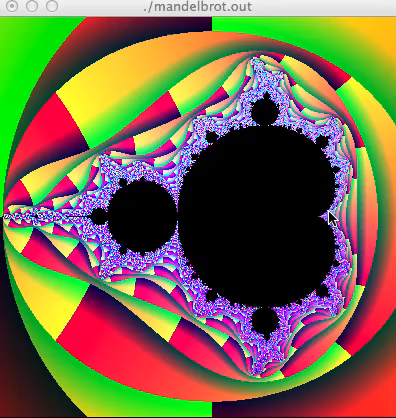
マンデルブロ集合[動画]

対話的に拡大、縮小が可能なマンデルブロ集合の描画。

コンパイルする際は-std=c++11オプションをつけてください。

右図を右クリックして再生を選ぶと動画が再生されます。

色づけは

abs(z)>2を超えた時点での

R：絶対値

G：偏角

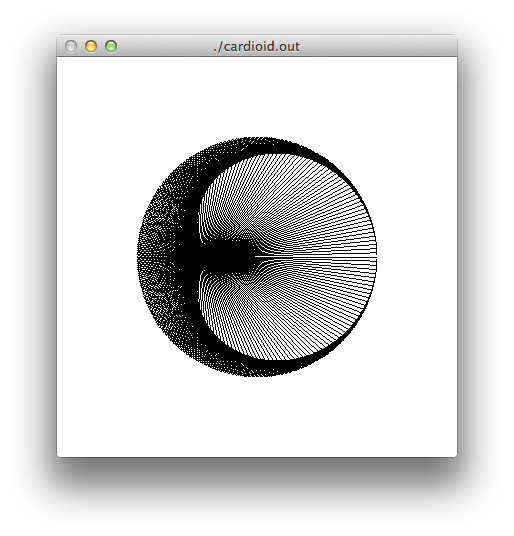
B：かかったステップ数

を0~1の範囲に納めています。

1 #include <GLUT/glut.h> // ライブラリ用ヘッダファイルの読み込み  
 2 #include <math.h>  
 3 #include <vector>  
 4 #include <complex>  
 5 #include <iostream>  
 6 /\* Mandelbrot setをOpenGLで鑑賞する  
 7 \* 左クリック:クリック位置を中央にして2倍拡大  
 8 \* 右クリック:同様に4倍縮小  
 9 \*/  
 10 const int W = 400; //画面サイズ  
 11 const int H = 400;  
 12 std::complex<double> zoom(0.7, 0.0); //拡大率  
 13 std::complex<double> center(-0.6, 0.0); //中心座標  
 14   
 15 void display(void){ //表示関数  
 16 glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
 17 glClear (GL\_COLOR\_BUFFER\_BIT ); // 画面消去  
 18 glBegin(GL\_POINTS);  
 19 for(double i = -1.0 ; i < 1; i += 2.0/W){  
 20 for(double j = -1.0 ; j < 1.0; j += 2.0/H){  
 21 std::complex<double> c(i, j);  
 22 c = c/zoom + center; //正規化座標系から投射  
 23 std::complex<double> z(0, 0);  
 24 int k = 0; //abs(z)>2を超えたら必ず発散する  
 25 for(k = 0; k < 100 & abs(z) < 2; k++){  
 26 z = z\*z + c; //漸化式  
 27 }  
 28 if(abs(z) > 2){  
 29 //r:絶対値, g:偏角, b:発散の速さ  
 30 glColor3d(abs(z) - 2, arg(z)/(2\*M\_PI) + 0.5, k / 10.0);  
 31 }else{  
 32 glColor3d(0.0, 0.0, 0.0);  
 33 }  
 34 glVertex2d(i, j);  
 35 }  
 36 }  
 37 glEnd();  
 38 glFlush(); // 画面出力  
 39 }  
 40   
 41 void mouse(int button, int state, int x, int y){ //クリックしたら呼ばれる  
 42 if(state == 0){ //ピクセル座標系から正規化座標系へ投射  
 43 std::complex<double> c(2\*double(x)/double(W)-1, 1-2\*double(y)/double(H));  
 44 center = c/zoom + center; //  
 45 if(button == 0){  
 46 zoom \*= 2.0;  
 47 }else if(button == 2){  
 48 zoom /= 4.0;  
 49 }  
 50 display();  
 51 }  
 52 }  
 53   
 54 // メインプログラム  
 55 int main (int argc, char \*argv[]) {  
 56 glutInit(&argc, argv); // ライブラリの初期化  
 57 glutInitWindowSize(W , H); // ウィンドウサイズを指定  
 58 glutCreateWindow(argv[0]); // ウィンドウを作成  
 59 glutDisplayFunc(display); // 表示関数を指定  
 60 glutMouseFunc(mouse); // マウス関数を指定  
 61 glViewport(0, 0, W, H);  
 62 glutMainLoop(); // イベント待ち  
 63 return 0;  
 64 }

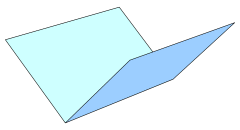
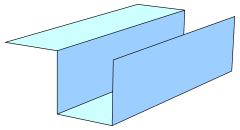
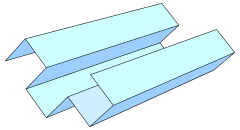
カージオイド

コーヒーカップに差し込む光が照らす領域を可視化。カージオイドの包絡線となる。詳細はコメントアウト。

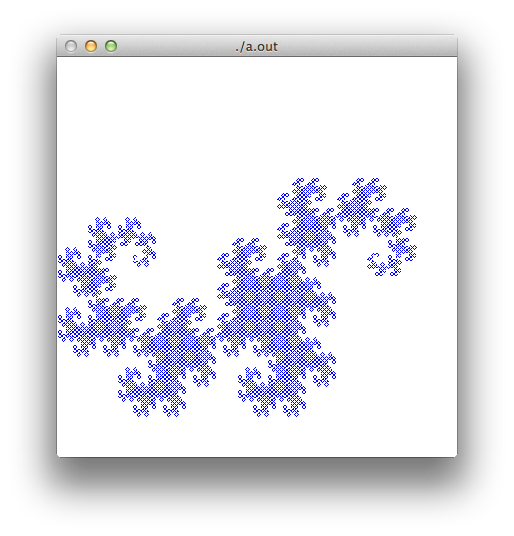
  
 1 #include <GLUT/glut.h> // ライブラリ用ヘッダファイルの読み込み  
 2 #include <math.h>  
 3   
 4 // 表示部分をこの関数で記入  
 5 void display(void) {  
 6 glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
 7 glClear (GL\_COLOR\_BUFFER\_BIT ); // 画面消去  
 8   
 9 glColor3d(0.0, 0.0, 0.0); // 色指定(R,G,B)で0～1まで  
 10 glBegin(GL\_LINES);  
 11 /\* コーヒーカップに西日が差し込むと次に表示されるようなハート型が現れることがある。  
 12 \* これは入射角=反射角の法則により任意のtに対して線分(cost, sint),(cos2t,sin2t)が光で照らされるからだ。  
 13 \* このような線分を包絡線にもつような曲線をcardioid(心臓形)という。  
 14 \* 代表的な図形は極方程式においてr=costで表される図だろう。  
 15 \*/  
 16 for(int i = 0; i < 360; i++) {  
 17 double x1 = cos(i \* 3.14159 /180.0);  
 18 double y1 = sin(i \* 3.14159 /180.0);  
 19 double x2 = cos(2 \* i \* 3.14159 /180.0);  
 20 double y2 = sin(2 \* i \* 3.14159 /180.0);  
 21 glVertex2d(x1 \* 0.6, y1 \* 0.6);  
 22 glVertex2d(x2 \* 0.6, y2 \* 0.6);  
 23 }  
 24 glEnd();  
 25   
 26 glFlush(); // 画面出力  
 27 }  
 28   
 29 // メインプログラム  
 30 int main (int argc, char \*argv[]) {  
 31 glutInit(&argc, argv); // ライブラリの初期化  
 32 glutInitWindowSize(400 , 400); // ウィンドウサイズを指定  
 33 glutCreateWindow(argv[0]); // ウィンドウを作成  
 34 glutDisplayFunc(display); // 表示関数を指定  
 35 glutMainLoop(); // イベント待ち  
 36 return 0;  
 37 }

ドラゴン曲線

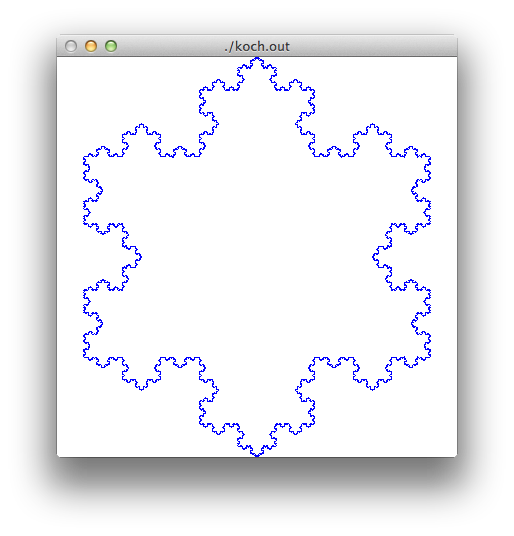
下図のようにn重に折った紙を広げた断面に現れる曲線

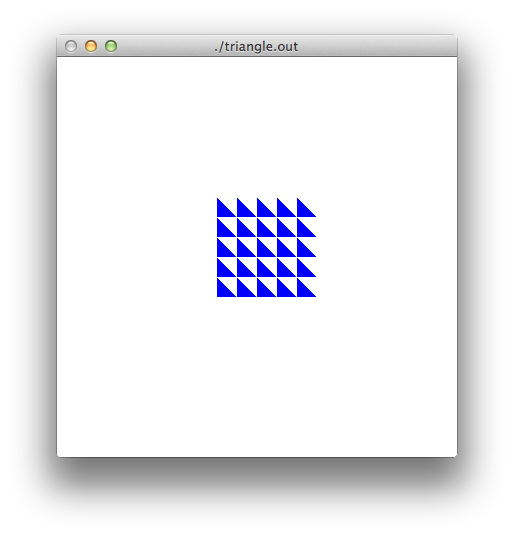
下は20次ドラゴン曲線

  
 1 #include <GLUT/glut.h> // ライブラリ用ヘッダファイルの読み込み  
 2 #include <math.h>  
 3 #include <vector>  
 4 #include <complex>  
 5 #include <iostream>  
 6 #include <algorithm>  
 7   
 8 //複素数zを用いて,点P = (x, y) を P(z) = (real(z), imag(z))として表す。  
 9   
 10 /\*頂点位置を反時計回りに格納した配列を引数にとり、  
 11 \* 一世代後の頂点を追加した配列を返す関数。  
 12 \*/  
 13 std::vector<std::complex<double>> dragon(const std::vector<std::complex<double>> &src){  
 14 std::vector<std::complex<double>> dst = {};  
 15 int i;  
 16 std::complex<double> a;  
 17 for(i=0; i+1<src.size(); i++){  
 18 if(i%2==0){  
 19 a = std::polar(sqrt(2.0) / 2.0, - M\_PI / 4.0) \* (src[i+1] - src[i]) + src[i];  
 20 }else{  
 21 a = std::polar(sqrt(2.0) / 2.0, M\_PI / 4.0) \* (src[i+1] - src[i]) + src[i];  
 22 }  
 23 dst.push\_back(src[i]);  
 24 dst.push\_back(a);  
 25 }  
 26 dst.push\_back(src[i]);  
 27 return dst;  
 28 }  
 29   
 30 // 表示部分をこの関数で記入  
 31 void display(void) {  
 32 //初期状態の線分  
 33 const std::complex<double> a(-0.6, 0);  
 34 const std::complex<double> b(0.6, 0);  
 35   
 36 //初期状態  
 37 std::vector<std::complex<double>> v{a, b};  
 38   
 39 //再起的に20世代後のドラゴン曲線の頂点を求める。  
 40 for(int i=0; i<20; i++){  
 41 v = dragon(v);  
 42 }  
 43 //debug  
 44 // for(int i=0; i<v.size(); i++){  
 45 // std::cout << v[i] << std::endl;  
 46 // }  
 47   
 48 glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
 49 glClear (GL\_COLOR\_BUFFER\_BIT ); // 画面消去  
 50   
 51 glColor3d(0.0, 0.0, 1.0); // 色指定(R,G,B)で0～1まで  
 52   
 53 // glBegin(GL\_LINE\_LOOP);  
 54 // glBegin(GL\_LINES);  
 55 glBegin(GL\_LINE\_STRIP);  
 56 //実際に複素平面をplot  
 57 for(int i=0; i<v.size(); i++){  
 58 glVertex2d(std::real(v[i]), std::imag(v[i]));  
 59 }  
 60 glEnd();  
 61   
 62 glFlush(); // 画面出力  
 63 }  
 64   
 65 // メインプログラム  
 66 int main (int argc, char \*argv[]) {  
 67 glutInit(&argc, argv); // ライブラリの初期化  
 68 glutInitWindowSize(400 , 400); // ウィンドウサイズを指定  
 69 glutCreateWindow(argv[0]); // ウィンドウを作成  
 70 glutDisplayFunc(display); // 表示関数を指定  
 71 glutMainLoop(); // イベント待ち  
 72 return 0;  
 73 }  
 74

課題のコッホ雪片。

  
 1 #include <GLUT/glut.h> // ライブラリ用ヘッダファイルの読み込み  
 2 #include <math.h>  
 3 #include <vector>  
 4 #include <complex>  
 5 #include <iostream>  
 6 #include <algorithm>  
 7   
 8 //複素数zを用いて,点P = (x, y) を P(z) = (real(z), imag(z))として表す。  
 9   
 10 /\*頂点位置を反時計回りに格納した配列を引数にとり、  
 11 \* 一世代後の頂点を追加した配列を返す関数。  
 12 \*/  
 13 std::vector<std::complex<double>> koch(const std::vector<std::complex<double>> &src){  
 14 std::vector<std::complex<double>> dst = {};  
 15 int i;  
 16 for(i=0; i+1<src.size(); i++){  
 17 std::complex<double> a = (2.0 \* src[i] + 1.0 \* src[i+1])/3.0;  
 18 std::complex<double> b = (1.0 \* src[i] + 2.0 \* src[i+1])/3.0;  
 19 std::complex<double> c = std::polar(1.0, - M\_PI / 3.0) \* (b - a) + a;  
 20 dst.push\_back(src[i]);  
 21 dst.push\_back(a);  
 22 dst.push\_back(c);  
 23 dst.push\_back(b);  
 24 }  
 25 dst.push\_back(src[i]);  
 26 return dst;  
 27 }  
 28   
 29 // 表示部分をこの関数で記入  
 30 void display(void) {  
 31 //ガウス平面における単位円上の正三角形の頂点をa,b,cとする  
 32 const std::complex<double> a = std::polar(1.0, M\_PI / 6.0 + 0.0 \* M\_PI / 3.0);  
 33 const std::complex<double> b = std::polar(1.0, M\_PI / 6.0 + 2.0 \* M\_PI / 3.0);  
 34 const std::complex<double> c = std::polar(1.0, M\_PI / 6.0 + 4.0 \* M\_PI / 3.0);  
 35   
 36 //初期状態として三角形の頂点を結んだ閉路を定義  
 37 std::vector<std::complex<double>> v{a, b, c, a};  
 38   
 39 //再起的に10世代後のコッホ曲線の頂点を求める。  
 40 for(int i=0; i<10; i++){  
 41 v = koch(v);  
 42 }  
 43 //debug  
 44 // for(int i=0; i<v.size(); i++){  
 45 // std::cout << v[i] << std::endl;  
 46 // }  
 47   
 48 glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
 49 glClear (GL\_COLOR\_BUFFER\_BIT ); // 画面消去  
 50   
 51 glColor3d(0.0, 0.0, 1.0); // 色指定(R,G,B)で0～1まで  
 52   
 53 glBegin(GL\_LINE\_LOOP);  
 54 // glBegin(GL\_LINES);  
 55 //実際に複素平面をplot  
 56 for(int i=0; i<v.size(); i++){  
 57 glVertex2d(std::real(v[i]), std::imag(v[i]));  
 58 }  
 59 glEnd();  
 60   
 61 glFlush(); // 画面出力  
 62 }  
 63   
 64 // メインプログラム  
 65 int main (int argc, char \*argv[]) {  
 66 glutInit(&argc, argv); // ライブラリの初期化  
 67 glutInitWindowSize(400 , 400); // ウィンドウサイズを指定  
 68 glutCreateWindow(argv[0]); // ウィンドウを作成  
 69 glutDisplayFunc(display); // 表示関数を指定  
 70 glutMainLoop(); // イベント待ち  
 71 return 0;  
 72 }  
 73

課題の5x5に並んだ三角形

  
 1 #include <GLUT/glut.h> // ライブラリ用ヘッダファイルの読み込み  
 2 #include <math.h>  
 3   
 4 // 表示部分をこの関数で記入  
 5 void display(void) {  
 6 glClearColor (1.0, 1.0, 1.0, 1.0); // 消去色指定  
 7 glClear (GL\_COLOR\_BUFFER\_BIT ); // 画面消去  
 8   
 9 glColor3d(0.0, 0.0, 1.0); // 色指定(R,G,B)で0～1まで  
 10 glBegin(GL\_TRIANGLES);  
 11   
 12 for(double i = -0.2; i < 0.3; i+=0.1) {  
 13 for(double j = -0.2; j < 0.3; j+=0.1) {  
 14 glVertex2d(i, j);  
 15 glVertex2d(i+0.1, j);  
 16 glVertex2d(i, j+0.1);  
 17 }  
 18 }  
 19   
 20 glEnd();  
 21   
 22 glFlush(); // 画面出力  
 23 }  
 24   
 25 // メインプログラム  
 26 int main (int argc, char \*argv[]) {  
 27 glutInit(&argc, argv); // ライブラリの初期化  
 28 glutInitWindowSize(400 , 400); // ウィンドウサイズを指定  
 29 glutCreateWindow(argv[0]); // ウィンドウを作成  
 30 glutDisplayFunc(display); // 表示関数を指定  
 31 glutMainLoop(); // イベント待ち  
 32 return 0;  
 33 }  
 34

感想

GPAがヤバいです。頑張るのでA+下さい。