# Laboratori Gràfics

- 7.5 sessions "gràfics" i interacció directa organitzades en 4 blocs + Examen laboratori (22 d'Abril 2014) (20%)
- 3.5 sessions "interficies en Qt"+ Examen (3 Juny 2014) (5%)
- Projecte Android: Sortiran enunciats aplicacions (7 de Juny 2014) (15%)

- Cal haver lliurat l'aplicació resultat de 3 dels 4 blocs del laboratori de "gràfics" per a poder presentar-se a l'examen del 22 d'abril.

- Bloc 1 (1 sessió): 10 minuts inicials de la primera classe del Bloc 2
  - Opcional: Millora quan es lliure el bloc 2.
- Bloc 2 (2 sessions): 10 minuts inicials de la primera classe del Bloc 3
- Bloc 3 (3 sessions): 10 minuts inicials de la primera classe del Bloc 4
- Bloc 4 (1.5 sessions): el dia abans de l'examen laboratori de gràfics



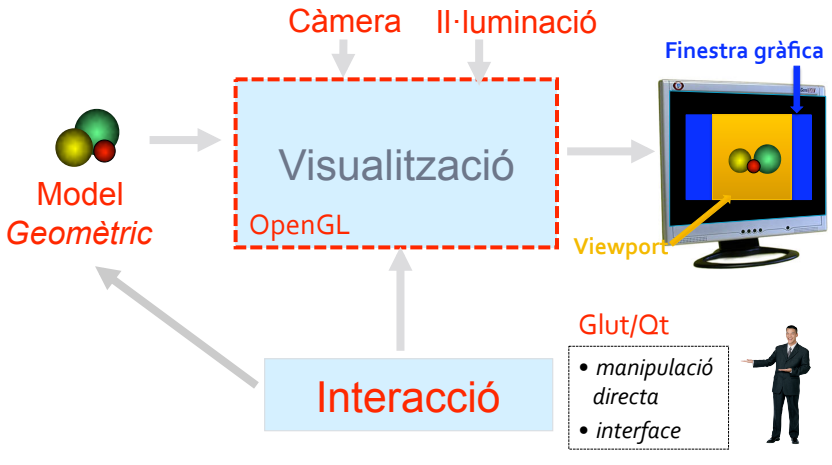
# Repetidors

- Us recomanem que vingueu a classe
- Convalidacions (Només si nota >= 7.0):
  - ProvaQt
  - Nota Projecte Android (ProjecteLab)
  - Nota OpenGL (ProvaGL)
- Cal demanar-ho via Racó:
  - Abans del dia 2 de Març
  - Demanar què es vol convalidar
  - El que no es demani, s'entén que no es vol convalidar
    - Es contestarà al Racó mateix

# Sessió 1 de laboratori



# Elements Sistema Gràfic

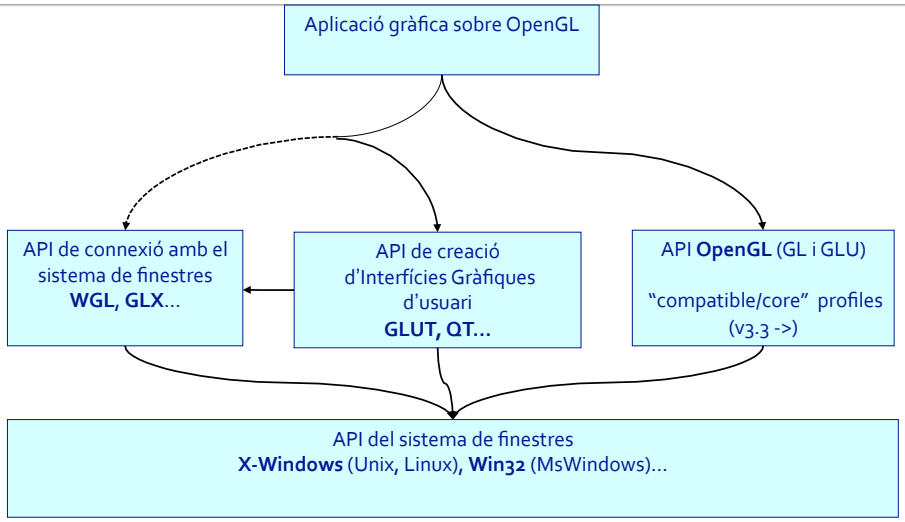


# Introducció

- **Què és OpenGL?**
  - OpenGL és una **API per a la programació d'aplicacions gràfiques 2D i 3D** que actua com a interfície amb el *hardware* gràfic.
  - L'API OpenGL és multiplataforma gràcies a la seva **independència del sistema de finestres**.
  - La interfície consisteix en una biblioteca de funcions (C/C++) que permeten la visualització d'una escena 3D.
  - **No gestiona "inputs"** d'usuari.



# Arquitectura bàsica d'una aplicació OpenGL



# Objectius sessió

- Mínima introducció a OpenGL i glut.
- Entendre les parts d'una aplicació glut i practicar registre de *callbacks* i tractament d'events.
- Desenvolupar una primera aplicació.

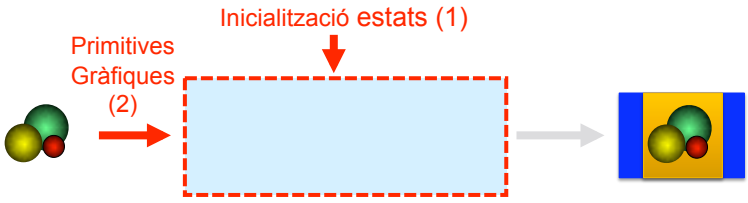




# Introducció a OpenGL

# Característiques d'OpenGL

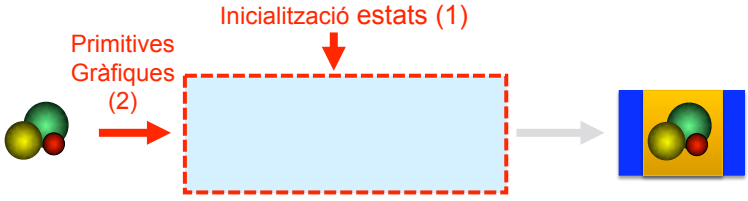
- Generals:
  - Entorn d'execució d'OpenGL: Client/Servidor
  - OpenGL és **independent del hardware**.
  - Es comporta com una **màquina d'estats**
  - **No proporciona** una estructura de representació per als objectes 3D. Simplement els pinta.





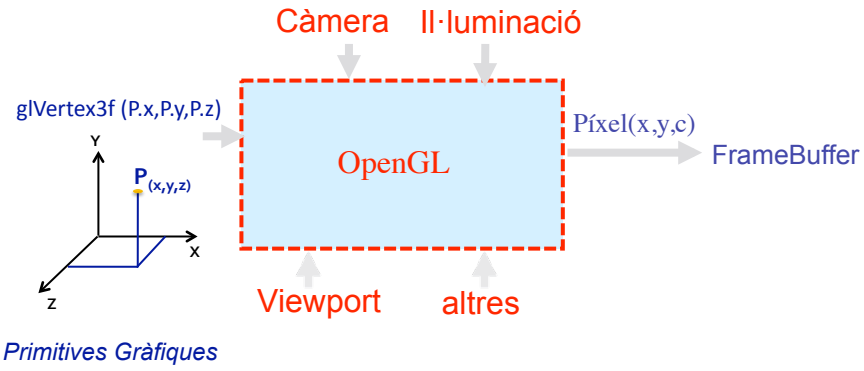

# Característiques d'OpenGL

- Les funcions d'OpenGL són de tres tipus:
  - Permeten modificar una variable d'estat (1):
    - Color, material, fonts de llum, ***matrius*** *-camera-*
  - Permeten consultar el valor d'una variable d'estat:
    - glGetBooleanv(),..
  - Permeten modificar de forma directa la informació visible (2).
    - Per exemple, el dibuixat de primitives gràfiques



# Repàs: OpenGL

- Màquina d'estats

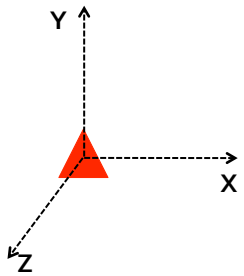





# Primitives gràfiques: exemple

- Les primitives geomètriques es dibuixen usant blocs **glBegin-glEnd**:

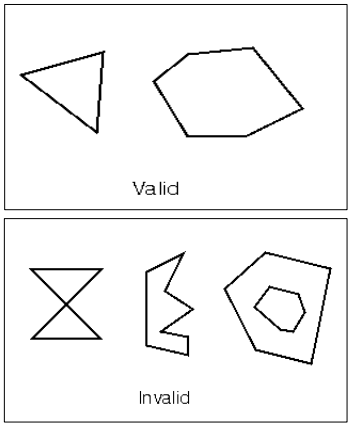
```
glBegin(GL_TRIANGLES);
    glColor3f (1,0,0);
    glVertex3f(-0.5,-0.5,0.0);
    glVertex3f(0.5,-0.5,0.0);
    glVertex3f(0.0,0.5,0.0);
glEnd();
```




En quina orientació? => funció de la càmera definida
En quin lloc de la finestra gràfica ho pinta? => funció del viewport

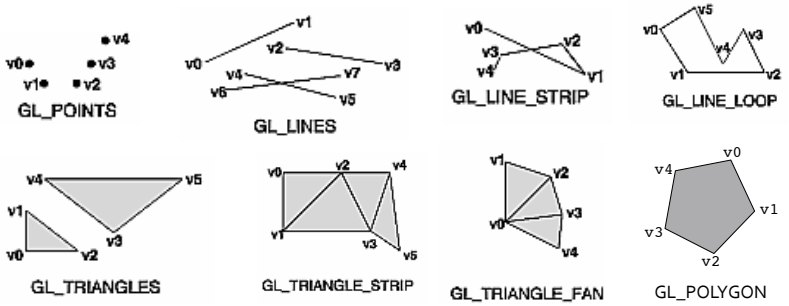
# Primitives gràfiques

- OpenGL soporta quatre primitives gràfiques:
  - Punts
  - Línies poligonals
  - Polígons simples convexos
  - Bitmaps






# Primitives gràfiques




```
glBegin(.....);
      glColor3f (1,0,0);
      glVertex3f(-0.5,-0.5,0.0);
      glVertex3f(0.5,-0.5,0.0);
      glVertex3f(0.0,0.5,0.0);
glEnd();
```

# Primitives gràfiques

| Tipus de primitiva | Significat |
|---|---|
| GL_POINTS | Punts independents |
| GL_LINES | Segment independent (2 vert.) |
| GL_LINE_STRIP | Poligonal |
| GL_LINE_LOOP | Polígon tancat. |
| GL_TRIANGLES | Triangle independent (3 vert.) |
| GL_TRIANGLE_STRIP | Triangle strip |
| GL_TRIANGLE_FAN | Triangle fan |
| GL_QUADS | Quadrilàters indep. (4 vert.) |
| GL_QUAD_STRIP | Quad strip |
| GL_POLYGON | Polígon simple i convex |



# Sintaxi de les comandes OpenGL

- Una mateixa funció adopta diferents prototipus (*poor man's mangling*) que varien únicament en:
  - Nombre d'arguments
  - Tipus dels arguments (*int*, *float*, *double*...)
  - Adreçament (referència o valor)

## Sintaxi de les comandes OpenGL. Exemple

void glVertex3d (GLdouble *x* , GLdouble *y* , GLdouble *z* );
void glVertex3f (GLfloat *x* , GLfloat *y* , GLfloat *z* );
void glVertex3i (GLint *x* , GLint *y* , GLint *z* );
void glVertex3s (GLshort *x* , GLshort *y* , GLshort *z* );
void glVertex3dv (const GLdouble *v );
void glVertex3fv (const GLfloat *v );
void glVertex3iv (const GLint *v );
void glVertex3sv (const GLshort *v );



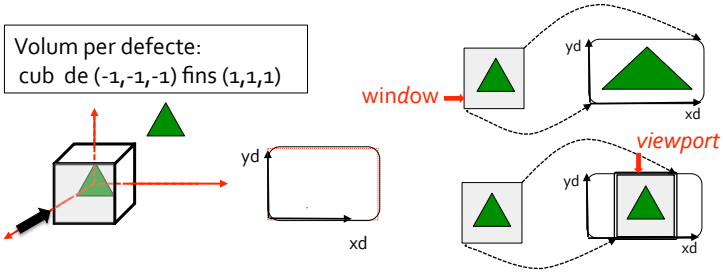
## Funcions per modificar l'estat

- OpenGL té valors per defecte de les variables d'estat:
  - Color de pintat: `glColor3f (r,g,b);`
  - Càmera
  - Focus de llum
  - Viewport
  - Netejar Buffers
  - Color del fons de la pantalla
  - ...

# Funcions per modificar l'estat

- De moment utilitzarem la càmera (volum de visió) per defecte:

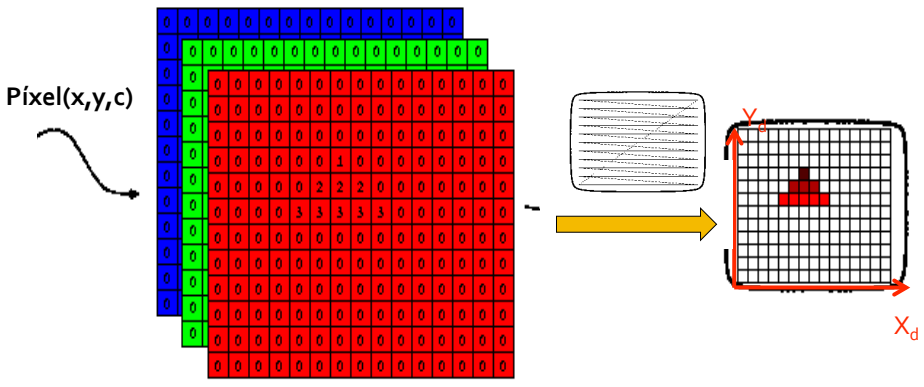
Volum per defecte:
cub de (-1,-1,-1) fins (1,1,1)




- Definició del *viewport* => `glViewport(xmin,ymin,xmax,ymax)`
  Ha de tenir la mateixa relació d'aspecte que el window per a no tenir deformacions.
  Per defecte és tota la pantalla
  (pista pel futur... en glut cal definir el viewport en el *callback* de glutReshapeFunc())



# Frame buffer

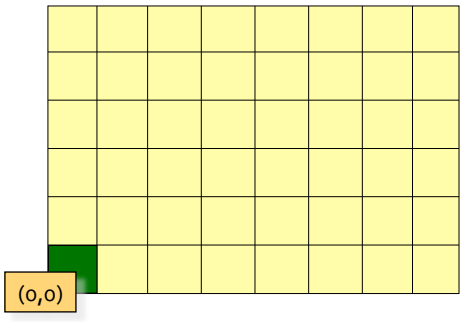



taula [res_x][res_y] de color

Pinta en un *Buffer*:
Definit per la seva resolució (n x m) –finestra gràfica- i
la seva profunditat (nombre de bits/pixel)

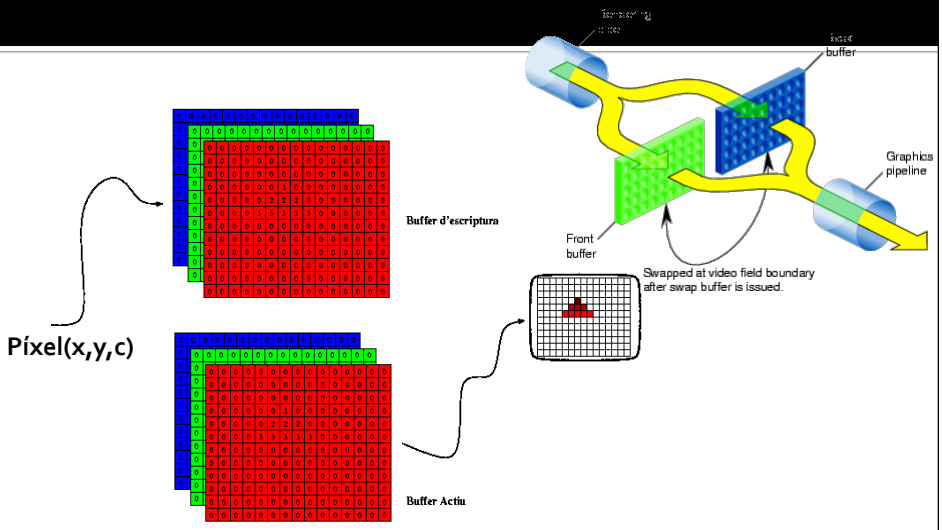
# Característiques d'OpenGL

- OpenGL considera l'origen del SC dispositiu a la cantonada inferior esquerra.



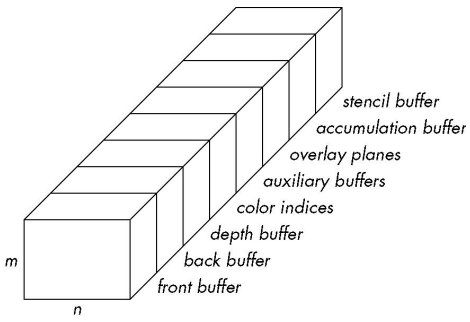


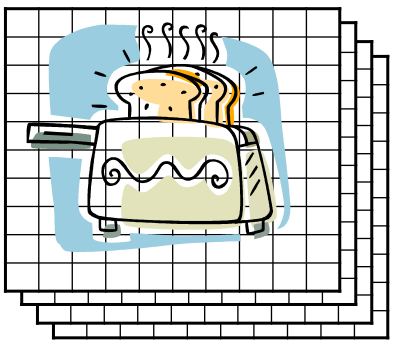

# Doble buffer



# Característiques d'OpenGL

Multi-layered framebuffer





# Altres funcions bàsiques de dibuix

- Netejar el buffer
  - Definir color neteja:
    **glClearColor(red, green, blue, alpha);**
  - Fer efectiva la neteja:
    **glClear(GL_COLOR_BUFFER_BIT);**
  - Podem netejar més d'un buffer:
    **glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);**

# Primer exemple de pintat

```
void refresh (void)
{  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glBegin(GL_TRIANGLES);
          glVertex3f(-0.5,0.0,0.0);
          glVertex3f(0.5,0.0,0.0);
           glVertex3f(0.0,0.5,0.0);
    glEnd();
   glutSwapBuffers();
}
```





# Introducció a Glut

# Introducció glut

- GLUT: OpenGL Utility Toolkit
  - Biblioteca de funcions implementada per Mark J. Kilgard
  - Construcció d'aplicacions d'OpenGL independents del sistema de finestres
    - Implementada inicialment per X windows, i portada a Microsoft Windows per Nate Robins
  - Útil per a construir petites aplicacions en OpenGL i aprendre
  - Continguts extrets del tutorial de Lighthouse3d a

    www.lighthouse3d.com



# Introducció glut

- Les distribucions de glut vénen amb molts exemples
- Recordeu utilitzar el Manual de glut.
- Hi ha versions de codi lliure:
  - freeGLUT (http://freeglut.sourceforge.net)
  - OpenGLUT (http://openglut.sourceforge.net)

# Primer exemple

```
#include <GL/gl.h>
#include <GL/freeglut.h>
void refresh (void)
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
        glVertex3f(-0.5,0.0,0.0);
        glVertex3f(0.5,0.0,0.0);
         glVertex3f(0.0,0.5,0.0);
   glEnd();
  glutSwapBuffers();
}
int main(int argc, const char *argv[])
{  glutInit(&argc, ((char **) argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
  glutInitWindowSize(600,600);
  glutCreateWindow("IDI: Practiques OpenGL");
  glutDisplayFunc (refresh);
  ...
 glutMainLoop();
 return 0;
}
```

1. inicialitzacions

2. Registre *callbacks*

3. Bucle processament events



# Inicialitzacions

- Totes les funcions de *glut* tenen prefix *glut*
- Les que fan quelcom relacionat amb inicialitzacions comencen per *glutInit*

- Inicialització bàsica:

  ***void glutInit(int *argc, char **argv);***

  - Inicialitza glut
  - Paràmetres:
    - *argc* – Punter a la variable *argc* del programa principal
    - *argv* – Punter a la variable *argv* del programa principal

# Inicialitzacions: configuració de finestres

***void glutInitWindowPosition(int x, int y);***

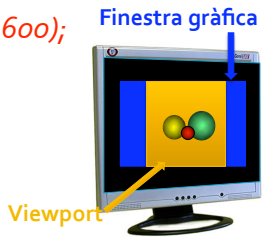
- Suggereix la posició de la finestra gràfica en pantalla
- Paràmetres:
  - *x* – Distància en píxels del costat esquerre. Un valor per defecte de -1 indica que el gestor de finestres la posi on consideri. El gestor de finestra pot no fer cas d'aquest valor ☹.
  - *y* – Ídem anterior en vertical



# Inicialitzacions: configuracions de finestres

***void glutInitWindowSize(int sizex, int sizey);***

- Determina la mida de la finestra gràfica
- Paràmetres:
  - *sizex* – Mida en horitzontal.
    El gestor de finestres pot no fer-ne cas
  - *sizey* – Ídem anterior en vertical
- Exemple: *glutInitWindowSize (600,600);*



# Inicialitzacions: definició del context d'OpenGL

***void glutInitDisplayMode(int mode);***

- Determina els buffers que conformen el context OpenGL
  - GLUT_RGB o GLUT_RGBA determina un buffer RGBA
  - GLUT_SINGLE, GLUT_DOUBLE: Un o dos buffers de pintat
  - GLUT_DEPTH: buffer de profunditat
  - GLUT_STENCIL: utilitza el buffer de stencil
- Exemple:

*glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);*



# Inicialitzacions: creació de finestres

**void glutCreateWindow(char* nom);**

- Crea la finestra amb la configuració determinada i el títol que li passem.
- Exemple:

*glutCreateWindow("IDI: Practiques OpenGL");*

# Inicialitzacions: exemple

- Integrant les inicialitzacions:

```
int main(int argc, char **argv)
{
  // init GLUT and create Window
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE |GLUT_RGBA);
  glutInitWindowPosition(100,100);
  glutInitWindowSize(600,600);
  glutCreateWindow("IDI: Practiques OpenGL")

  // register callbacks
 // enter GLUT event processing cycle
}
```



# Registre de callbacks

- *glut* permet definir funcions –***callbacks***- que atendran els events del sistema (ratolí, teclat...).
- El procés d'associació de rutines d'atenció als events. s'anomena **registre.**
- Exemple:

  **void glutDisplayFunc(void (*funcName)void);**
  - Registra la funció de pintat.
  - S'executa cada vegada que glut detecta que el contingut de la finestra s'ha de repintar.
  - El paràmetre conté un punter a la funció de pintat que no requereix cap paràmetre

# Primer exemple

```
#include <GL/gl.h>
#include <GL/freeglut.h>
void refresh (void)
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
        glVertex3f(-0.5,-0.5,0.0);
        glVertex3f(0.5,0.0,0.0);
        glVertex3f(0.0,0.5,0.0);
   glEnd();
  glutSwapBuffers();
}
int main(int argc, const char *argv[])
 { glutInit(&argc, ((char **) argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
  glutInitWindowSize(600,600);
  glutCreateWindow("IDI: Practiques OpenGL");
  glutDisplayFunc (refresh);
  ...
 glutMainLoop();
 return 0;
}
```

1. inicialitzacions

2. Registre *callbacks*

3. Bucle processament events

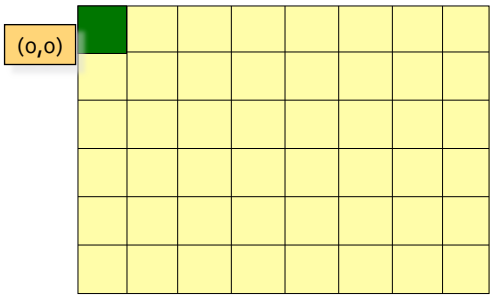


# Registre de callbacks: exemples

- void glutDisplayFunc (void (*funcName)void);
- void glutReshapeFunc (void (*func)(int width, int height));
- void glutKeyboardFunc (void (*func) (unsigned char key, int x, int y));
- void glutMouseFunc (void (*func)(int button, int state, int x, int y));
- void glutMotionFunc (void (*func) (int x,int y)
- … altres que anirem veient
- *glutPostRedisplay();*

# Consideració IMPORTANT

- glut considera l'origen del SC dispositiu a la cantonada superior esquerra de la finestra gràfica.






# Processament d'events

- Una vegada s'ha configurat tot i els callbacks estan registrats, cal entrar en el bucle de processament d'events:

**void glutMainLoop(void)**

# Primer exemple

```
#include <GL/gl.h>
#include <GL/freeglut.h>
void refresh (void)
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glBegin(GL_TRIANGLES);
        glVertex3f(-0.5,-0.5,0.0);
        glVertex3f(0.5,0.0,0.0);
        glVertex3f(0.0,0.5,0.0);
   glEnd();
  glutSwapBuffers();
}
int main(int argc, const char *argv[])
 { glutInit(&argc, ((char **) argv);
  glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA);
  glutInitWindowSize(600,600);
  glutCreateWindow("IDI: Practiques OpenGL");
  glutDisplayFunc (refresh);
  ...
 glutMainLoop();
 return 0;
}
```

1. inicialitzacions

2. Registre *callbacks*

3. Bucle processament events



# Més informació

- Links:
  - http://www.opengl.org/resources/libraries/glut/
  - http://freeglut.sourceforge.net
  - http://openglut.sourceforge.net
- Alternatives a glut:
  - Nui: http://libnui.net
  - CPW: http://mathies.com/cpw/about.html
  - ...

# Bloc 1



# Bloc 1: visualització bàsica

1. - Editar codi (crea finestra), makefile, executar, entendre ordre.
   - Implementar:
     - InitGL() inicialitzant el color de fons: `glClearColor (r,g,b);`
     - Callback *refresh ()* amb neteja buffers

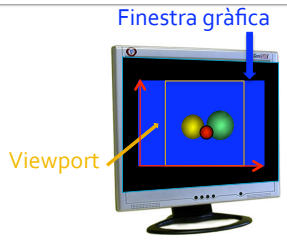
```
void refresh (void)
{ glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glutSwapBuffers();
}
```

2. - Completar refresh() per pintar un triangle

# Bloc 1: visualització bàsica

2. - Completar refresh() per pintar un triangle

   - Registrar i implementar el callback "resize"




```
glutReshapeFunc(resize)

void resize (int w, int h){
   …
   glViewport (xvmin, yvmin, w_v,h_v);
}
```

per a evitar deformacions al modificar la finestra gràfica i generar el dibuix optimitzant l'espai en la finestra ➔ modificar viewport : glViewport(…)

- Modificar les coordenades dels vèrtexs i analitzar l'efecte de "retallat" si algun vèrtex del triangle no està dins del volum visió de defecte.



# Bloc 1: interacció bàsica

3. Esdeveniments ratolí i callbacks associats
   - **glutMouseFunct (…)**
     callback funció amb 4 paràmetres: botó, press/un_press, posició ratolí)
   - **glutMotionFunc(…)**
     callback funció amb 2 paràmetres: posició ratolí
   - ➡ a l'arrosegar el ratolí, el color de fons es faci més fosc o més clar

```
void glutMouseFunc(void(*func)(int button,int state,int x,int y));
void glutMotionFunc (void (*func) (int x,int y)

      glutMouseFunc(iniraton);

      void iniraton(int boto,int state,int xclic,int yclic){
        …
        glutPostRedisplay();
      }
```

# Bloc 1: interacció bàsica

4. Esdeveniment teclat i callbacks associats:
   callback funció amb tres paràmetres: carácter, posició ratolí
   ➡escriure help al premer "h"
   ➡ si ESC (ASCII 27) es tanqui aplicació

```
void glutKeyboardFunc(void (*func)(unsigned char key,int x,int y));
```

```
glutKeyboardFunc (teclat);

void teclat(unsigned char c, int x, int y){
    if (c=='h')….;
    glutPostRedisplay();
}
```