* 作品の意図

　これは「さめがめ」というゲームです。二つ以上同じ色のブロックがくっついたら消すことができ、一番良いクリアの状態は画面にブロックが一つもない状態です。現在ではあまり見ることのないゲームだったので、ぜひ知ってもらいたいと思い、作りました。

　尚、「pf[i][o] = 1+int(random(4));」の行にて、random()の()内の数字を変えると、出てくるブロックの色の種類を増減できます。

　また、コードの数値をいろいろ変えてみたものの、エラーが表示され、なかなかエラーの解読ができなかった為、原作に似てしまっているところが失敗です。

* 引用元サイト

http://www.monlynx.de/samegame/index.html

* ソースコード（赤い部分が元コードからの変更部分です。）

// sketch\_160615a.pde

// Source code is available under the Creative Commons Attribution License

// cc by Yuki Fukushima, 2016

static int hs = 0;

final int pfx = 16;

final int pfy = 8;

final int cs = 40;

final int ppx = cs+80;

final int ppy = pfy\*cs+50;

final int scorex = cs+80;

final int scorey = pfy\*cs+30;

final int hscorex = cs\*pfx/2;

final int hscorey = scorey;

Game game;

boolean clearing = false;

PFont font;

int lp = -1;

int score;

final color white = color(255, 255, 255);

final color gray = color(128, 128, 128);

final color green = color(0, 255, 0);

void setup()

{

size(660, 400);

background(250,250,0);

frameRate(60);

stroke(gray);

noFill();

font = loadFont("MS-PGothic-48.vlw");

textFont(font, 12);

rect(10, 10, pfx\*cs, pfy\*cs);

fill(green);

text("Score:", scorex-80, scorey);

fill(255, 0, 0);

text("HScore:", hscorex-80, hscorey);

text(hs, hscorex, hscorey);

fill(gray);

text("Possible:", ppx+340, hscorey);

game = new Game(pfx, pfy);

}

void mousePressed()

{

int cx, cy;

if ( clearing ) return;

if ( mouseX < 10 || mouseX > pfx\*cs+10 ) return;

if ( mouseY < 10 || mouseY > pfy\*cs+10 ) {

game.newGame();

return;

}

cx = (mouseX-10)/cs;

cy = (mouseY-10)/cs;

if ( game.selected(cx, cy) ) {

clearing = game.clearArea(cx, cy);

}

}

void draw()

{

if (! clearing ) {

int cx, cy;

cx = (mouseX-10)/cs;

cy = (mouseY-10)/cs;

if ( mouseX > 10 && mouseY > 10 && cx >= 0 && cx < pfx && cy >= 0 && cy < pfy ) {

if ( !game.selected(cx, cy) ) {

game.resetArea();

int n = game.selectArea(cx, cy);

if ( n == 1 ) {

game.resetArea();

} else {

int p = 0;

if ( n > 2 ) {

p = (n-2)\*(n-2);

}

if ( p != lp ) {

fill(white);

stroke(white);

rect(ppx+400, hscorey, 30, -15);

fill(0);

stroke(white);

text(p, ppx+400, hscorey);

lp = p;

}

}

}

} else {

game.resetArea();

}

} else {

game.compact();

clearing = game.clearArea(-1, -1);

if ( !clearing ) {

score += lp;

drawScore();

}

delay(25);

}

game.redraw();

}

void drawScore()

{

fill(white);

stroke(white);

rect(scorex, scorey, 40, -15);

fill(0);

stroke(white);

text(score, scorex, scorey);

if ( score > hs ) {

hs = score;

drawHScore();

}

}

void drawHScore()

{

fill(white);

stroke(white);

rect(hscorex, hscorey, 40, -15);

fill(255, 0, 0);

text(hs, hscorex, hscorey);

}

class Game {

color [] bc;

int [][] pf;

final int pfx, pfy;

int pfx0;

int pts;

int clear;

int tiles;

void pfe(int x, int y)

{

int c = pf[x][y];

color col = bc[abs(c)];

stroke(0);

fill(white);

rect(11+x\*cs, 11+y\*cs, cs-2, cs-2);

stroke(col);

fill(col);

if ( c < 0 ) {

rect(10+cs/4+x\*cs, 10+cs/4+y\*cs, cs/2, cs/2);

} else if ( c > 0 ) {

rect(12+x\*cs, 12+y\*cs, cs-4, cs-4);

}

}

boolean selected(int x, int y)

{

return pf[x][y] < 0;

}

void resetArea()

{

for (int x = 0; x < pfx0; ++x) {

for (int y = 0; y < pfy; ++y) {

pf[x][y] = abs(pf[x][y]);

}

}

}

int selectArea(int x0, int y0)

{

int c0 = pf[x0][y0];

int x, y;

color col = bc[abs(c0)];

int n = 0;

if ( c0 == 0 ) return n;

y = y0;

for ( x = x0; x < pfx0 && pf[x][y] == c0; ++x ) {

++n;

pf[x][y] = -c0;

if ( y > 0 && (pf[x][y-1] == c0) ) n += selectArea(x, y-1);

if ( y < pfy-1 && (pf[x][y+1] == c0) ) n += selectArea(x, y+1);

}

for ( x = x0-1; x >= 0 && pf[x][y] == c0; --x ) {

++n;

pf[x][y] = -c0;

if ( y > 0 && (pf[x][y-1] == c0) ) n += selectArea(x, y-1);

if ( y < pfy-1 && (pf[x][y+1] == c0) ) n += selectArea(x, y+1);

}

return n;

}

boolean compact()

{

int n;

for (int x = 0; x < pfx0; ++x) {

n = 0;

for (int y = pfy-1; y > 0; --y ) {

if ( pf[x][y] == 0 ) {

if ( pf[x][y-1] != 0) {

pf[x][y] = pf[x][y-1];

pf[x][y-1] = 0;

} else {

++n;

}

}

}

if ( n == pfy-1 && x < pfx0-1) {

for (int o = 0; o < pfy; ++o ) {

--tiles;

pf[x][o] = pf[x+1][o];

pf[x+1][o] = 0;

}

if ( x == pfx0-1 ) {

--pfx0;

}

}

}

return false;

}

boolean clearArea(int x0, int y0)

{

int c0;

int x, y;

int n = 0;

if ( x0 >= 0 ) {

clear = pf[x0][y0];

}

for (x = 0; x < pfx0; ++x) {

for (y = 0; y < pfy; ++y) {

if ( pf[x][y] == clear ) {

pf[x][y] = 0;

return true;

}

}

}

return false;

}

void redraw()

{

for (int i = 0; i < pfx; ++i) {

for (int o = 0; o < pfy; ++o) {

pfe(i, o);

}

}

}

void newGame()

{

score = 0;

pfx0 = pfx;

tiles = pfx\*pfy;

for (int i = 0; i < pfx; ++i) {

for (int o = 0; o < pfy; ++o) {

pf[i][o] = 1+int(random(4));

//random=color variation

}

}

}

Game(int x, int y) {

pfx = pfx0 = x;

pfy = y;

pf = new int[pfx][pfy];

bc = new color[6];

bc[0] = color(0, 0, 0);

bc[1] = color(255, 0, 0);

bc[2] = color(green);

bc[3] = color(0, 0, 255);

bc[4] = color(0, 255, 255);

bc[5] = color(255, 0, 255);

clear = 0;

newGame();

};

void dump()

{

for (int y = 0; y < pfy; ++y ) {

for (int x = 0; x < pfx; ++x ) {

print(pf[x][y]+" ");

}

println("");

}

}

}