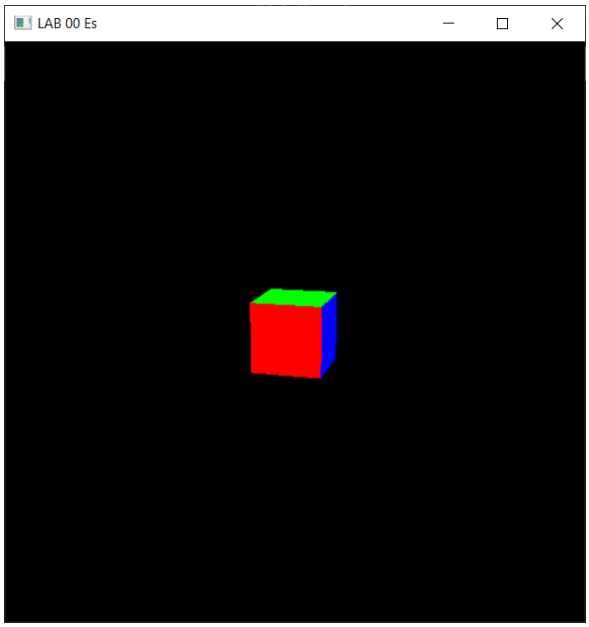
Fondamenti di Computer Graphics M (8 cfu)

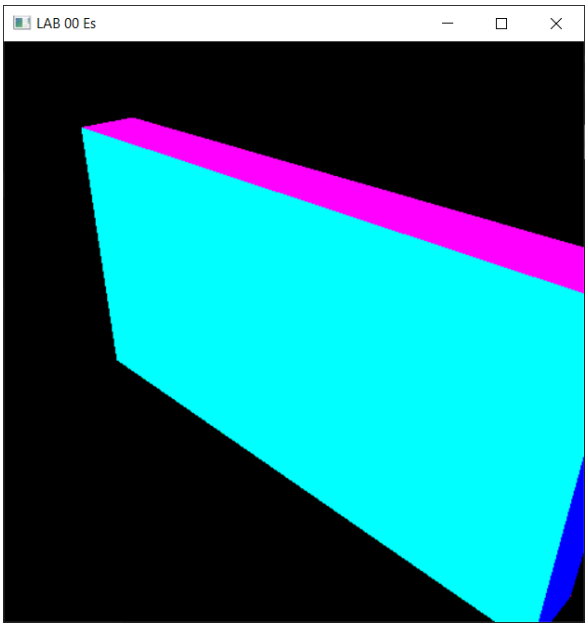
Lab 0 - Introduzione ad OpenGL

Michele Righi

Settembre 2023



(a) Cubo con transform di default



(b) Cubo ruotato e scalato

Figure 1: Demo

Traccia

Estendere il programma secondo le seguenti specifiche:

1. Disegnare un solo cubo.
2. Cambiare il colore del cubo da colori ai vertici a colori alle facce.
3. Permettere di scalare di un fattore di scala $sf\_act = 1.1$ lungo uno dei tre assi tramite mouse. In particolare implemenatare la callback *mymouse()* per permettere di selezionare tramite mouse button l'asse di scala mediante *GLUT_LEFT_BUTTON* per l'asse x, *GLUT_MIDDLE_BUTTON* per l'asse y, *GLUT_RIGHT_ BUTTON* per l'asse z.

1 Implementazione

1.1 Colori alle Facce

Per cambiare il colore dei cubi, da colori ai vertici a colori alle facce, ho modificato la funzione `polygon()`: ho aggiunto un parametro per il colore, in modo tale che si possa specificare in modo arbitrario per ciascun poligono:

```
void polygon(int a, int b, int c, int d, colorRGBA color)
{
        vPositions[Index] = positions[a]; vColors[Index] = color; Index++;
        vPositions[Index] = positions[b]; vColors[Index] = color; Index++;
        vPositions[Index] = positions[c]; vColors[Index] = color; Index++;
        vPositions[Index] = positions[a]; vColors[Index] = color; Index++;
        vPositions[Index] = positions[c]; vColors[Index] = color; Index++;
        vPositions[Index] = positions[d]; vColors[Index] = color; Index++;
}
```

In questo modo, ciascun poligono avrà lo stesso colore per tutta la superficie.

Per rendere il codice più leggibile, invece di un array per i colori, ho utilizzato delle costanti di tipo `colorRGBA`:

```
typedef glm::vec4 colorRGBA;

const colorRGBA BLACK = colorRGBA(1.0, 0.0, 0.0, 1.0);
const colorRGBA RED = colorRGBA(1.0, 0.0, 0.0, 1.0);
const colorRGBA YELLOW = colorRGBA(1.0, 1.0, 0.0, 1.0);
const colorRGBA GREEN = colorRGBA(0.0, 1.0, 0.0, 1.0);
const colorRGBA BLUE = colorRGBA(0.0, 0.0, 1.0, 1.0);
const colorRGBA MAGENTA = colorRGBA(1.0, 0.0, 1.0, 1.0);
const colorRGBA WHITE = colorRGBA(1.0, 1.0, 1.0, 1.0);
const colorRGBA CYAN = colorRGBA(0.0, 1.0, 1.0, 1.0);

void colorcube()
{
        polygon(1, 0, 3, 2, RED);
        polygon(2, 3, 7, 6, BLUE);
        polygon(3, 0, 4, 7, MAGENTA);
        polygon(6, 5, 1, 2, GREEN);
        polygon(4, 5, 6, 7, CYAN);
        polygon(5, 4, 0, 1, YELLOW);
}
```

1.2 Controlli

Per l'input ho utilizzato un file separato `input.h` che salva in una struttura dati lo stato corrente degli input (tastiera e mouse). Così facendo, si può controllare se un tasto è stato premuto direttamente dalla funzione di update, e ciò consente di aggiornare la scena - in base agli input - in

modo più fluido (prima avveniva a scatti). Inoltre, ho reso i vari controlli di input non mutuamente esclusivi, così da poterli combinare.

1.2.1 Spostamento Camera

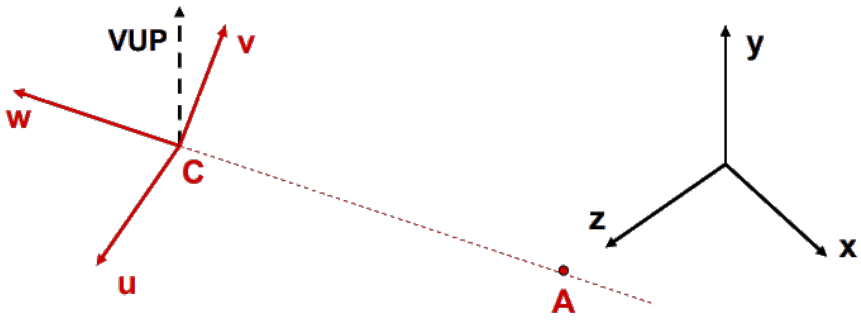


Figure 2: Camera Frame in WCS

Per lo spostamento della camera ho utilizzato i seguenti controlli da tastiera:

- `Barra Spaziatrice/Maiuscolo` per muovere la camera su/giù, sfruttando il vettore $\vec{v}$:

```
// Move camera upwards
if (input.keys[' ']) {
    cameraPos -= alfa * cameraUp;
}
// Move camera downwards
if (input.specialKeys[0160]) {
    cameraPos += alfa * cameraUp;
}
```

- `'a'/'d'` per muovere la camera a sinistra/destra, ottenendo la direzione (vettore $\vec{u}$) grazie al cross product tra vettore verso cui la camera è orientata $\overrightarrow{CA}$ e vettore $\vec{v}$:

```
// Move camera to the left
if (input.keys['a']) {
    glm::vec3 direction = normalize(cross(cameraFront, cameraUp));
    cameraPos -= alfa * direction;
}
// Move camera to the right
if (input.keys['d']) {
    glm::vec3 direction = normalize(cross(cameraFront, cameraUp));
    cameraPos += alfa * direction;
}
```

- `'w'/'s'` per avvicinare/allontanare la camera, sfruttando il vettore $\overrightarrow{CA}$:

```
// Move camera close (same direction)
if (input.keys['w']) {
    cameraPos += alfa * cameraFront;
}
// move camera away (same direction)
if (input.keys['s']) {
    cameraPos -= alfa * cameraFront;
}
```

- `'t'` per resettare la posizione della camera.

1.2.2 Rotazione Cubo

Per la rotazione del cubo ho utilizzato i seguenti controlli da tastiera:

- `Freccia su`/`Freccia giù` per ruotare il cubo lungo l'asse X in senso orario/antiorario:

```
// Rotate along X axis (clockwise)
if (input.specialKeys[GLUT_KEY_UP]) {
    rotateX -= 15;
    if (rotateX < 0)
        rotateX += 360;
}
// Rotate along X axis (counter-clockwise)
if (input.specialKeys[GLUT_KEY_DOWN]) {
    rotateX += 15;
    if (rotateX >= 360)
        rotateX -= 360;
}
```

- `Freccia sinistra`/`Freccia destra` per ruotare il cubo lungo l'asse Y in senso orario/antiorario:

```
// Rotate along Y axis (clockwise)
if (input.specialKeys[GLUT_KEY_LEFT]) {
    rotateY -= 15;
    if (rotateY < 0)
        rotateY += 360;
}
// Rotate along Y axis (counter-clockwise)
if (input.specialKeys[GLUT_KEY_RIGHT]) {
    rotateY += 15;
    if (rotateY >= 360)
        rotateY -= 360;
}
```

- `Pagina su/Pagina giù` per ruotare il cubo lungo l'asse Z in senso orario/antiorario:

```
// Rotate along Z axis (clockwise)
if (input.specialKeys[GLUT_KEY_PAGE_UP]) {
    rotateZ -= 15;
    if (rotateZ < 0)
        rotateZ += 360;
}
// Rotate along Z axis (counter-clockwise)
if (input.specialKeys[GLUT_KEY_PAGE_DOWN]) {
    rotateZ += 15;
    if (rotateZ >= 360)
        rotateZ -= 360;
}
```

- `'r'` per resettare la rotazione del cubo.

1.2.3 Scaling Cubo

Per lo scaling del cubo ho utilizzato i seguenti controlli da mouse:

- `Click sinistro` per aumentare lo scaling del cubo lungo l'asse X:

```
// Scale along X axis
if (input.mouse.keys[GLUT_LEFT_BUTTON]) {
    scaleX = MIN(10.0, scaleX * 1.1);
}
else scaleX = MAX(1.0, scaleX * 0.9);
```

- `Click centrale` (rotella) per aumentare lo scaling del cubo lungo l'asse Y:

```
// Scale along Y axis
if (input.mouse.keys[GLUT_MIDDLE_BUTTON]) {
    scaleY = MIN(10.0, scaleY * 1.1);
}
else scaleY = MAX(1.0, scaleY * 0.9);
```

- `Click destro` per aumentare lo scaling del cubo lungo l'asse Z:

```
// Scale along Z axis
if (input.mouse.keys[GLUT_RIGHT_BUTTON]) {
    scaleZ = MIN(10.0, scaleZ * 1.1);
}
else scaleZ = MAX(1.0, scaleZ * 0.9);
```