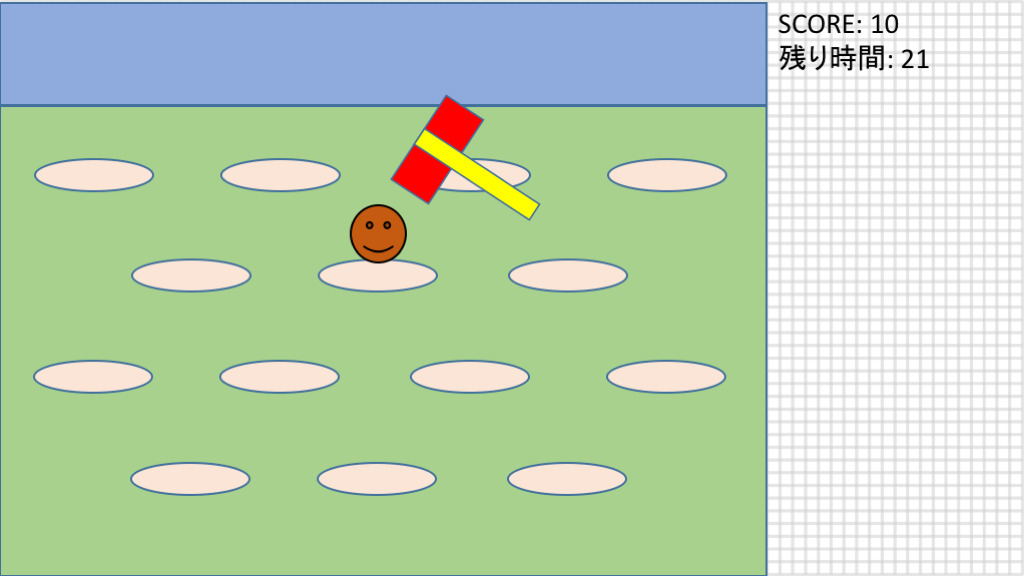
# 「モグラたたき」を作ろう

https://github.com/akokubo/whack\_a\_mole

## 画面デザイン



## 1.ディスプレイ・ウィンドウのサイズを指定

https://github.com/akokubo/whack\_a\_mole/tree/d2661f5f916f039212241181d46a991f1b190f2c/whack\_a\_mole

****void setup() {****

****// ディスプレイ・ウィンドウのサイズを640x360に****

****size(640, 360);****

****}****

****void draw() {****

****}****

## 2.円をランダムな位置に表示

https://github.com/akokubo/whack\_a\_mole/tree/67ddfdf250665bb79477f48cf0bdcf6cf61d055a/whack\_a\_mole

****// 円の座標****

****float x;****

****float y;****

****// 円の直径****

****float diameter;****

void setup() {

// ディスプレイ・ウィンドウのサイズを640x360に

size(640, 360);

// 直径を指定

diameter = 72;

// ランダムな位置を指定

x = random(0, width);

y = random(0, height);

}

void draw() {

// 円を表示

ellipse(x, y, diameter, diameter);

}

## 3.円をクリックすると、ランダムな位置に移動する

https://github.com/akokubo/whack\_a\_mole/tree/463fef5ee43cdd4523febde167da820ae197e97b/whack\_a\_mole

// 円の座標

float x;

float y;

// 円の直径

float diameter;

void setup() {

[中略]

}

void draw() {

****// 残像を消す****

****background(204);****

// 円を表示

ellipse(x, y, diameter, diameter);

****// マウスをクリックしたら****

****if (mousePressed) {****

****// 円の内側にマウスがあったら****

****if (dist(x, y, mouseX, mouseY) < diameter / 2) {****

****// ランダムな位置を再指定****

****x = random(0, width);****

****y = random(0, height);****

****}****

****}****

}

## 4.ステージ、モグラ、ハンマーの画像を追加

https://github.com/akokubo/whack\_a\_mole/tree/57520de0fdacac1f749bc2eab8a10c0766689d4a/whack\_a\_mole

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

PImage hammerImage; // ハンマー

// 円の座標

// モグラの座標

float x;

float y;

// 円の直径

float diameter;

void setup() {

// ディスプレイ・ウィンドウのサイズを640x360に

size(640, 360);

// 画像を表示するときに中心を指定するモードに設定

imageMode(CENTER);

// 直径を指定

diameter = 72;

// 画像の読み込み

stageImage = loadImage("stage.png");

moleImage = loadImage("mole.png");

hammerImage = loadImage("hammer.png");

// ランダムな位置を指定

x = random(0, width);

y = random(0, height);

}

void draw() {

// 残像を消す

background(204);

// ステージを表示

image(stageImage, 240, 180);

// 円を表示

ellipse(x, y, diameter, diameter);

// モグラを表示

image(moleImage, x, y);

// ハンマーを表示

image(hammerImage, mouseX, mouseY);

// マウスをクリックしたら

if (mousePressed) {

// 円の内側にマウスがあったら

if (dist(x, y, mouseX, mouseY) < diameter / 2) {

// モグラとハンマーが当たったら

if (dist(x, y, mouseX, mouseY) < (moleImage.width + hammerImage.width) / 2) {

// ランダムな位置を再指定

x = random(0, width);

y = random(0, height);

}

}

}

### stage.png: 480x360ピクセル



### mole.png: 72x72ピクセル



### hammer.png: 96x96ピクセル



## 5.モグラがステージからはみ出さないようにする

https://github.com/akokubo/whack\_a\_mole/tree/2fb38c2fc6dc8ee2dbdd2f9e124c4791a8141591/whack\_a\_mole

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

PImage hammerImage; // ハンマー

// モグラの座標

float x;

float y;

void setup() {

[中略]

// ランダムな位置を指定

x = random(0, width);

y = random(0, height);

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

}

void draw() {

[中略]

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (dist(x, y, mouseX, mouseY) < (moleImage.width + hammerImage.width) / 2) {

// ランダムな位置を再指定

x = random(0, width);

y = random(0, height);

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

}

}

}

## 6.スコアを表示

https://github.com/akokubo/whack\_a\_mole/tree/6547fd8bc26ac2db6c622b3025d8effebb5f32de/whack\_a\_mole

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

PImage hammerImage; // ハンマー

// モグラの座標

float x;

float y;

// フォント

PFont font;

// スコア

int score;

void setup() {

[中略]

// ランダムな位置を指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

// スコアを0に

score = 0;

// フォントを読み込み

font = createFont("MS Gothic", 20);

}

void draw() {

[中略]

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (dist(x, y, mouseX, mouseY) < (moleImage.width + hammerImage.width) / 2) {

// ランダムな位置を再指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

// スコアを増やす

score++;

}

}

// スコアを表示

textFont(font);

fill(0);

text("SCORE: " + score, stageImage.width, 20);

}

## 7.残り時間を表示

https://github.com/akokubo/whack\_a\_mole/tree/4d44bb911ae047417c835f3110d3f5f1e269078d/whack\_a\_mole

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

PImage hammerImage; // ハンマー

// モグラの座標

float x;

float y;

// フォント

PFont font;

// スコア

int score;

// 経過時間

int time;

// 制限時間

int timeMax;

void setup() {

[中略]

// フォントを読み込み

font = createFont("MS Gothic", 20);

// 制限時間を設定

timeMax = 20;

}

void draw() {

// 経過時間を求める

time = millis() / 1000;

if (time <= timeMax) {

// 残像を消す

background(204);

[中略]

// スコアを表示

textFont(font);

fill(0);

text("SCORE: " + score, stageImage.width, 20);

// 残り時間の表示

text("残り時間: " + (timeMax - time), stageImage.width, 40);

}

}

※実際には大幅なインデントの変更がある。

## 8.ハンマー・クラスの作成

https://github.com/akokubo/whack\_a\_mole/tree/07c9d6bf8f21fd6f38e31312fc64c21e7f02c8c6/whack\_a\_mole

### Hammerタブ

// ハンマー・クラス

class Hammer {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ

Hammer(PImage image) {

this.image = image;

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

void move() {

x = mouseX;

y = mouseY;

}

}

### メイン・タブ

// オブジェクト

Hammer hammer; // ハンマー

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

PImage hammerImage; // ハンマー

// モグラの座標

float x;

float y;

[中略]

void setup() {

// ディスプレイ・ウィンドウのサイズを640x360に

size(640, 360);

// 画像を表示するときに中心を指定するモードに設定

imageMode(CENTER);

// オブジェクトを作る

hammer = new Hammer(loadImage("hammer.png"));

// 画像の読み込み

stageImage = loadImage("stage.png");

moleImage = loadImage("mole.png");

hammerImage = loadImage("hammer.png");

// ランダムな位置を指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

[中略]

}

void draw() {

[中略]

// モグラを表示

image(moleImage, x, y);

// ハンマーを表示

image(hammerImage, mouseX, mouseY);

// ハンマーを移動して表示

hammer.move();

hammer.display();

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (dist(x, y, mouseX, mouseY) < (moleImage.width + hammerImage.width) / 2) {

if (dist(x, y, mouseX, mouseY) < (moleImage.width + hammer.image.width) / 2) {

// ランダムな位置を再指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

[中略]

}

## 9.モグラ・クラスの作成

https://github.com/akokubo/whack\_a\_mole/tree/4a6d210e5dfa2ec33b76018ccb4ddc249f1bd0b0/whack\_a\_mole

### Hammerタブ

変更なし

### Moleタブ

// モグラ・クラス

class Mole {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ

Mole(PImage image) {

this.image = image;

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

void move() {

x = random(this.image.width / 2, stageImage.width - this.image.width / 2);

y = random(64 + this.image.height / 2, stageImage.height - this.image.height / 2);

}

}

### メイン・タブ

// オブジェクト

Mole mole; // モグラ

Hammer hammer; // ハンマー

// 画像

PImage stageImage; // ステージ

PImage moleImage; // モグラ

// モグラの座標

float x;

float y;

// フォント

PFont font;

[中略]

void setup() {

// ディスプレイ・ウィンドウのサイズを640x360に

size(640, 360);

// 画像を表示するときに中心を指定するモードに設定

imageMode(CENTER);

// オブジェクトを作る

mole = new Mole(loadImage("mole.png"));

hammer = new Hammer(loadImage("hammer.png"));

// 画像の読み込み

stageImage = loadImage("stage.png");

mole.image = loadImage("mole.png");

// ランダムな位置を指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

// モグラをランダムな位置に移動

mole.move();

// スコアを0に

score = 0;

[中略]

}

void draw() {

[中略]

// モグラを表示

image(moleImage, x, y);

mole.display();

// ハンマーを移動して表示

hammer.move();

hammer.display();

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (dist(x, y, hammer.x, hammer.y) < (moleImage.width + hammerImage.width) / 2) {

if (dist(mole.x, mole.y, hammer.x, hammer.y)

< (mole.image.width + hammer.image.width) / 2) {

// ランダムな位置を再指定

x = random(moleImage.width / 2, stageImage.width - moleImage.width / 2);

y = random(64 + moleImage.height / 2, stageImage.height - moleImage.height / 2);

// モグラをランダムな位置に移動

mole.move();

// スコアを増やす

score++;

}

}

[中略]

}

## 10.スプライト・クラスの作成

https://github.com/akokubo/whack\_a\_mole/tree/e0d43823c839435b030870b8e0fa49ef2b39997b/whack\_a\_mole

### Spriteタブ

// スプライト・クラス

class Sprite {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ(デフォルト)

Sprite() {

}

// コンストラクタ(画像を指定するとき)

Sprite(PImage image) {

this.image = image;

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

void move() {

}

}

### Hammerタブ

// ハンマー・クラス

class Hammer {

// スプライト・クラスを継承したハンマー・クラス

class Hammer extends Sprite {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ

Hammer(PImage image) {

this.image = image;

// 親クラスのコンストラクタをそのまま使う

super(image);

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

// 移動(オーバーライド)

void move() {

x = mouseX;

y = mouseY;

}

}

### Moleタブ

// モグラ・クラス

class Mole {

// スプライト・クラスを継承したモグラ・クラス

class Mole extends Sprite {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ

Mole(PImage image) {

this.image = image;

// 親クラスのコンストラクタをそのまま使う

super(image);

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

// 移動(オーバーライド)

void move() {

x = random(this.image.width / 2, stageImage.width - this.image.width / 2);

y = random(64 + this.image.height / 2, stageImage.height - this.image.height / 2);

}

}

### メイン・タブ

変更なし

## 11.当たり判定をスプライト・クラスに移動

https://github.com/akokubo/whack\_a\_mole/tree/b01a230d3b9fb4eea543a7101bfc8ccbdd43f8e8/whack\_a\_mole

### Spriteタブ

// スプライト・クラス

class Sprite {

[中略]

// 移動

void move() {

}

// 当たり判定

boolean isContactedWith(Sprite sprite) {

// 判定結果を入れる変数。デフォルトはfalse

boolean result = false;

// 当たったらtrueに

if (dist(x, y, sprite.x, sprite.y) < (this.image.width + sprite.image.width) / 2) {

result = true;

}

return result;

}

}

### メイン・タブ

// オブジェクト

Mole mole; // モグラ

Hammer hammer; // ハンマー

[中略]

void setup() {

[中略]

}

void draw() {

// 経過時間を求める

time = millis() / 1000;

[中略]

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (dist(mole.x, mole.y, hammer.x, hammer.y)

< (mole.image.width + hammer.image.width) / 2) {

if (mole.isContactedWith(hammer)) {

// モグラをランダムな位置に移動

mole.move();

[中略]

}

## 12.ステージ・クラスを作る

https://github.com/akokubo/whack\_a\_mole/tree/38ffc163e0b7a9beb336db2c43d64627c64593fd/whack\_a\_mole

### Spriteタブ

変更なし

### Hammerタブ

変更なし

### Stageタブ

// スプライト・クラスを継承したステージ・クラス

class Stage extends Sprite {

// フォント

PFont font;

// スコア

int score;

// 経過時間

int time;

// 制限時間

int timeMax;

Stage(PImage image) {

// 親クラスのコンストラクタをそのまま使う

super(image);

// 画像の中心を指定

x = this.image.width / 2;

y = this.image.height / 2;

// スコアを0に

score = 0;

// フォントを読み込み

font = createFont("MS Gothic", 20);

// 制限時間を設定

timeMax = 20;

}

// 表示(オーバーライド)

void display() {

super.display();

// スコアを表示

textFont(font);

fill(0);

text("SCORE: " + score, this.image.width, 20);

// 残り時間の表示

text("残り時間: " + (timeMax - time), this.image.width, 40);

}

// 制限時間を過ぎたか？

boolean isTimeOver() {

// 判定結果を入れる変数

boolean result = false;

// 経過時間を求める

time = millis() / 1000;

// 制限時間を過ぎたか？

if (time > timeMax) {

result = true;

}

// 判定結果を返す

return result;

}

// スコアを増やす

void scoreUp() {

score++;

}

}

### Moleタブ

// スプライト・クラスを継承したモグラ・クラス

class Mole extends Sprite {

[中略]

// 移動(オーバーライド)

void move() {

x = random(this.image.width / 2, stageImage.width - this.image.width / 2);

y = random(64 + this.image.height / 2, stageImage.height - this.image.height / 2);

x = random(this.image.width / 2, stage.image.width - this.image.width / 2);

y = random(64 + this.image.height / 2, stage.image.height - this.image.height / 2);

}

}

### メイン・タブ

// オブジェクト

Stage stage; // ステージ

Mole mole; // モグラ

Hammer hammer; // ハンマー

// 画像

PImage stageImage; // ステージ

// フォント

PFont font;

// スコア

int score;

// 経過時間

int time;

// 制限時間

int timeMax;

void setup() {

size(640, 360);

// 画像を表示するときに中心を指定するモードに設定

imageMode(CENTER);

// オブジェクトを作る

stage = new Stage(loadImage("stage.png"));

mole = new Mole(loadImage("mole.png"));

hammer = new Hammer(loadImage("hammer.png"));

// 画像の読み込み

stageImage = loadImage("stage.png");

// モグラをランダムな位置に移動

mole.move();

// スコアを0に

score = 0;

// フォントを読み込み

font = createFont("MS Gothic", 20);

// 制限時間を設定

timeMax = 20;

}

void draw() {

// 経過時間を求める

time = millis() / 1000;

if (time <= timeMax) {

// 制限時間を過ぎていない

if (stage.isTimeOver() == false) {

// 残像を消す

background(204);

// ステージを表示

image(stageImage, 240, 180);

stage.display();

// モグラを表示

mole.display();

// ハンマーを移動して表示

hammer.move();

hammer.display();

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが接触したら

if (mole.isContactedWith(hammer)) {

// モグラをランダムな位置に移動

mole.move();

// スコアを増やす

score++;

stage.scoreUp();

}

}

// スコアを表示

textFont(font);

fill(0);

text("SCORE: " + score, stageImage.width, 20);

// 残り時間の表示

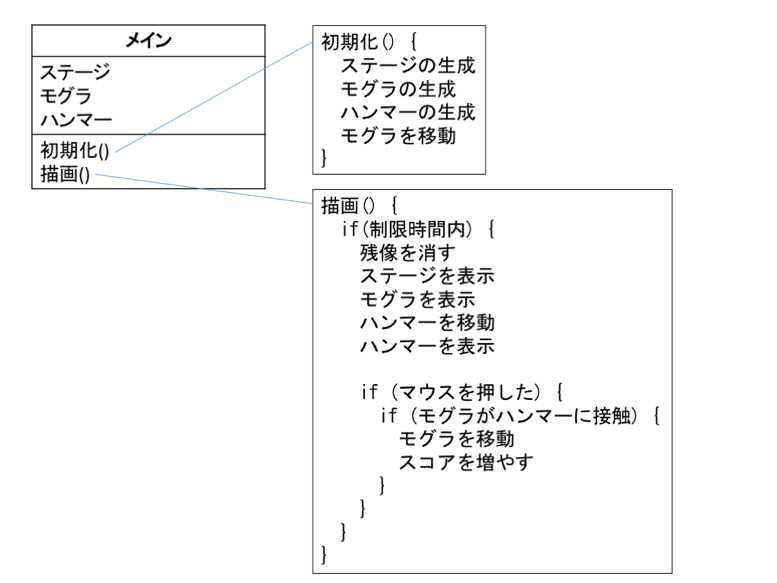
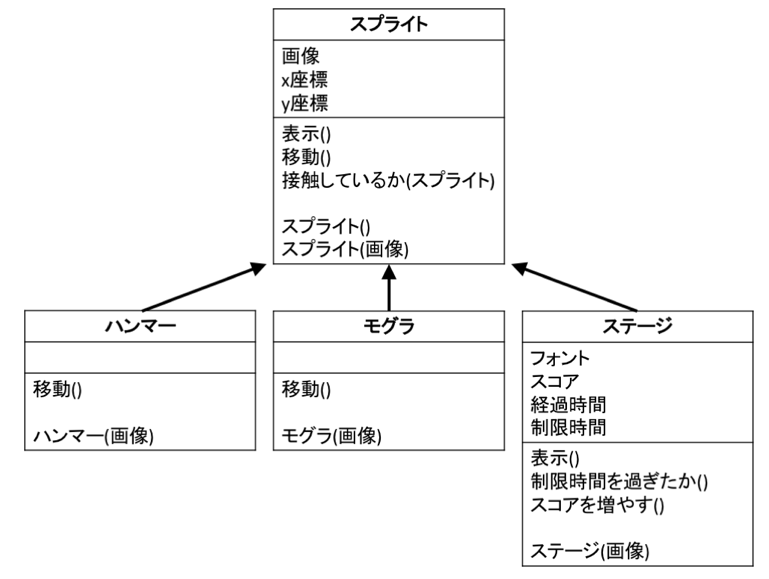
text("残り時間: " + (timeMax - time), stageImage.width, 40);

}

}

## 13.プログラムの最終版

https://github.com/akokubo/whack\_a\_mole/whack\_a\_mole



### Spriteタブ

// スプライト・クラス

class Sprite {

// 画像

PImage image;

// 座標

float x;

float y;

// コンストラクタ(デフォルト)

Sprite() {

}

// コンストラクタ(画像を指定するとき)

Sprite(PImage image) {

this.image = image;

}

// 表示

void display() {

image(this.image, x, y);

}

// 移動

void move() {

}

// 当たり判定

boolean isContactedWith(Sprite sprite) {

// 判定結果を入れる変数。デフォルトはfalse

boolean result = false;

// 当たったらtrueに

if (dist(x, y, sprite.x, sprite.y) < (this.image.width + sprite.image.width) / 2) {

result = true;

}

return result;

}

}

### Hammerタブ

// スプライト・クラスを継承したハンマー・クラス

class Hammer extends Sprite {

// コンストラクタ

Hammer(PImage image) {

// 親クラスのコンストラクタをそのまま使う

super(image);

}

// 移動(オーバーライド)

void move() {

x = mouseX;

y = mouseY;

}

}

### Moleタブ

// スプライト・クラスを継承したモグラ・クラス

class Mole extends Sprite {

// コンストラクタ

Mole(PImage image) {

// 親クラスのコンストラクタをそのまま使う

super(image);

}

// 移動(オーバーライド)

void move() {

x = random(this.image.width / 2, stage.image.width - this.image.width / 2);

y = random(64 + this.image.height / 2, stage.image.height - this.image.height / 2);

}

}

### Stageタブ

// スプライト・クラスを継承したステージ・クラス

class Stage extends Sprite {

// フォント

PFont font;

// スコア

int score;

// 経過時間

int time;

// 制限時間

int timeMax;

Stage(PImage image) {

// 親クラスのコンストラクタをそのまま使う

super(image);

// 画像の中心を指定

x = this.image.width / 2;

y = this.image.height / 2;

// スコアを0に

score = 0;

// フォントを読み込み

font = createFont("MS Gothic", 20);

// 制限時間を設定

timeMax = 20;

}

// 表示(オーバーライド)

void display() {

super.display();

// スコアを表示

textFont(font);

fill(0);

text("SCORE: " + score, this.image.width, 20);

// 残り時間の表示

text("残り時間: " + (timeMax - time), this.image.width, 40);

}

// 制限時間を過ぎたか？

boolean isTimeOver() {

// 判定結果を入れる変数

boolean result = false;

// 経過時間を求める

time = millis() / 1000;

// 制限時間を過ぎたか？

if (time > timeMax) {

result = true;

}

// 判定結果を返す

return result;

}

// スコアを増やす

void scoreUp() {

score++;

}

}

### メイン・タブ

// オブジェクト

Stage stage; // ステージ

Mole mole; // モグラ

Hammer hammer; // ハンマー

void setup() {

// ディスプレイ・ウィンドウのサイズを640x360に

size(640, 360);

// 画像を表示するときに中心を指定するモードに設定

imageMode(CENTER);

// オブジェクトを作る

stage = new Stage(loadImage("stage.png"));

mole = new Mole(loadImage("mole.png"));

hammer = new Hammer(loadImage("hammer.png"));

// モグラをランダムな位置に移動

mole.move();

}

void draw() {

// 制限時間を過ぎていない

if (stage.isTimeOver() == false) {

// 残像を消す

background(204);

// ステージを表示

stage.display();

// モグラを表示

mole.display();

// ハンマーを移動して表示

hammer.move();

hammer.display();

// マウスをクリックしたら

if (mousePressed) {

// モグラとハンマーが当たったら

if (mole.isContactedWith(hammer)) {

// モグラをランダムな位置に移動

mole.move();

// スコアを増やす

stage.scoreUp();

}

}

}

}