

- パソコンにマイクが付いていれば「マイクから音声を入力する」に進む
- マイクが付いていなければ「音声ファイルをスケッチに入れる」に進む

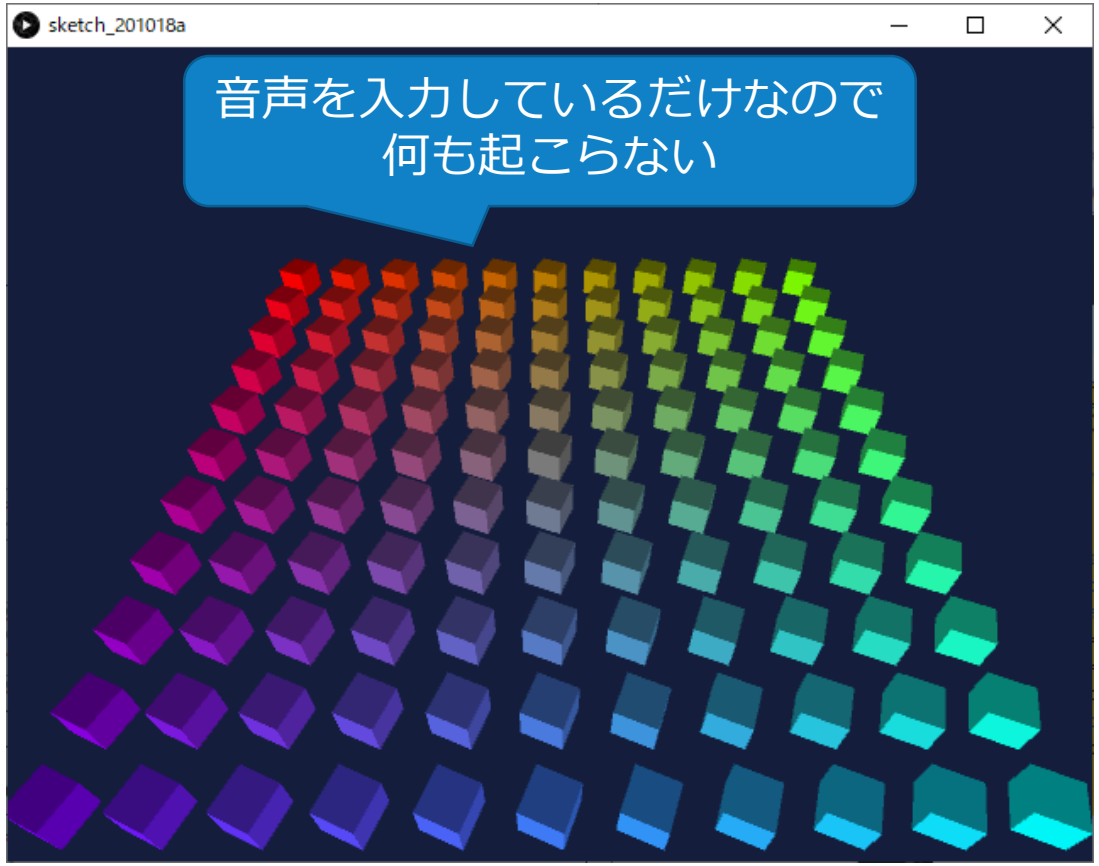
サウンド入力ライブラリを追加する





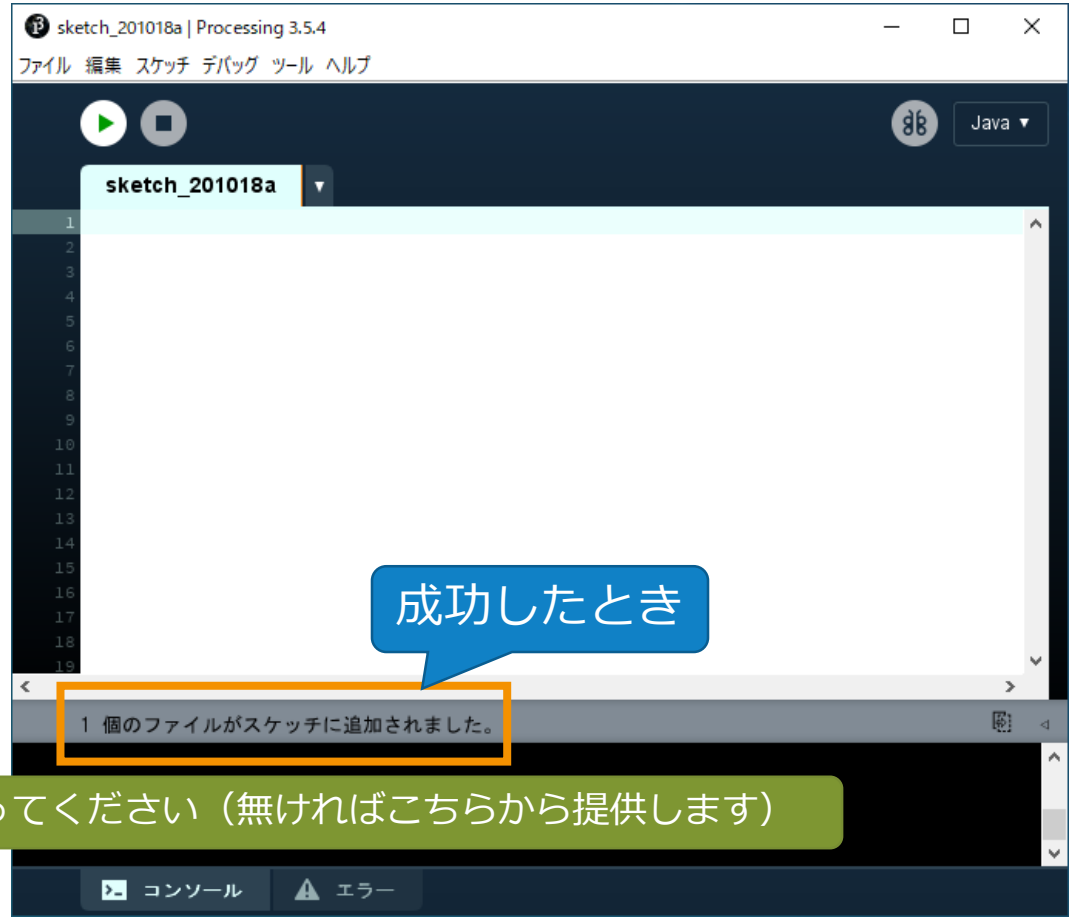
マイクから音声を入力する

```
PShape box;
import processing.sound.*;
AudioIn snd;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
  box.setStroke(false);
  snd = new AudioIn(this, 0);
  snd.start();
}

// 以下省略
```



フーリエ変換による分析に進んでください

音声ファイルをスケッチに入れる



音声ファイルは手持ちのもの (mp3, wav, aiff) を使ってください（無ければこちらから提供します）

ファイルから音声を入力する

```
PShape box;
import processing.sound.*;
SoundFile snd;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
  box.setStroke(false);
  snd = new SoundFile(this, "sound.mp3");
  snd.loop();
}

// 以下省略
```

追加した音声ファイル名

フーリエ変換による分析

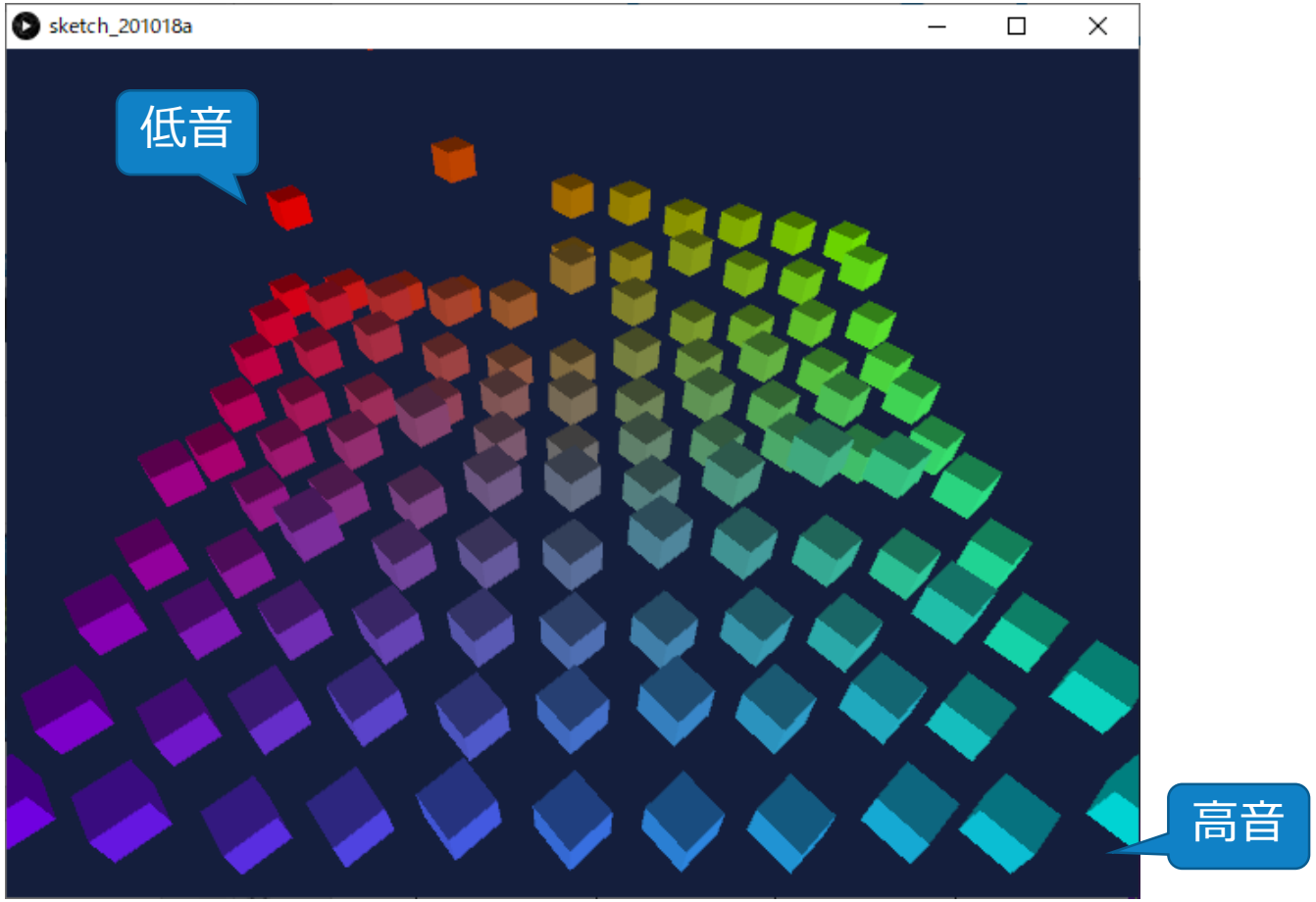
```
PShape box;
import processing.sound.*;
AudioIn snd;                        // マイク使用時
FFT fft;
int bands = 512;

void setup() {
  size(640, 480, P3D);
  box = createShape(BOX, 100);
  box.setStroke(false);
  snd = new AudioIn(this, 0);  // マイク使用時
  snd.start();                       // マイク使用時
  fft = new FFT(this, bands);
  fft.input(snd);
}

// 右に続く
```

```
void draw() {
  // 途中省略
  lights();
  float t = millis() * 0.001;
  fft.analyze();
  int i = 0;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      float y = -3000 * fft.spectrum[i++];
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, y, z);
      // 途中省略
      shape(box);
    }
  }
}
```

マイクの感度に
合わせて増減する

これも忘れずに

音に合わせて箱が上下する



音声入力の準備

```
import processing.sound.*;
```

- 音声入力ライブラリの読み込み

```
AudioIn snd;
```

- 音声入力のクラス

```
snd = new AudioIn(this, 0);
```

- 音声入力のオブジェクトの作成

```
snd.start();
```

- 音声入力の開始

音声入力の処理はグラフィックス表示と並行動作する

音声ファイルからの入力

```
SoundFile snd;
```

- 音声ファイルのクラス

```
snd = new SoundFile(this, "sound.mp3");
```

- 音声ファイルを読み込んで音声のオブジェクトを作成

```
snd.play()
```

- 音声オブジェクトの再生

```
snd.loop()
```

- 音声オブジェクトのループ再生

高速フーリエ変換 (FFT)

- 入力信号の周波数スペクトル (Frequency Spectrum) を求める

```
FFT fft;
```

  - FFT のクラス

```
fft = new FFT(this, bands);
```

  - this : たいてい this を指定する
  - bands : 周波数スペクトルの帯域の数、2 のべき乗 (16, 32, 64, 128, ...)

```
fft.analyze();
```

  - 高速フーリエ変換を実行して周波数スペクトルを求める

```
fft.input(snd);
```

  - snd : 入力信号

```
float y = -3000 * fft.spectrum[i++];
```

  - fft.spectrum[i] には i 番目 (0 ≦ i < bands) の周波数帯域のパワーが格納されている

降下時のスピードを遅くする

```
PShape box;
import processing.sound.*;
AudioIn snd;                          // マイク使用時
FFT fft;
int bands = 512;
float[] volume;

void setup() {
  // 途中省略
  fft.input(snd);
  volume = new float[bands];
}

// 右に続く
```

```
void draw() {
  background(20, 30, 60);
  fft.analyze();
  lights();
  float t = millis() * 0.001;
  int i = 0;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      volume[i] *= 0.95;
      if (volume[i] < fft.spectrum[i]) {
        volume[i] = fft.spectrum[i];
      }
      float y = -3000 * volume[i++];
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, y, z);
      // 以下省略
```

マイクの感度に
合わせて増減する

音量ピークメーター

```
volume[i] *= 0.95;
```

- 以前の音量レベルを下げる

```
if (volume[i] < fft.spectrum[i]) {
        volume[i] = fft.spectrum[i];
}
```

- 新しい音量が以前の音量レベルを超えていたら以前の音量レベルを新しい音量に更新する

```
float y = -3000 * volume[i++];
```

- 更新された音量レベルから高さを求めバンドの番号 i を進める

課題 7

音声入力

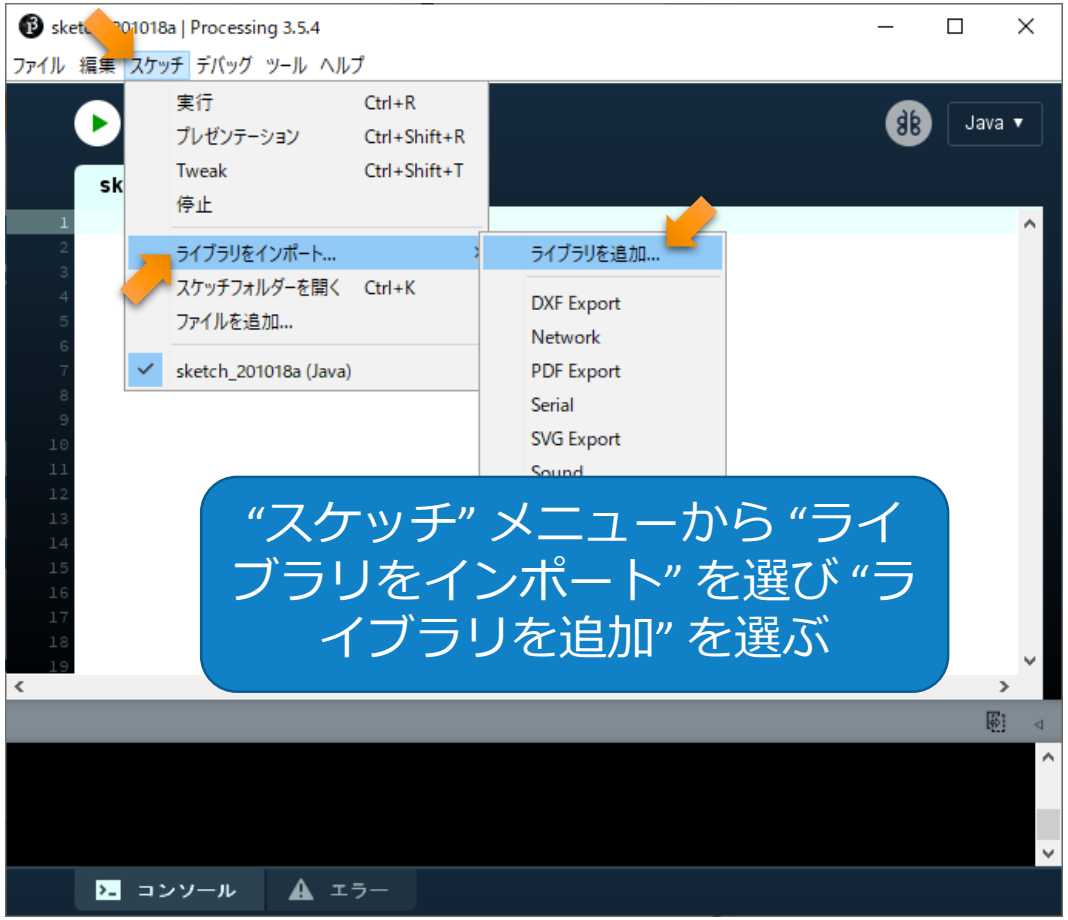
スクリーンショットをアップロードする

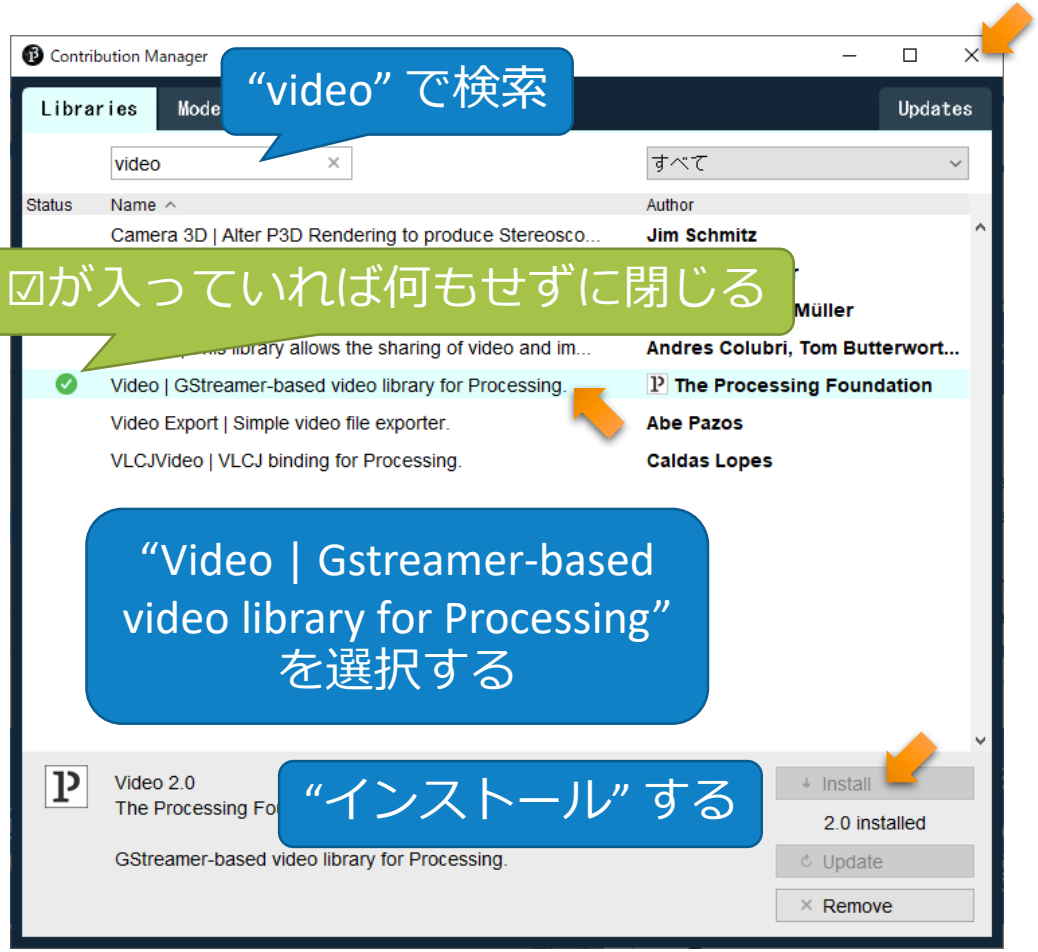
- 音声を入力したときの実行結果が表示されているウィンドウのスクリーンショットを適当なタイミングで撮って **07.png** というファイル名で保存して、Moodle の「スクリーンショットのアップロード」からアップロードしてください。

テクスチャマッピング

パソコンにカメラが付いていなければ「画像ファイルをスケッチに入れる」に進んでください

ビデオ入力ライブラリを追加する





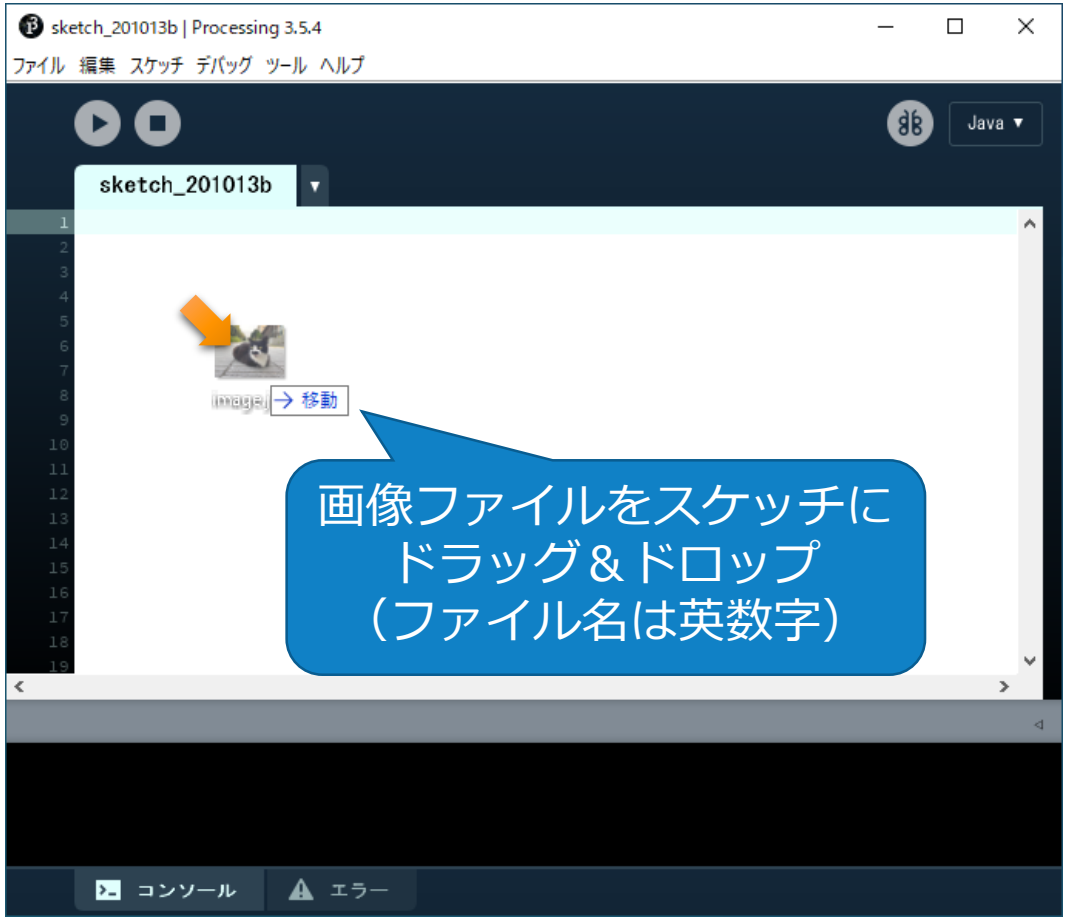
カメラから画像を入力する

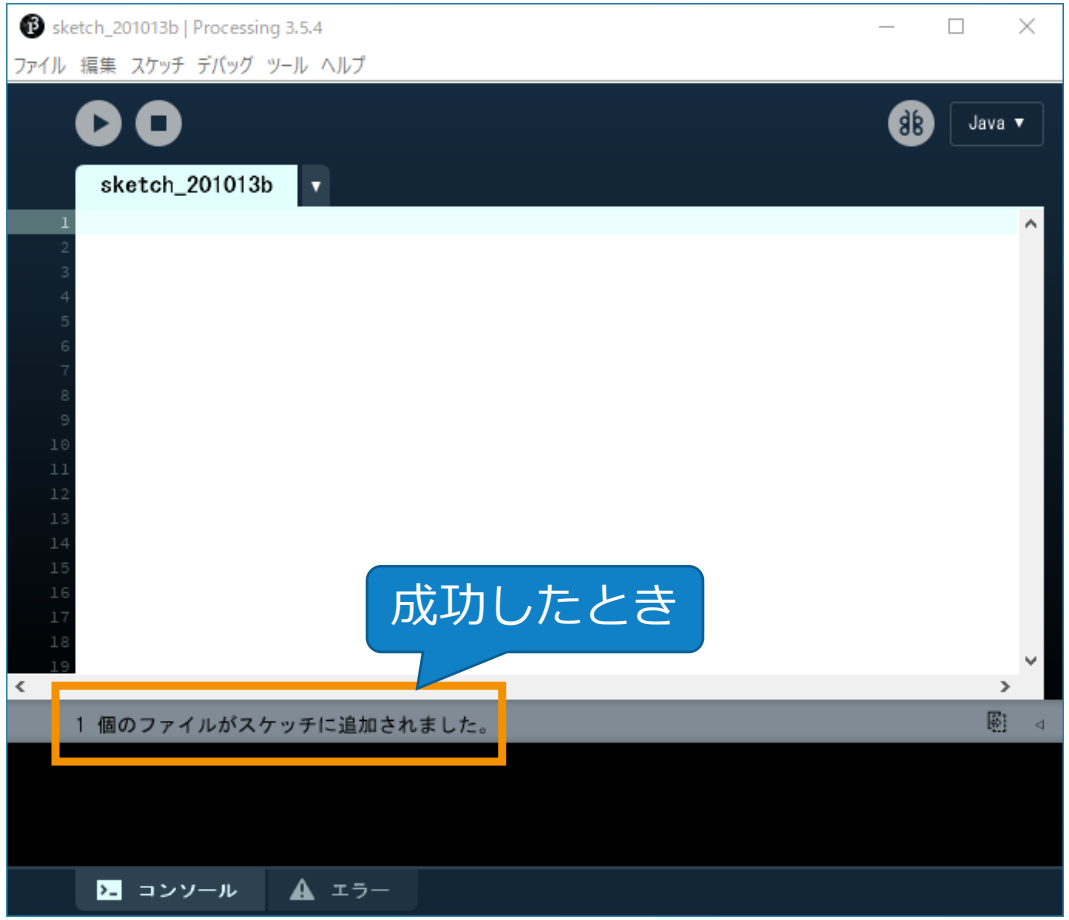
```
PShape box;
import processing.sound.*;
AudioIn snd;                          // マイク使用時
FFT fft;
int bands = 512;
float[] volume;
import processing.video.*;
Capture img;

void setup() {
  // 途中省略
  fft.input(snd);
  volume = new float[bands];
  img = new Capture(this);
  img.start();
}

// 右に続く
```

```
void draw() {
  // 途中省略
  lights();
  fft.analyze();
  if (img.available()) {
    img.read();
    img.resize(width, height);
    box.setTexture(img);
  }
  float t = millis() * 0.001;
  int i = 0;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      volume[i] *= 0.95;
      if (volume[i] < fft.spectrum[i]) {
        volume[i] = fft.spectrum[i];
      }
      // 以下省略
```

課題８に
進んでください

画像ファイルをスケッチに入れる





ファイルから画像を入力する

```
PShape box;
import processing.sound.*;
AudioIn snd;                          // マイク使用時
FFT fft;
int bands = 512;
float[] volume;
PImage img;

void setup() {
  // 途中省略
  fft.input(snd);
  volume = new float[bands];
  img = loadImage("image.jpg");
  box.setTexture(img);
}

// 右に続く
```

追加した画像ファイル名

```
void draw() {
  // 途中省略
  lights();
  fft.analyze();
  float t = millis() * 0.001;
  int i = 0;
  for (int z = -1000; z <= 1000; z += 200) {
    for (int x = -1000; x <= 1000; x += 200) {
      volume[i] *= 0.95;
      if (volume[i] < fft.spectrum[i]) {
        volume[i] = fft.spectrum[i];
      }
      float y = -3000 * volume[i++];
      box.resetMatrix();
      box.rotateY(PI * t);
      box.translate(x, y, z);
      // 以下省略
```

変更箇所なし

課題 8

箱にテクスチャをマッピングする

スクリーンショットをアップロードする

- 音声を入力したときの実行結果が表示されているウィンドウのスクリーンショットを適当なタイミングで撮って **08.png** というファイル名で保存して、Moodle の「スクリーンショットのアップロード」からアップロードしてください。