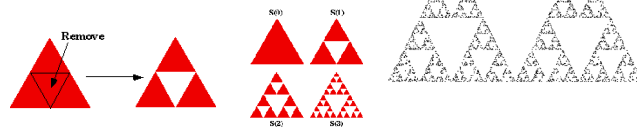


Sierpinski gasket

- ❖ Si parte da un triangolo equilatero
- ❖ Si rimuove quello centrale
- ❖ Si procede ricorsivamente per i tre triangoli rimasti.



1

Sierpinski Gasket

- ❖ Approccio Generativo dell'insieme di punti P che appartengono al gasket:

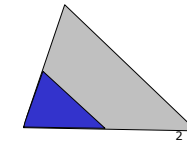
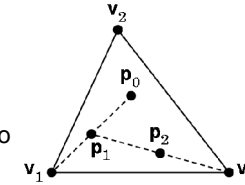
p = punto a caso del triangolo

while true

vi = vertice a caso del triangolo

p = (p+vi)/2

P = P U {p}



Stuttura del programma

Struttura classica dei programmi a linea di comando:

```
main()
{
    init();
    do_my_beautiful_algorithm();
    exit();
}
```

Non ha molto senso per i programmi con un'interfaccia utente.

Come avviene il processo di interazione tra l'utente e l'applicazione?

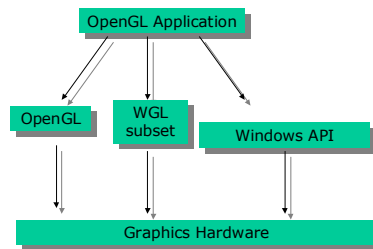
3

Event driven programming

- ❖ Gestione interazione applicazione-utente tramite *callback (message handlers ecc)* o gestione diretta degli eventi del sistema
 - ❖ funzioni che sono attivate in risposta a vari eventi (messaggi) gestiti dal sistema operativo (pressione di un tasto del mouse o della tastiera, reshape della finestra, necessita' di ridisegnare il contenuto della finestra ecc)
 - ❖ Il flusso principale dell'applicazione e' in mano all'utente o meglio al sistema operativo.

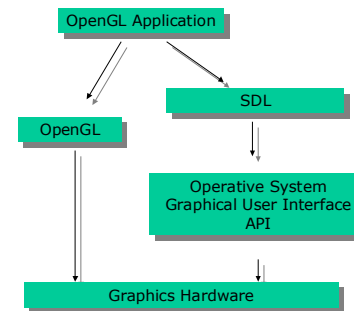
4

Struttura Windows OpenGL App



5

Struttura Applicazione OpenGL+SDL



6

SDL

- ❖ Simple DirectMedia Layer is a cross-platform multimedia library designed to provide level access to audio, keyboard, mouse, joystick, 3D hardware via OpenGL, and 2D video framebuffer.
- ❖ Simple DirectMedia Layer supports Linux, Windows, BeOS, MacOS Classic, MacOS X, ecc.
- ❖ It implements a simple *portable* application programming interface (API) for the windowing part of OpenGL.
- ❖ So you can write a single OpenGL program that works on both Win32 PCs and X11 workstation
- ❖ <http://www.libsdl.org>

7

Schema Minimo Applicazione SDL

- ❖ Inizializzare
 - ❖ `SDL_Init(SDL_INIT_VIDEO)`
 - ❖ `SDL_SetVideoMode(640, 480, 0, SDL_OPENGL)`
- ❖ Gestire il ciclo degli eventi

```
...
while ( ! done ) {
  SDL_Event event;
  SDL_WaitEvent(&event);
  switch(event.type)
  {
    case ...
  }
}
```

Per quello che ci riguarda la cosa piu' importante e' la gestione dell'evento: "quit"

8

La minima applicazione SDL 1

```
#ifndef WIN32
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#endif

#include <GL/gl.h>
#include <GL/glu.h>
#include <stdlib.h>
#include <SDL.h>

void DrawGLScene()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    SDL_GL_SwapBuffers();
}

...
```

9

La minima applicazione SDL 2

```
int main(int argc, char **argv)
{
    SDL_Init(SDL_INIT_VIDEO);
    SDL_SetVideoMode(640, 480, 0, SDL_OPENGL);

    int done = 0;
    while ( ! done ) /* Loop, drawing and checking events */
    {
        DrawGLScene();
        SDL_Event event;
        SDL_WaitEvent(&event);
        switch(event.type)
        {
            case SDL_QUIT      : done = 1;   break ;
            case SDL_KEYDOWN  :
                if ( event.key.keysym.sym == SDLK_ESCAPE )
                    done = 1;
                break;
        }
    }
    SDL_Quit();
    return 1;
}
```

10

Dove si disegna?

❖ Ricordate la pipeline di rendering



- ❖ La prima cosa che fa il renderer e' di spostare tutto nel sistema di riferimento della camera
- ❖ Poi taglia quel che non si vede
- ❖ Infine appiattisce il mondo sul piano di vista

11

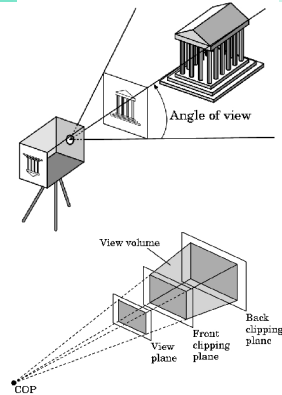
Trasformazioni e Clipping

- ❖ Noi non abbiamo definito nessuna trasformazione quindi vedremo solo quello che si trova *davanti* alla camera.
- ❖ Più precisamente vedremo quello che si trova nel *Volume di Vista*:
Porzione di spazio, nel sistema di riferimento della camera, che e' visibile dalla camera.

12

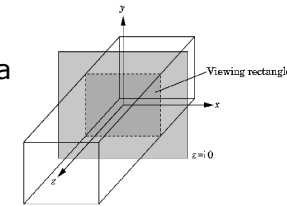
Volume di Vista

- ❖ Normalmente ci si aspetta che il volume di vista sia una piramide infinita.
- ❖ Per ragioni di praticità si aggiungono due piani (front and back o near and far) che ulteriormente delimitano lo spazio d'interesse, e quindi il volume di vista è un tronco di piramide.



Proiezione Ortografica

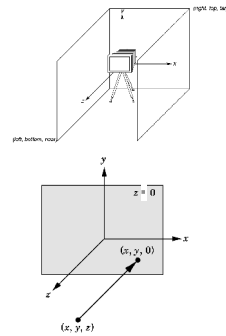
- ❖ Se ci immaginiamo la camera posta ad una distanza infinita il volume di vista diventa un parallelepipedo.
- ❖ Questo genere di vista è detto **ortogonale**



14

Proiezione Ortografica

- ❖ In una proiezione ortografica tutti i punti nel volume di vista vengono semplicemente proiettati perpendicolarmente sul piano di vista.



15

Disegno del sierpinski set

- ❖ Questioni principali
 - ❖ Come si disegna un insieme di punti
 - ❖ Dove si disegna?

16

Come si disegna in OpenGL

- ❖ Disegnare significa definire una scena da far passare nella pipeline

glBegin(Primitiva)

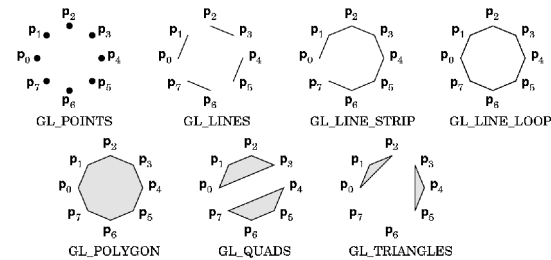
Dati della primitiva (Coordinate vertici, e attributi vari)

glEnd()

- ❖ Le coordinate dei vertici si specificano con il comando
- ❖ glVertex*();

17

Primitive OpenGL



18

Note Su OpenGL

Note

- ❖ OpenGL e' il layer di base
- ❖ GLU e'un insieme di funzioni di utility costruite sopra OpenGL, piu'comode da usare
- ❖ GLUT o SDL sono il toolkit di interfaccia con il sistema operativo
- ❖ Wgl e GLx sono i sottoinsiemi di OpenGL che dipendono dal SO e che permettono di dire al SO ad esempio che l'interno di una certa finestra deve essere *adatto* a OpenGL. Spesso nascosto dal layer
- ❖ Tutto quanto sopra e' C (e non C++).

19

GL syntax

- ❖ Tutte le funzioni di Opengl si chiamano
- ❖ glSomethingXXX
- ❖ Dove XXX specifica (numero e) il tipo dei parametri:
- ❖ glColor3f(float, float, float)
 - f: float
 - d: double ecc.
- ❖ Non e' C++...

20

Disegnare il sierpinski gasket

- ❖ La generazione e' facile, quindi si può evitare di memorizzare e disegnare durante il processo di generazione.

```

GLfloat triangle[3][2]={
    { -1.0f, -1.0f},{ 1.0f, -1.0f},{ 0.0f, 1.0f}    };

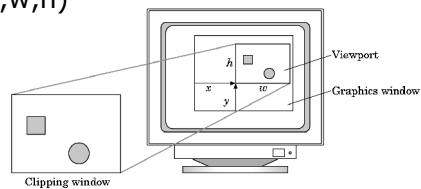
GLfloat p[2]={0.0f, 0.0f};
int i, j;
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POINTS);
for(j=0;j<20000;j++)
{
    i=rand()%3;
    p[0]=(p[0]+triangle[i][0])/2.0f;
    p[1]=(p[1]+triangle[i][1])/2.0f;
    glVertex2f(p[0],p[1]);
}
glEnd();
    
```

21

Viewport

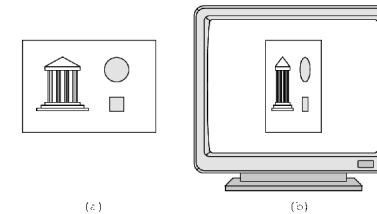
- ❖ Di default si disegna su tutto la finestra, ma si può specificare una sottoporzione rettangolare della finestra (contesto) su cui si disegna

`glViewport(x,y,w,h)`



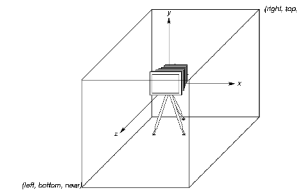
Gestione Reshape

- ❖ Il comportamento di default e' che tutto il volume di vista viene mappato nella finestra.
- ❖ Aspect Ratio sbagliata



Adattare la camera alla finestra

- ❖ In opengl una vista ortogonale, si specifica definendo il view volume `glOrtho(left,right,bottom,top,near,far);`



24

Adattare la camera alla finestra

❖ Il View Volume deve avere le stesse proporzioni della finestra

❖ In SDL:

❖ E dare la possibilità di ridimensionare la finestra

```
SDL_SetVideoMode(640,480,0, SDL_OPENGL | SDL_RESIZABLE)
```

❖ Si deve gestire il messaggio di resize

❖ Nota: in sdl si deve anche rifare setvideomode quando arriva il msg di resize.

```
case SDL_VIDEORESIZE :
    SDL_SetVideoMode(event.resize.w,event.resize.h,
        0, SDL_OPENGL | SDL_RESIZABLE);
    myReshapeFunc(event.resize.w,event.resize.h);
```

25

Adattare la camera alla finestra

```
void myReshapeFunc(GLsizei w, GLsizei h)
{
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    float ratio=(float)h/(float)w;
    glOrtho(-1,1,-ratio,ratio,-1,1);
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_MODELVIEW);
}
```

26

O ancora meglio

```
void myReshapeFunc(GLsizei w, GLsizei h)
{
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    float ratio=(float)h/(float)w;
    if(ratio>=1)
        glOrtho(-1,1,-ratio,ratio,-1,1);
    else {
        ratio=1.0/ratio;
        glOrtho(-ratio,ratio,-1,1,-1,1);
    }
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode (GL_MODELVIEW);
}
```

27

Gestire meglio il disegno

❖ Ridisegna la scena troppe volte

❖ Gestiamo anche l'evento "necessita' di ridisegnare"

❖ Nel ciclo degli eventi:

```
case SDL_VIDEOEXPOSE :
    DrawGLScene();
    break;
```

❖ E togliamo il `DrawGLScene` dal ciclo degli eventi

28