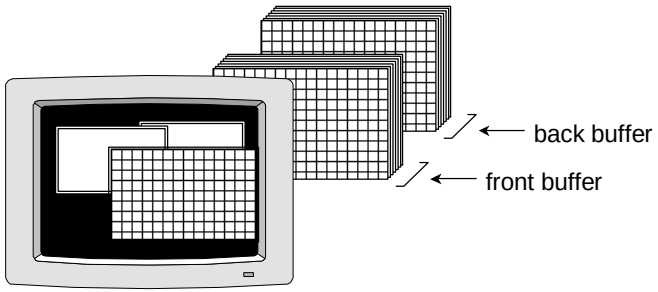
- INTRODUZIONE
- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING



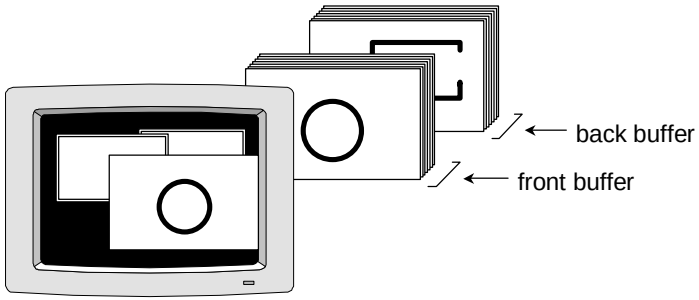
# DOUBLE BUFFERING

- serve a nascondere la fase di drawing
- utilizzato soprattutto nelle animazioni

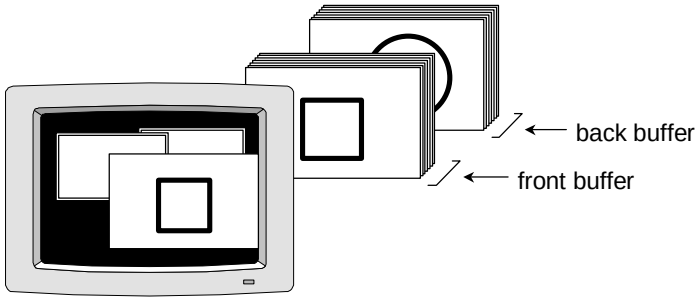



- due color buffer: “front” e “back”
- viene visualizzato solo il front buffer mentre il back buffer e` “nascosto”
- i due buffer possono essere scambiati

# DOUBLE BUFF. (cont.)




- il drawing si effettua nel back buffer
- quando il drawing e` completato i buffer vengono scambiati






# ESERC.: DOUBLE BUFF.

- animazione SENZA double buffering

```
void redraw( void )
{
static GLfloat angle = 0.0;

   glMatrixMode( GL_MODELVIEW );
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glColor3f( 0.0, 0.0, 1.0 );
   triangle();

   glColor3f( 0.0, 1.0, 0.0 );
   glPushMatrix();
   glTranslatef( 1.5, 0.0, 0.0 );
   glRotatef( angle, 0.0, 0.0, 1.0 );
   triangle();
   glPopMatrix();

   glColor3f( 1.0, 0.0, 0.0 );
   glPushMatrix();
   glRotatef( -angle, 0.0, 0.0, 1.0 );
   glTranslatef( 1.5, 0.0, 0.0 );
   triangle();
   glPopMatrix();

   glFlush();

   angle += 1.0;
}
void main( int argc, char** argv )
{
   ...
   glutReshapeFunc ( resize );
   glutDisplayFunc ( redraw );
   glutIdleFunc    ( redraw );
   glutMainLoop    (        );
}
```

# OpenGL DOUBLE BUFF.

- la gestione del double buffering dipende dal sistema utilizzato
- alternative possibili:
  - utilizzare la gestione fornita dal sistema specifico
  - utilizzare una libreria che fornisca un livello di astrazione (es. glut)
- modalita` double buffering (glut):

  specificare la flag **`GLUT_DOUBLE`** nella funzione `glutInitDisplayMode()`
- per scambiare i buffer (glut):

  `void` **`glutSwapBuffers`**`(void)`



# ESERC.: DOUBLE BUFF. (2)

- animazione CON double buffering
- esercizio: aggiungere il double buffering all'esempio precedente
- ricordare di:
  - abilitare il double buffering specificando la flag **`GLUT_DOUBLE`** nella funzione `glutInitDisplayMode()`
  - scambiare i buffer con **`glutSwapBuffers`**`();`
- soluzione:

```
void draw()
{
   ...
   glutSwapBuffers();
}
int main( int argc, char** argv )
{
 glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
   ...
}
```

- INTRODUZIONE
- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING

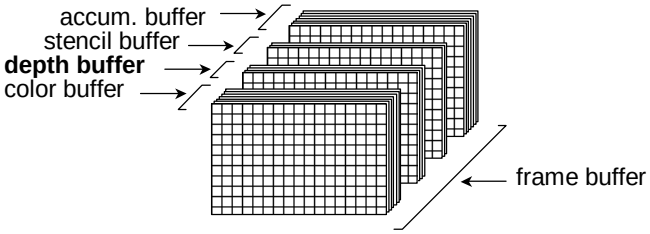


# Z-BUFFERING

- un algoritmo utilizzato per la rimozione delle parti nascoste della scena
- le parti nascoste dipendono dal punto di vista
- basato sul confronto della coordinata z (in eye coordinate) associata ai pixel
- viene applicato dopo la rasterizzazione e la mappatura sulla viewport

# DEPTH BUFFER

- in OpenGL lo z-buffer si chiama **depth buffer**




- il colore di un pixel viene posto nel color buffer (ed il valore di z nel depth buffer) solo se il valore di z del pixel precedentemente in tale posizione e` inferiore
- occorre una inizializzazione opportuna del depth buffer



# ESERC.: Z-BUFFER

- drawing SENZA z-buffer

```
void redraw( void )
{
   ...
   glBegin( GL_TRIANGLES );
      glColor3f( 0.1, 0.1, 0.8 );
      glVertex3f( -30.0, -20.0,  -5.0 );
      glVertex3f( -20.0,  30.0,   5.0 );
      glVertex3f(  20.0, -25.0,  16.0 );

      glColor3f( 0.1, 0.8, 0.1 );
      glVertex3f( -40.0, -10.0, 4.0 );
      glVertex3f(   0.0,  30.0, 5.0 );
      glVertex3f(  20.0,  -5.0, 6.0 );
   glEnd();
   ...
}
```

- osservare cosa accade invertendo l’ordine dei triangoli

# DEPTH BUFFER (cont.)

- per selezionare la modalita` specificare la flag **`GLUT_DEPTH`** in `glutInitDisplayMode()`
- per inizializzare il depth buffer specificare la flag **`GL_DEPTH_BUFFER_BIT`** in `glClear()`
- per abilitare lo z-buffering

  **`glEnable( GL_DEPTH_TEST )`**
- per disabilitare lo z-buffering

  **`glDisable( GL_DEPTH_TEST )`**



# ESERC.: Z-BUFFER (2)

- drawing CON z-buffer

```
void redraw( void )
{
   ...
   glClear( GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT );
   glEnable( GL_DEPTH_TEST );
   ...
}

int main( int argc, char** argv )
{
   glutInitDisplayMode( GUT_RGB | GLUT_DEPTH );
   ...
}
```

- osservare cosa accade invertendo l’ordine dei triangoli

# ESERC.: ROTAZIONI

- ruotare interattivamente la scena in tutte le direzioni
  - interagire a piacere con tastiera e/o mouse
  - ruotare tutta la scena sui tre assi
  - double buffering e z-buffering



# ESERC.: ROTAZIONI (cont.)

- `void` **`glutPostRedisplay`**`( void )`
  genera un evento di “expose” (causa il ridisegno della scena non appena possibile)

- suggerimenti:
  - definire 3 variabili globali corrispondenti all'angolo di rotazione sui tre assi
  - legare il valore delle tre variabili a tastiera e/o mouse
  - gestire in modo opportuno lo stack di matrici "modelview"
  - utilizzare i comandi per la manipolazione delle matrici

## ESERC.: ROTAZIONI (cont.)

```
static GLfloat angle_x = 0.0;
...
void key( unsigned char k, int x, int y )
{  switch( k ) {
   case '1': angle_x  += 5.0; break;
   case '2': angle_x  -= 5.0; break;
   ...
   case ESC: exit(0); }
   glutPostRedisplay();
}
void redraw( void )
{  ...
   glClear( GL_COLOR_BUFFER_BIT |
            GL_DEPTH_BUFFER_BIT );
   glEnable( GL_DEPTH_TEST );

   glPushMatrix();
   glRotatef( angle_x, 1.0, 0.0, 0.0 );
   glRotatef( angle_y, 0.0, 1.0, 0.0 );
   glRotatef( angle_z, 0.0, 0.0, 1.0 );

   /* drawing here */

   glPopMatrix();
   glutSwapBuffers();
}
void main( int argc, char** argv )
{
   glutInitDisplayMode( GLUT_RGB |
                        GLUT_DEPTH | GLUT_DOUBLE );
   ...
   glutKeyboardFunc( key );
   ...
}
```



## ES.: ROTOTRASLAZIONI

- modificare l’esercizio precedente
- rototraslare interattivamente una parte della scena rispetto al resto
  - interagire a piacere con tastiera e/o mouse
  - double buffering e z-buffering
- suggerimenti:
  - gestire in modo opportuno lo stack di matrici "modelview"

# VARIABILI DI STATO

- e` possibile leggere il contenuto delle variabili di stato:

```
void glGet*( <name>, <result> )
```

  `<name>` e` il nome della variabile letta

  `<result>` e` la variabile scritta

- segnature possibili:

```
Booleanv, Integerv,
Floatv,   Doublev
```

- esempi:

```
GLdouble model[16], project[16];
GLint    view[4];

...

glGetDoublev (GL_MODELVIEW_MATRIX, model);
glGetDoublev (GL_PROJECTION_MATRIX, project);
glGetIntegerv(GL_VIEWPORT, view);
```



# TRASFORMAZIONI

- e` possibile conoscere il valore numerico che assumono i vertici in window coordinates:

```
void gluProject( objx, objy, objz,
                 model, project, view,
                 winx, winy, winz )
```

  - (`objx`, `objy`, `objz`) sono le coordinate del punto da trasformare
  - `model` e` la matrice modelview
  - `project` e` la matrice projection
  - `view` sono i parametri della trasformazione di viewport
  - il risultato viene copiato in (`winx`, `winy`, `winz`)

## ESERC.: TRASFORMAZ.

- disegnare un vertice in `( 0, 0, 0 )` e determinare le sue corrispondenti window coordinates
  - leggere i parametri relativi alla trasformazione di viewing corrente
  - utilizzare la funzione **`gluProject()`**
  - notare il valore di **`winz`**
- osservare come le win. coord. variano modificando le dimensioni della finestra
- determinare le win. coord. corrispondenti a vertici in posizioni arbitrarie
  - notare il valore di **`winz`**



## TRASFORMAZ. (cont.)

- esiste la trasformazione inversa di **`gluProject()`**:

```
void gluUnProject(  winx, winy, winz,
                    model, project, view,
                    objx, objy, objz )
```

  - (**`winx`**, **`winy`**, **`winz`**) sono le coordinate del punto da antitrasformare
  - **`model`** e` la matrice modelview
  - **`project`** e` la matrice projection
  - **`view`** sono i parametri della trasformazione di viewport
  - il risultato viene copiato in (**`objx`**, **`objy`**, **`objz`**)

# ESERC.: TRASF. (cont.)

- sapendo che:
  - la posizione del mouse puo` essere letta in qualsiasi momento con:

```
void glutPassiveMotionFunc(void(*f)(int,int));
```

  - `win_x = mouse_x` e

    `win_y = win_height - mouse_y`

  disegnare un oggetto (es. una sfera) che segua il movimento del mouse
  - leggere la posizione corrente del mouse
  - `win_z` arbitrario
  - utilizzare **`gluUnProject`**`()`
  - modificare `win_z` interattivamente

© F. DE ANGELIS