IFS Fraktálok deformációja

A probléma megoldása, részfeladatok

A feladat

- két azonos számú affin függvénnyel definiált iterált függvényrendszer által generált fraktál egymásba transzformálása úgy, hogy a transzformáció során a generált fraktálok box-dimenziójának valamilyen alsó becslést tudjunk adni.
- a definiáló háromszögek csúcsai között olyan folytonos görbék keresése (valamilyen genetikus függvénnyel), amelyek mentén a lépésenként generált fraktálok box-dimenziója egy alsó korlát alá nem csökken.
- Apophysis-hez hasonló kezelőfelület kialakítása
- mentési lehetőség megvalósítása, a generált fraktálok animációja és a függvények szintjén egyaránt
- lehetőseég teremtése több generált fraktál egymásrahelyezésére, V-fraktálok, multifraktálok kezelésére és fraktáldomborzatok transzformálására

Részfeladatok

A probléma megoldására a következő részfeladatokat dolgozutk ki:

- a matematikai háttér alapos ismerete és kidolgozása
- algoritmusok kidolgozása: (fraktálgenerálás, fraktál deformáció, box-dimenzió kiszámítása, genetikus algoritmus amely olyan folytonos görbéket keres amelyek mentén a lépésenként generált fraktálok box-dimenziója nem csökken egy alsó korlát alá)
- a kidolgozott algoritmusok implementációja c++ nyelvbe
- dokumentáció készítése a program megvalósításáról és működéséről

Ennek a feladatnak az elkészítése önmagában is nagy kihívást jelent, a legnagyobb problémát azonban valószínűleg a box-dimenzió becslése fogja majd jelenteni. A legtöbb időt biztosan az algoritmusok kidolgozása és ezek implementálása fogja majd igényelni. A megvalósítást c++ környezetben képzeltük el, a grafikus megjelenítést pedig az OpenGl nyílt forráskódú függvénykönyvtár segítségével fogjuk megvalósítani. A fejlesztés GNU/Linux operációs rendszer alatt történik majd.

Időbeosztás és személyre szabott feladatleosztás

Mivel egy elég nagyméretű nehéz feladatról van szó, ami nagyon sok munkát igényel, úgy képzeltük el, hogy naponta legalább 1-2 órát dolgozunk majd a projekten, persze számításba kell vennünk, hogy más tantárgyakból is vannak házi feladatok, valamint készülnünk kell a vizsgákra is.

A személyre szabott feladatleosztás fele-fele arányban történik, mindketten résztveszünk úgy az algoritmusok kidolgozásában és implementálásában mint a matematikai háttér előszészítésében és a dokumentáció készítésében.

A munka végeredménye

Egy olyan programot szeretnénk létrehozni, amely különböző fraktálok egybetranszormálását teszi lehetővé. A program képes lesz háromszögek, valamint iterált függvényrendszerek segítségével fraktálokat kigenerálni, valamint ezeket egymásba transzformálni. Feltétel, hogy a két fraktált azonos számú affin függvény kell majd meghatározza.

Folytatási lehetőségek

Olyan könnyen érthető nyílt forráskódú programot szeretnénk létrehozni, ami jól szemlélteti a fraktálok szerkezetét, deformációját, és ami példaként szolgál majd mindazoknak akik ezeket jobban szeretnék megérteni. A kódot majd bárki továbbadhatja és továbbfejlesztheti. Elképzelhető, hogy a végeredmény a jövőben, majd valamilyen tudományos konferencián is bemutatásra kerül.

Alapfogalmak

Fraktál

A legegyszerűbb, de nem feltétlenül pontos megközelítésben – „önhasonló”, végtelenül komplex geometriai alakzatok. Az önhasonlóság azt jelenti, hogy egy kisebb rész felnagyítva ugyanolyan struktúrát mutat, mint egy nagyobb rész. Az eredeti és talán legszélesebb körben elfogadott definíció, miszerint fraktálnak nevezünk egy geometriai alakzatot akkor, ha „induktív” dimenziója (Lebesgue-dimenziója) nem esik egybe (szigorúan kisebb) a Hausdorrff-féle dimenziójával.

Transzformáció

A matematikában, elsősorban a geometriában transzformáció alatt egy halmaz önmagába való leképezéseit értjük. Speciális esetként a geometriai transzformációk olyan függvények, amelyek pontokhoz pontokat rendelnek hozzá, tehát értékkészletük és értelmezési tartományuk ponthalmaz.

Affin transzformáció

Az affin geometriában használt, illetve a lineáris algebra részeként is tárgyalható fogalom. Egy affin transzformáció során a transzformált koordináták az eredeti koordináták lineáris függvényeként állnak elő. Ide tartoznak a lineáris transzformációk és az eltolás is.

Iterált függvényrendszer, Iterated Function System(IFS) – A matematikában használt módszer, amellyel fraktálokat építenek fel. Ezek az IFS fraktálok mindig önhasonlóak. Az iterált függvényrendszerek a dinamikai rendszerek tanulmányozásában is használatosak.

Példa:

Legyen l_1 egy lineáris transzformáció, amely egy pontot leképez egy másik pontba. Az l_1 transzformációt a következőképpen definiáljuk:

$$l_1(0,0)=(0,0); \quad l_1(1,0)=(\frac{1}{2},0); \quad l_1(0,1)=(0,\frac{1}{2});$$

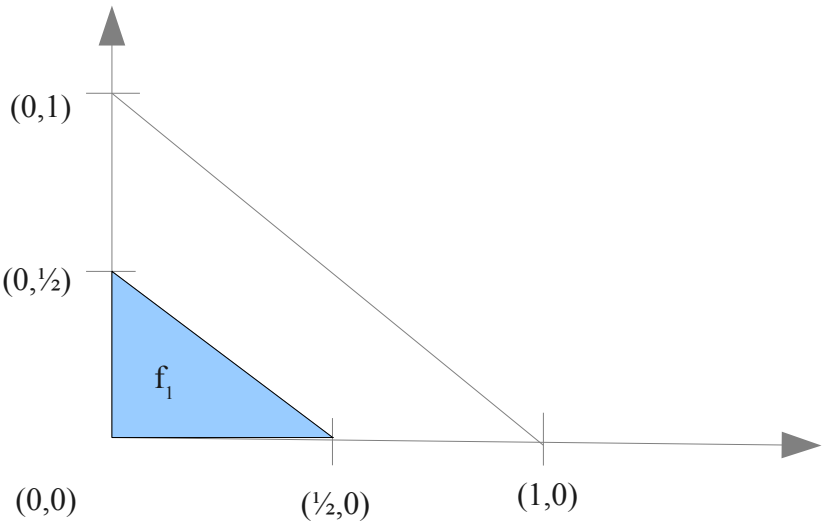
$f_1, f_2, f_3 \mathbb{R} \to \mathbb{R}$ Három pont által meghatározott affin transzformációk.

$$f_1 = l_1 + \vec{v}$$

Az f_1 egy affin transzformáció amit úgy kapunk meg, hogy egy lineáris transzformációt eltolunk

egy $\vec{v}$ vektorral. Ebben az esetben a $\vec{v}=(0,0)$

Grafikusan a következőt jelenti:

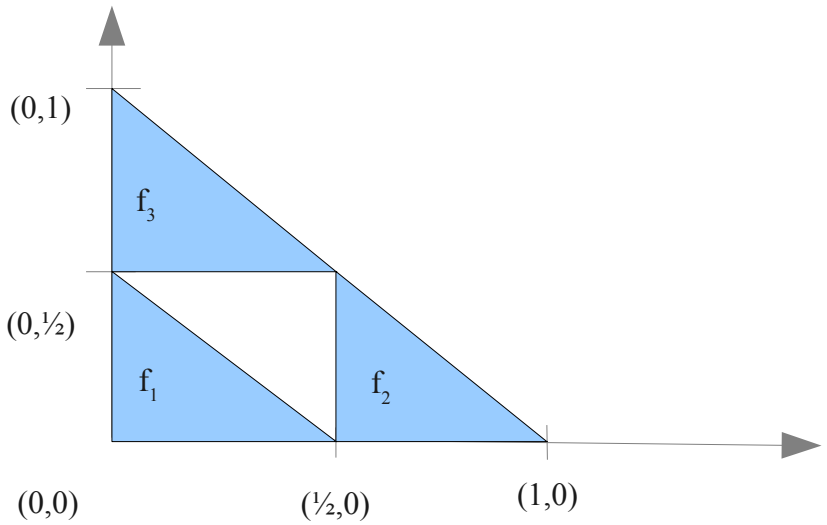


Az f_2, f_3 transzformációt az f_1 -hez hasonlóképpen kapom meg:

$$\{f_2(x,y)=f_1(x,y)+\vec{v}\}\ \vec{v}=(\frac{1}{2},0);$$

$$\{f_3(x,y)=f_1(x,y)+\vec{u}\}\ \vec{u}=(0,\frac{1}{2});$$

Grafikusan a következőképpen néz ki:



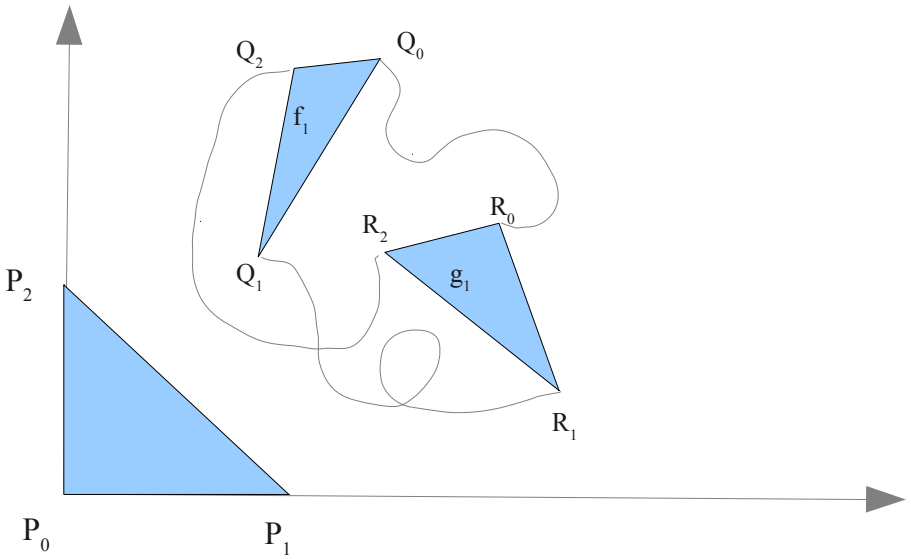
Ha a transzformációk segítségével megkapott kicsi háromszögekre újra és újra elvégezzük az előbbi transzformációkat, akkor egy önhasonló struktúrát fogunk kapni. Nem szükséges minden lépésben

(iterációban) elvégeznünk mind a három transzformációt, elég ha minden lépésben véletlenszerűen kiválasztunk egyet. Esetünkben a Sierpinski háromszöget fogjuk megkapni.

Fraktál deformáció

Fraktál deformáció alatt azt a folyamatot értjük, amely egy meglevő fraktál képét, átképezi egy másik fraktál képébe. Feltétel, hogy a két fraktál képét ugyanannyi pont határozza meg. A fraktál deformációt a következő példa segítségével szemléltetjük:

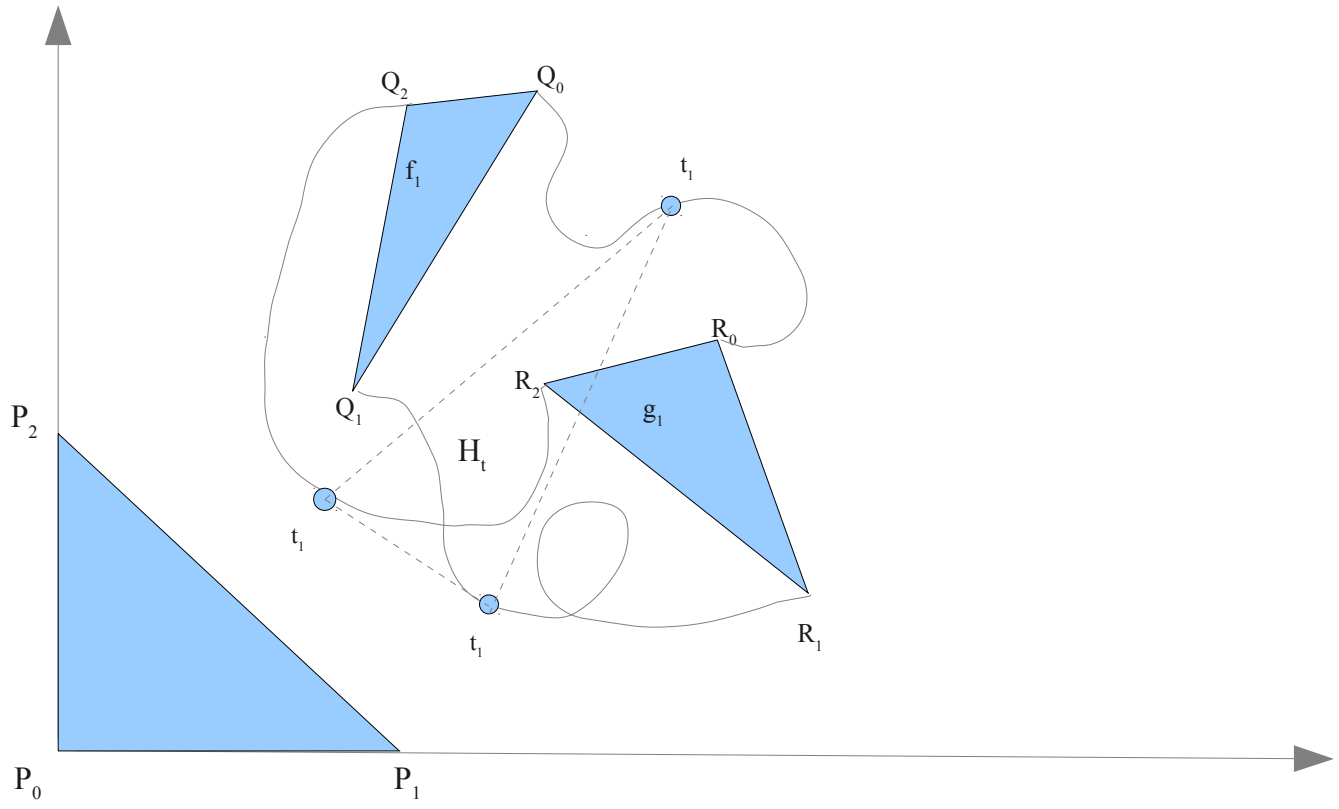
Legyen $f_1, g_1 : \mathbb{R} \to \mathbb{R}$



Az f_1 transzformáció a P_0,P_1,P_2 pontokat viszi át a Q_0, Q_1, Q_2 pontokba, míg a g_1 transzformáció az R_0, R_1, R_2 pontokba. A fraktál deformáció esetén beszélhetünk egy γ függvényről, amely valamilyen síkgörbe mentén, T idő alatt leképezi a Q_0, Q_1, Q_2 pontokat, az R_0, R_1, R_2 pontokra.

γ: $[0,1] \to \mathbb{R}^2$

Egy köztes t_1 időpillanatban, egy köztes H_t fraktál képét tekinthetjük meg. Ezt a következő ábrán szemléltetjük.



Box-dimenzió (Hausdorff-dimenzió, fraktáldimenzió)

A fraktálgeometriában egy módszer, amellyel meg lehet határozni a fraktál dimenzióját egy Euklideszi térben. A box-dimenzió kiszámításához, képzeljük el az adott fraktált egy egyenletesen elosztott négyzethálóban elhelyezve és számoljuk meg, mennyi négyzetre van szükség, hogy befedje a teljes halmazt. Minél sűrűbb ez a négyzetháló, annál jobban nő ez a szám.

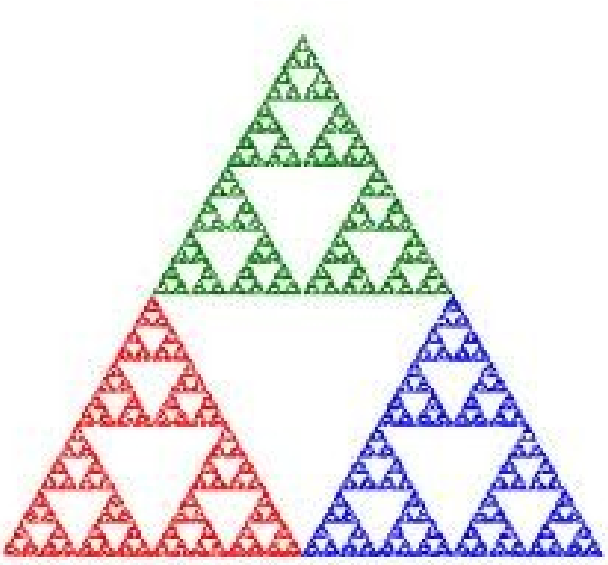
Legyen N(ε) a négyzetek száma, és ε a négyzetek oldalainak hossza. Ekkor a következőképpen határozzuk meg a box-dimenziót:

$$dim_{box}(S) := \lim_{a \to b} \frac{lnN\,\varepsilon}{\ln 1/\varepsilon}$$

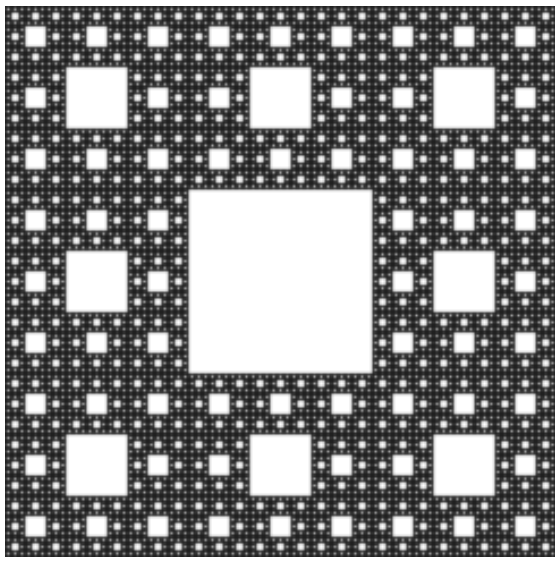
„A fraktál olyan halmaz, aminek a Hausdorff-dimenziója nagyobb, mint a Lebesgue-dimenziója.” – Mandelbrot.

Ahol a vonal Lebesgue-dimenziója egy, a felületé kettő, és így tovább. Ez alapján számítva a fraktálok hossza vagy felszíne végtelen. A Hausdorff-dimenziót szemléletesen az adja, hogy hány példányra van szükség az adott alakzatból ahhoz, hogy kirakjuk az alakzat egy nagyobb példányát. Ez csak szabályos fraktál esetén alkalmazható.

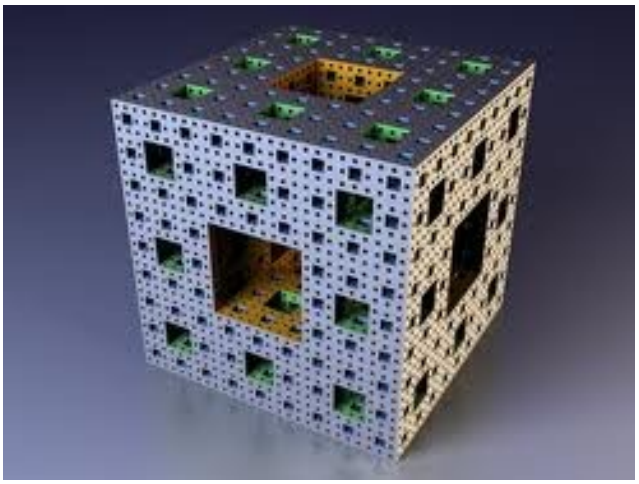
Például a Sierpinski-háromszög, ami önmagának három felére kicsinyített példányából áll, Hausdorff-dimenziója (box-dimenziója) $\frac{ln3}{ln2} \approx 1{,}585$, míg Lebesgue-dimenziója 1.



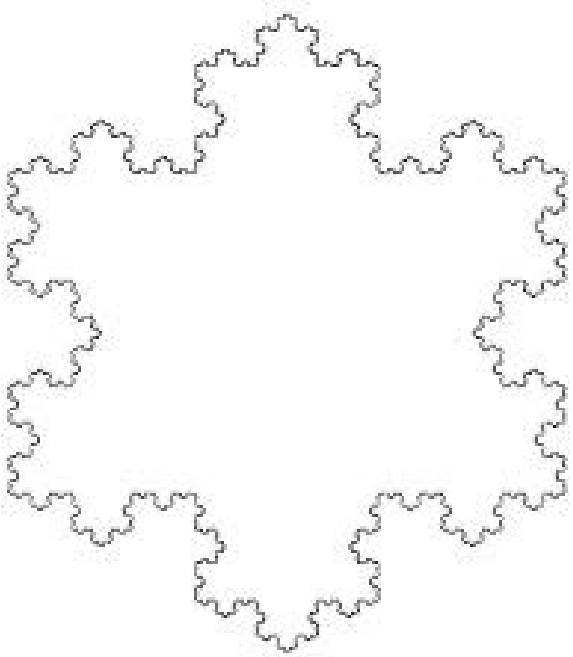
A Sierpinski-szőnyeg box-dimenziója $\frac{ln8}{ln3} \approx 1{,}89$



A Menger-szivacs box-dimenziója $\frac{ln20}{ln3} \approx 2{,}73$



A Koch-hópehely box-dimenziója $\frac{ln4}{ln3} \approx 1{,}26$



Genetikus algoritmusok

Olyan keresési technikák egy osztálya, melyekkel optimumot vagy egy adott tulajdonságú elemet lehet keresni. A genetikus algoritmusok speciális evolúciós algoritmusok, a technikáikat az evolúcióbiológiából kölcsönözték.

A genetikus algoritmusok sokfélék lehetnek, de az alábbi részeket mindig tartalmazzák:

Inicializáció

A kezdeti populációt legegyszerűbb véletlenszerűen generálni. A populáció mérete a probléma természetétől függ, de leggyakrabban néhány száz vagy néhány ezer egyedből áll. Hagyományosan az gyedek a keresési téren egyenletesen oszlanak el, viszont egyes esetekben olyan részeken több egyedet generálnak, ahol sejthető az optimum.

Szaporítás (keresztezés)

Egyedekből újabb egyedeket a kétoperandusú keresztezés (vagy rekombináció) művelettel, és az egyoperandusú mutáció művelettel lehet. Ezeket az operátorokat általában véletlenszerűen alkalmazzák.

Kiválasztás (szelekció)

Az egyedek generálása után kiválasztásra kerül sor. A kiválasztás lehet determinisztikus vagy stochasztikus. Az első esetben szigorúan az egy adott küszöbértéknél jobb állóképességű egyedek maradnak fenn, a stochasztikus kiválasztásnál a rosszabb állóképességű egyedek közül is néhány fennmarad. A stochasztikus megoldás népszerűbb, mert elkerülhető vele a lokális optimumhoz való konvergencia.

Leállás

A genetikus algoritmusok rendszerint addig futnak, amíg egy leállási feltétel nem teljesül. Gyakori leállási feltételek a következők:

- Adott generációszám elérése.
- Ha a legjobb egyed állóképessége már nem javul jelentős mértékben egy-egy iterációval.

Megvalósítás

A következőkben azt fogom leírni, hogy hogyan hozhatunk létre olyan Gtk-s felületet ami tartalmaz egy rajzterületet is amire OpenGl segítségével tudunk rajzolni. Szükségünk lesz a következő függvénykönyvtárak telepítésére:

$ sudo apt-get install libgtk2.0-dev

$ sudo apt-get install libgtkgl2.0-dev

$ sudo apt-get install libgtkglarea-cil-dev

Szükségünk lehet még egy eclipse pluginre ami nagyon megkönyítheti a fordítást. Ez a következő plugin:

pkg-config-support-for-eclipse-cdt

Ezt a következő címről tudjuk telepíteni:

http://petrituononen.com/pkg-config-support-for-eclipse-cdt/update

Ennek a pluginak a letöltése nem kötelező, de nagyon megkönnyíti a fordítást. Ha nem töltjük le akkor a következő paranccsal is tudunk fordítani:

g++ simple.cpp -o simple `pkg-config --cflags --libs gtk+-2.0 gtkgl-2.0`

Ebben az esetben a simple.cpp a forráskód.

Lássunk egy nagyon egyszerű alkalmazást ami Gtk-t és OpenGl-t is használ egyben. A fejlesztéshez Eclipse-t használtam. A main.cpp állományba a következőt írtam:

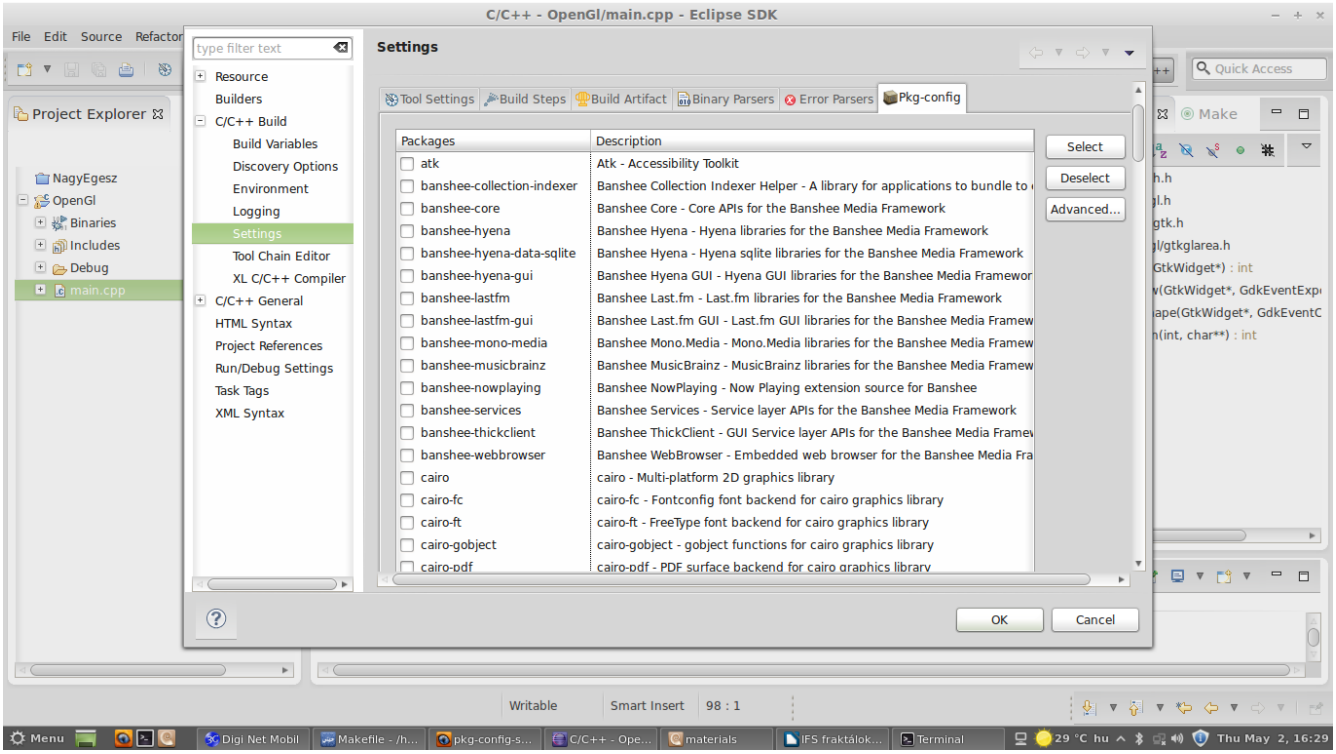
```
#include <math.h>            //Matematikai muveletek
#include <GL/gl.h>           //OpenGl
#include <gtk/gtk.h>         //Gtk kezelofelulet
#include <gtkgl/gtkglarea.h> //Gtkglarea

//A rajzterulet beallitasai
int init(GtkWidget *widget) {
      if (gtk_gl_area_make_current(GTK_GL_AREA(widget) )) {
            glViewport(0, 0, widget->allocation.width, widget-
>allocation.height);
            glMatrixMode(GL_PROJECTION);
            glLoadIdentity();
            glOrtho(0, 100, 100, 0, -1, 1);
```

```
            glMatrixMode(GL_MODELVIEW);
            glLoadIdentity();
      }

      return TRUE;
}

//Rajzolas a rajzpanelre
int draw(GtkWidget *widget, GdkEventExpose *event) {
      if (event->count > 0)
            return TRUE;

      if (gtk_gl_area_make_current(GTK_GL_AREA(widget) )) {
            glClearColor(0, 0, 0, 1);
            glClear(GL_COLOR_BUFFER_BIT);
            glColor3f(1, 1, 1);
            glBegin(GL_TRIANGLES);
            glVertex2f(10, 10);
            glVertex2f(10, 90);
            glVertex2f(90, 90);
            glEnd();
            gtk_gl_area_swap_buffers(GTK_GL_AREA(widget) );
      }

      return TRUE;
}

//Ha az ablakot ujrameretezzuk, akkor ez a fuggveny meghivodik
int reshape(GtkWidget *widget, GdkEventConfigure *event) {
      if (gtk_gl_area_make_current(GTK_GL_AREA(widget) ))
            glViewport(0, 0, widget->allocation.width, widget-
>allocation.height);

      return TRUE;
}

//A foprogram
int main(int argc, char **argv) {
      GtkWidget *window, *glarea;
```

```
    int attrlist[] = { GDK_GL_RGBA, GDK_GL_RED_SIZE, 1, GDK_GL_GREEN_SIZE, 1,
                GDK_GL_BLUE_SIZE, 1, GDK_GL_DOUBLEBUFFER, GDK_GL_NONE };

    //Meghivjuk az init fuggvenyt
    gtk_init(&argc, &argv);
    if (gdk_gl_query() == FALSE)
          return 0;

    //Letrehozunk egy uj ablakot
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    //Megadjuk az ablak cimet
    gtk_window_set_title(GTK_WINDOW(window), "Gtk+Opengl");
    //Beallitjuk a szelek vastagsagat
    gtk_container_set_border_width(GTK_CONTAINER(window), 10);
    //Kilepeskor meghivja a delete_event-t
    g_signal_connect(window, "delete_event", G_CALLBACK (gtk_main_quit), NULL);
    gtk_quit_add_destroy(1, GTK_OBJECT(window) );

    //Letrehozom az OpenGl rajzfeluletet
    glarea = GTK_WIDGET(gtk_gl_area_new (attrlist));
    gtk_widget_set_size_request(GTK_WIDGET(glarea), 100, 100);

    //Az ablakra vontakozo esemenyek eseten a `glarea`-t is ujrarajzoljuk
    gtk_widget_set_events(GTK_WIDGET(glarea),
                GDK_EXPOSURE_MASK | GDK_BUTTON_PRESS_MASK);

    g_signal_connect(glarea, "expose_event", G_CALLBACK(draw), NULL);
    g_signal_connect(glarea, "configure_event", G_CALLBACK(reshape), NULL);
    g_signal_connect(glarea, "realize", G_CALLBACK(init), NULL);

    //Az ablakhoz hozzaadok egy tarolot, majd ehhez a tarolohoz egy rajzfeluletet
    gtk_container_add(GTK_CONTAINER(window), GTK_WIDGET(glarea) );
    gtk_widget_show(GTK_WIDGET(glarea) );
    gtk_widget_show(GTK_WIDGET(window) );

    gtk_main();
    return 0;
}
```

Fordítás előtt a következőket kell beállítanunk:

Project->Properties->C/C++ Build->Settings->Pkg-config menüpont alatt válasszuk ki a: gdk-2.0, glib-2.0, gtk+-2.0, gtk-sharp-2.0,gtkgl-2.0 csomagokat.



Ha az előzőekben nem telepítettük volna az eclipse-hez a pkg-config-support plugint, akkor most egy kicsit nehezebb dolgunk lett volna, mert akkor a Project->Properties->C/C++ Build->Settings->Tools Settings->Cross G++ Compiler->Includes menüpont alatt kellett volna mindent kézzel hozzáadjunk.

Ha ezzel megvagyunk menjunk a Project->Build Project menüpontra ami létrehozza a futtatható állományt. Ezután nincs más dolgunk hátra mint futtatni a programot. Ha mindent jól csináltunk akkor a következő kell megjelenjen.

Könyvészet

http://www.berzsenyi.hu/~dcsonka/matek/Mandelbrot/Mandelbrot/fraktal.htm

http://hu.wikipedia.org/wiki/Mandelbrot-halmaz

http://en.wikipedia.org/wiki/Iterated_function_system

http://hu.wikipedia.org/wiki/Frakt%C3%A1l

http://hu.wikipedia.org/wiki/Hausdorff-dimenzi%C3%B3

http://hu.wikipedia.org/wiki/Sierpinski-h%C3%A1romsz%C3%B6g

http://hu.wikipedia.org/wiki/Affin_transzform%C3%A1ci%C3%B3

http://hu.wikipedia.org/wiki/Transzform%C3%A1ci%C3%B3_(matematika)

http://en.wikipedia.org/wiki/Minkowski%E2%80%93Bouligand_dimension

http://hu.wikipedia.org/wiki/Genetikus_algoritmus