

# Department of CSE

## SSN College of Engineering

Vishakan Subramanian - 18 5001 196 - Semester VII

14 September 2021

---

### UCS 1712 - Graphics And Multimedia Lab

---

#### Exercise 7: 3D Transformations in C++ using OpenGL

##### **Aim:**

To apply the following 3D transformations on objects and to render the final output along with the original object.

- Translation
- Rotation
- Scaling
- Reflection
- Shearing

Display the original and the transformed object.

## Code: 3D Transformations:

```
1  /*
2  3D Transformations - Translation, Rotation, Scaling, Reflection and
   Shearing
3  */
4
5  #include <stdio.h>
6  #include <math.h>
7  #include <GL/glut.h>
8  #include <iostream>      //for cin, cout
9  #include <cstring>      //for memcpy
10
11 using namespace std;
12
13 enum Axes {xAxis = 0, yAxis = 1, zAxis = 2};
14 enum Planes {xyPlane = 0, yzPlane = 1, zxPlane = 2};
15
16 const int WINDOW_HEIGHT = 800;
17 const int WINDOW_WIDTH = 800;
18 const int FPS = 60;
19
20 class Face{      //wrapper class for a face of a 3D object
21     private:
22         GLfloat r = 1, g = 1, b = 1, a = 1; //black
23         GLfloat vertices[4][4];      // V * (x, y, z, h)
24         int vertexCount = 4;         // V
25
26     public:
27         Face(){
28             r = g = b = a = 1; //black
29             vertexCount = 4;
30         }
31
32         Face(GLfloat R, GLfloat G, GLfloat B, GLfloat A){
33             //Set colors
34             Face();
35             r = R;
36             g = G;
37             b = B;
38             a = A;
39         }
40
41         void setColor(GLfloat R, GLfloat G, GLfloat B, GLfloat A){
42             //Set colors
43             r = R;
44             g = G;
45             b = B;
46             a = A;
```

```

47     }
48
49     void setIthVertex(int i, GLfloat x, GLfloat y, GLfloat z, GLfloat
h = 1){
50         //Set the ith vertex coordinates
51         vertices[i][0] = x;
52         vertices[i][1] = y;
53         vertices[i][2] = z;
54         vertices[i][3] = h;
55     }
56
57     void drawFace(){
58         glColor4f(r, g, b, a);
59
60         glBegin(GL_POLYGON);
61
62         for(int i = 0; i < vertexCount; i++){
63             glVertex3f(vertices[i][0], vertices[i][1], vertices[i][2])
;
64         }
65
66         glEnd();
67     }
68
69     Face transform(double transformationMatrix[4][4]){
70         Face fDash(r, g, b, a - 0.5);
71         double values[4];
72
73         // [V x 4] x [4 x 1] = [V x 1] matrix
74         for(int i = 0; i < vertexCount; i++){
75             for(int j = 0; j < 4; j++){
76                 values[j] = transformationMatrix[j][0] * vertices[i
] [0] +
77                                     transformationMatrix[j][1] * vertices[i
] [1] +
78                                     transformationMatrix[j][2] * vertices[i
] [2] +
79                                     transformationMatrix[j][3] * vertices[i
] [3];
80             }
81
82             fDash.setIthVertex(i, values[0], values[1], values[2],
values[3]);
83             //cout << "\nVertex" << values[0] << " " << values[1] << "
" << values[2] << " " << values[3];
84
85         }
86
87         return fDash;
88     }
89 };

```

```

90
91 class Object3D{           //wrapper class for a 3D object with multiple faces
92     private:
93         Face *faces;
94         int numFaces;
95
96     public:
97         Object3D(){
98             numFaces = 0;
99         }
100
101         Object3D(int noFaces){
102             numFaces = noFaces;
103             faces = new Face[numFaces];
104         }
105
106         void setIthFace(int i, Face face){
107             faces[i] = face;
108         }
109
110         void drawObject3D(){
111             for(int i = 0; i < numFaces; i++){
112                 faces[i].drawFace();
113             }
114         }
115
116         Object3D translateObject3D(GLfloat tx, GLfloat ty, GLfloat tz){
117             //To translate the 3D object wrt. a translation vector
118
119             double translationMatrix[4][4] = { {1, 0, 0, tx},
120                                                 {0, 1, 0, ty},
121                                                 {0, 0, 1, tz},
122                                                 {0, 0, 0, 1}};
123
124             Object3D transformedObject(numFaces);
125
126             for(int i = 0; i < numFaces; i++){
127                 Face fDash = faces[i].transform(translationMatrix);
128                 transformedObject.setIthFace(i, fDash);
129             }
130
131             return transformedObject;
132         }
133
134         Object3D scaleObject3D(GLfloat sx, GLfloat sy, GLfloat sz){
135             //To scale the 3D object wrt. the given scaling factors
136
137             double scalingMatrix[4][4] = { {sx, 0, 0, 0},
138                                             {0, sy, 0, 0},
139                                             {0, 0, sz, 0},
140                                             {0, 0, 0, 1}};

```

```

141
142     Object3D transformedObject(numFaces);
143
144     for(int i = 0; i < numFaces; i++){
145         Face fDash = faces[i].transform(scalingMatrix);
146         transformedObject.setIthFace(i, fDash);
147     }
148
149     return transformedObject;
150 }
151
152 Object3D rotateObject3D(int axis, double rotationAngle){
153     //To rotate the 3D object about an axis and an angle
154
155     double rotationAngleInRadians = rotationAngle * 3.14159 / 180;
156     double cosAngle = cos(rotationAngleInRadians);
157     double sinAngle = sin(rotationAngleInRadians);
158     double rotationMatrix[4][4];
159
160     switch(axis){
161         case xAxis:{
162             double temp[4][4] = {    {1, 0, 0, 0},
163                                     {0, cosAngle, -sinAngle, 0},
164                                     {0, sinAngle, cosAngle, 0},
165                                     {0, 0, 0, 1}};
166
167             memcpy(rotationMatrix, temp, sizeof(temp));
168             break;
169         }
170
171         case yAxis:{
172             double temp[4][4] = {    {cosAngle, 0, sinAngle, 0},
173                                     {0, 1, 0, 0},
174                                     {-sinAngle, 0, cosAngle, 0},
175                                     {0, 0, 0, 1}};
176
177             memcpy(rotationMatrix, temp, sizeof(temp));
178             break;
179         }
180
181         case zAxis:{
182             double temp[4][4] = {    {cosAngle, -sinAngle, 0, 0},
183                                     {sinAngle, cosAngle, 0, 0},
184                                     {0, 0, 1, 0},
185                                     {0, 0, 0, 1}};
186
187             memcpy(rotationMatrix, temp, sizeof(temp));
188             break;
189         }
190     }
191 }

```

```

192     Object3D transformedObject(numFaces);
193
194     for(int i = 0; i < numFaces; i++){
195         Face fDash = faces[i].transform(rotationMatrix);
196         transformedObject.setIthFace(i, fDash);
197     }
198
199     return transformedObject;
200 }
201
202 Object3D shearObject3D(int axis, double shx = 0, double shy = 0,
203 double shz = 0){
204     //To shear the 3D object wrt. an axis and the given shear
205     parameters
206
207     double shearingMatrix[4][4];
208
209     switch(axis){
210         case xAxis:{
211             double temp[4][4] = {    {1, 0, 0, 0},
212                                     {shy, 1, 0, 0},
213                                     {shz, 0, 1, 0},
214                                     {0, 0, 0, 1}};
215
216             memcpy(shearingMatrix, temp, sizeof(temp));
217             break;
218         }
219         case yAxis:{
220             double temp[4][4] = {    {1, shx, 0, 0},
221                                     {0, 1, 0, 0},
222                                     {0, shz, 1, 0},
223                                     {0, 0, 0, 1}};
224
225             memcpy(shearingMatrix, temp, sizeof(temp));
226             break;
227         }
228         case zAxis:{
229             double temp[4][4] = {    {1, 0, shx, 0},
230                                     {0, 1, shy, 0},
231                                     {0, 0, 1, 0},
232                                     {0, 0, 0, 1}};
233
234             memcpy(shearingMatrix, temp, sizeof(temp));
235             break;
236         }
237     }
238
239     Object3D transformedObject(numFaces);
240

```

```

241     for(int i = 0; i < numFaces; i++){
242         Face fDash = faces[i].transform(shearingMatrix);
243         transformedObject.setIthFace(i, fDash);
244     }
245
246     return transformedObject;
247 }
248
249 Object3D reflectObject3D(int plane){
250     //To reflect the 3D Object about a given plane
251
252     double reflectionMatrix[4][4];
253
254     switch(plane){
255         case xyPlane:{
256             double temp[4][4] = {    {1, 0, 0, 0},
257                                     {0, 1, 0, 0},
258                                     {0, 0, -1, 0},
259                                     {0, 0, 0, 1}};
260
261             memcpy(reflectionMatrix, temp, sizeof(temp));
262             break;
263         }
264
265         case yzPlane:{
266             double temp[4][4] = {    {-1, 0, 0, 0},
267                                     {0, 1, 0, 0},
268                                     {0, 0, 1, 0},
269                                     {0, 0, 0, 1}};
270
271             memcpy(reflectionMatrix, temp, sizeof(temp));
272             break;
273         }
274
275         case zxPlane:{
276             double temp[4][4] = {    {1, 0, 0, 0},
277                                     {0, -1, 0, 0},
278                                     {0, 0, 1, 0},
279                                     {0, 0, 0, 1}};
280
281             memcpy(reflectionMatrix, temp, sizeof(temp));
282             break;
283         }
284     }
285
286     Object3D transformedObject(numFaces);
287
288     for(int i = 0; i < numFaces; i++){
289         Face fDash = faces[i].transform(reflectionMatrix);
290         transformedObject.setIthFace(i, fDash);
291     }

```

```

292
293         return transformedObject;
294     }
295 };
296
297 void dummyFunction();
298 void mainLoop(int val);
299 void initializeDisplay();
300 void initializeBaseCube();
301 void plotBase3DObject();
302 void plotTransformation();
303
304 Object3D cube;
305
306 int main(int argc, char **argv){
307     glutInit(&argc, argv);
308     glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
309     glutInitWindowSize(WINDOW_WIDTH, WINDOW_HEIGHT);
310     glutCreateWindow("3D Transformations - Examples");
311
312     initializeBaseCube();
313     initializeDisplay();
314
315     glutDisplayFunc(dummyFunction);
316     glutTimerFunc(1000/FPS, mainLoop, 0);
317     glutMainLoop();
318
319     return 1;
320 }
321
322 void mainLoop(int val){
323     //Render the display using the timer function
324     plotTransformation();
325 }
326
327 void dummyFunction(){
328     //Placeholder function
329 }
330
331 void initializeDisplay(){
332     glClearColor(1, 1, 1, 1);
333     glOrtho(-800, 800, -800, 800, -800, 800);    //Orthographic projection
334     glEnable(GL_DEPTH_TEST);    //Enable depth
335
336     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
337
338     //Rotate the entire display so that different sides of the 3D object
    can be seen
339     glRotatef(50, 1, 0, 0);
340     glRotatef(50, 0, 1, 0);
341     glRotatef(50, 0, 0, 1);

```



```

342
343     glEnable(GL_BLEND);           //enable blending (translucent colors)
344     glDepthMask(GL_FALSE);
345     glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA); //set the blend
function for translucency
346 }
347
348 void initializeBaseCube(){
349     //Set the coordinates for the base cube
350
351     cube = Object3D(6);
352
353     Face front, back, left, right, bottom, top;
354
355     front = Face(1, 0, 0, 0.75);    //Red
356     front.setIthVertex(0, -100, 100, 100);
357     front.setIthVertex(1, 100, 100, 100);
358     front.setIthVertex(2, 100, -100, 100);
359     front.setIthVertex(3, -100, -100, 100);
360
361     back = Face(0, 1, 0, 0.75);    //Green
362     back.setIthVertex(0, -100, 100, -100);
363     back.setIthVertex(1, 100, 100, -100);
364     back.setIthVertex(2, 100, -100, -100);
365     back.setIthVertex(3, -100, -100, -100);
366
367     left = Face(0, 0, 1, 0.75);    //Blue
368     left.setIthVertex(0, -100, 100, -100);
369     left.setIthVertex(1, -100, 100, 100);
370     left.setIthVertex(2, -100, -100, 100);
371     left.setIthVertex(3, -100, -100, -100);
372
373     right = Face(1, 1, 0, 0.75);   //Yellow
374     right.setIthVertex(0, 100, 100, -100);
375     right.setIthVertex(1, 100, 100, 100);
376     right.setIthVertex(2, 100, -100, 100);
377     right.setIthVertex(3, 100, -100, -100);
378
379     bottom = Face(0, 1, 1, 0.75);  //Cyan
380     bottom.setIthVertex(0, -100, -100, -100);
381     bottom.setIthVertex(1, 100, -100, -100);
382     bottom.setIthVertex(2, 100, -100, 100);
383     bottom.setIthVertex(3, -100, -100, 100);
384
385     top = Face(1, 0, 1, 0.75);     //Magenta
386     top.setIthVertex(0, -100, 100, -100);
387     top.setIthVertex(1, 100, 100, -100);
388     top.setIthVertex(2, 100, 100, 100);
389     top.setIthVertex(3, -100, 100, 100);
390
391     cube.setIthFace(0, front);

```

```

392     cube.setIthFace(1, back);
393     cube.setIthFace(2, left);
394     cube.setIthFace(3, right);
395     cube.setIthFace(4, bottom);
396     cube.setIthFace(5, top);
397 }
398
399 void plotBase3DObject(){
400     //Plot the base 3D object without any transformations
401
402     glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);    //Clear window
403     cube.drawObject3D();
404     glFlush();
405 }
406
407 void plotTransformation(){
408     //Plot the transformation
409
410     plotBase3DObject();
411     glFlush();
412
413     int transform = 0;
414     Object3D cubeDash;
415
416     while(true){
417         //Await user input
418
419         cout << "\nChoose Transformation: " << endl;
420         cout << "\t1 for Translation" << endl;
421         cout << "\t2 for Rotation" << endl;
422         cout << "\t3 for Scaling" << endl;
423         cout << "\t4 for Reflection" << endl;
424         cout << "\t5 for Shearing" << endl;
425         cout << "\t0 to Exit" << endl;
426         cout << "\tYour Option -> ";
427         cin >> transform;
428
429         switch(transform){
430             case 0:{
431                 exit(0);
432             }
433
434             case 1:{
435                 float tx, ty, tz;
436                 cout << endl << "-----TRANSLATION-----" << endl;
437                 cout << "\nEnter Translation Vector Coordinates: " << endl
;
438                 cout << "\nEnter X: "; cin >> tx;
439                 cout << "\nEnter Y: "; cin >> ty;
440                 cout << "\nEnter Z: "; cin >> tz;
441

```

```

442         cubeDash = cube.translateObject3D(tx, ty, tz);
443         cout << endl << "-----TRANSLATION DONE-----" << endl;
444
445         break;
446     }
447
448     case 2:{
449         double angle; int axis;
450         cout << endl << "-----ROTATION-----" << endl;
451         cout << "\nEnter Rotation Axis: " << endl;
452         cout << "\t0 for X-Axis" << endl;
453         cout << "\t1 for Y-Axis" << endl;
454         cout << "\t2 for Z-Axis" << endl;
455         cout << "\tYour Option -> ";
456         cin >> axis;
457
458         cout << "\nEnter Rotation Angle: "; cin >> angle;
459
460         cubeDash = cube.rotateObject3D(axis, angle);
461         cout << endl << "-----ROTATION DONE-----" << endl;
462         break;
463     }
464
465     case 3:{
466         float sx, sy, sz;
467
468         cout << endl << "-----SCALING-----" << endl;
469         cout << "\nEnter Scale Factors: " << endl;
470         cout << "\nEnter X Factor: "; cin >> sx;
471         cout << "\nEnter Y Factor: "; cin >> sy;
472         cout << "\nEnter Z Factor: "; cin >> sz;
473
474         cubeDash = cube.scaleObject3D(sx, sy, sz);
475         cout << endl << "-----SCALING DONE-----" << endl;
476
477         break;
478     }
479
480     case 4:{
481         int plane;
482         cout << endl << "-----REFLECTION-----" << endl;
483         cout << "\nEnter Reflection Plane: " << endl;
484         cout << "\t0 for XY-Plane" << endl;
485         cout << "\t1 for YZ-Plane" << endl;
486         cout << "\t2 for ZX-Plane" << endl;
487         cout << "\tYour Option -> ";
488         cin >> plane;
489
490         cubeDash = cube.reflectObject3D(plane);
491         cout << endl << "-----REFLECTION DONE-----" << endl;
492         break;

```

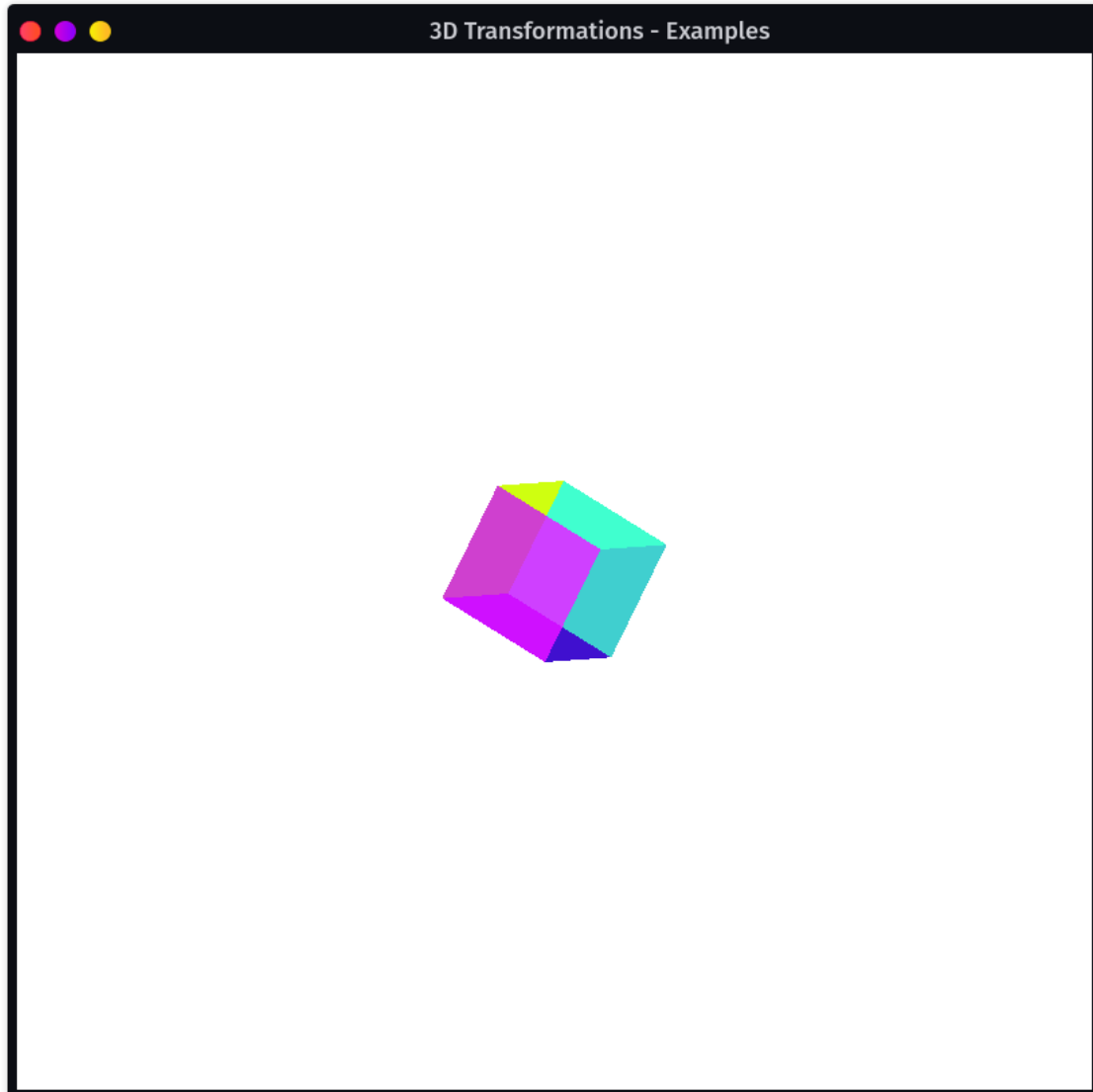
```

493     }
494
495     case 5:{
496         double shx = 0, shy = 0, shz = 0; int axis;
497
498         cout << endl << "-----SHEARING-----" << endl;
499         cout << "\nEnter Shear Axis: " << endl;
500         cout << "\t0 for X-Axis" << endl;
501         cout << "\t1 for Y-Axis" << endl;
502         cout << "\t2 for Z-Axis" << endl;
503         cout << "\tYour Option -> ";
504         cin >> axis;
505
506         cout << "\nEnter Shear Factors: " << endl;
507
508         switch(axis){
509             case xAxis:{
510                 cout << "\n Enter Y Factor: "; cin >> shy;
511                 cout << "\n Enter Z Factor: "; cin >> shz;
512                 break;
513             }
514
515             case yAxis:{
516                 cout << "\n Enter X Factor: "; cin >> shx;
517                 cout << "\n Enter Z Factor: "; cin >> shz;
518                 break;
519             }
520
521             case zAxis:{
522                 cout << "\n Enter X Factor: "; cin >> shx;
523                 cout << "\n Enter Y Factor: "; cin >> shy;
524                 break;
525             }
526         }
527
528         cubeDash = cube.shearObject3D(axis, shx, shy, shz);
529         cout << endl << "-----SHEARING DONE-----" << endl;
530         break;
531     }
532 }
533
534 plotBase3DObject();
535 cubeDash.drawObject3D();
536 glFlush();
537 }
538 }

```

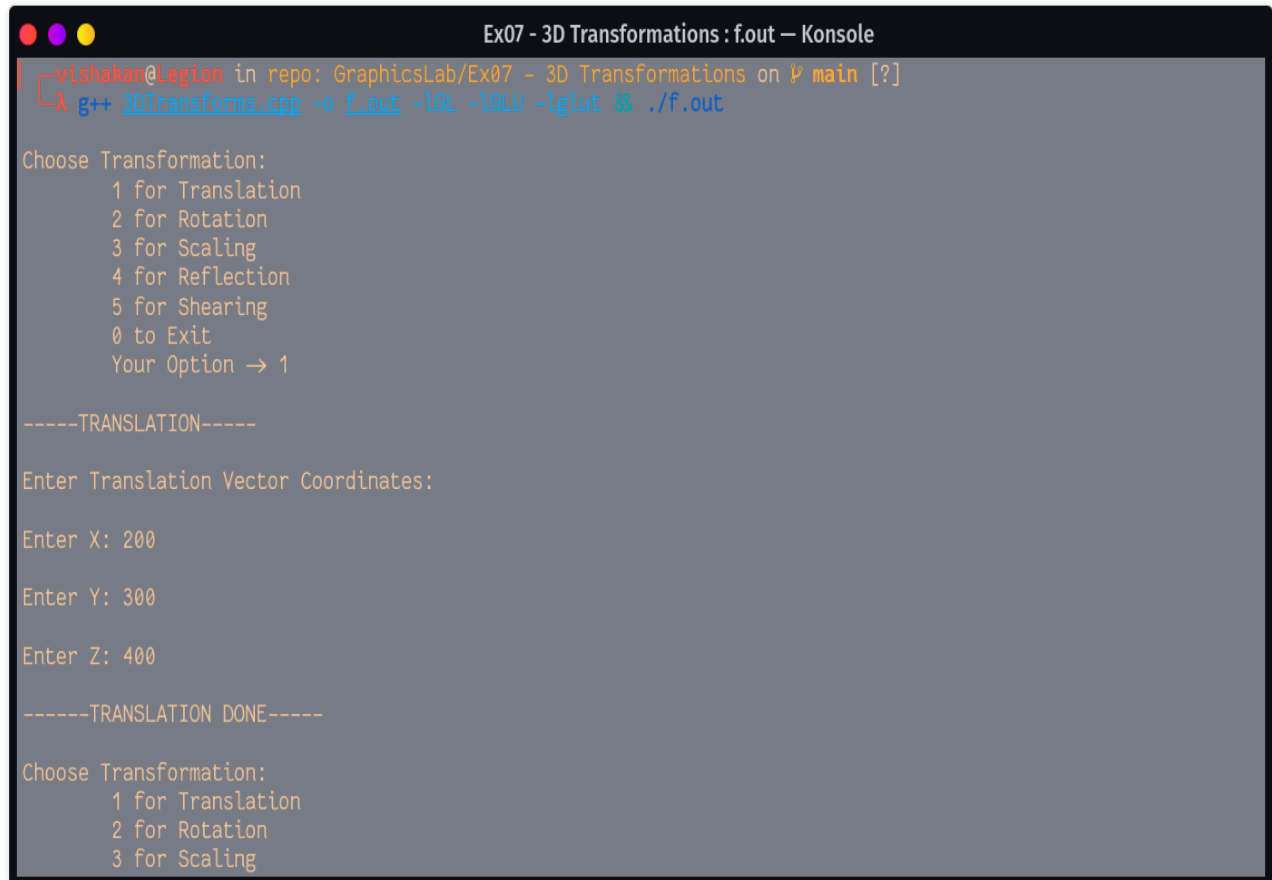
## Output: Base 3D Object

Figure 1: Base 3D Object.



## Output: Console - Translation, (200, 300, 400)

Figure 2: Output: Console - Translation, (200, 300, 400).



```
Ex07 - 3D Transformations : f.out - Konsole
vishakan@Legion in repo: GraphicsLab/Ex07 - 3D Transformations on P main [?]
λ g++ 3DTransforms.cpp -o f.out -lGL -lGLU -lglut && ./f.out

Choose Transformation:
    1 for Translation
    2 for Rotation
    3 for Scaling
    4 for Reflection
    5 for Shearing
    0 to Exit
Your Option → 1

-----TRANSLATION-----

Enter Translation Vector Coordinates:

Enter X: 200

Enter Y: 300

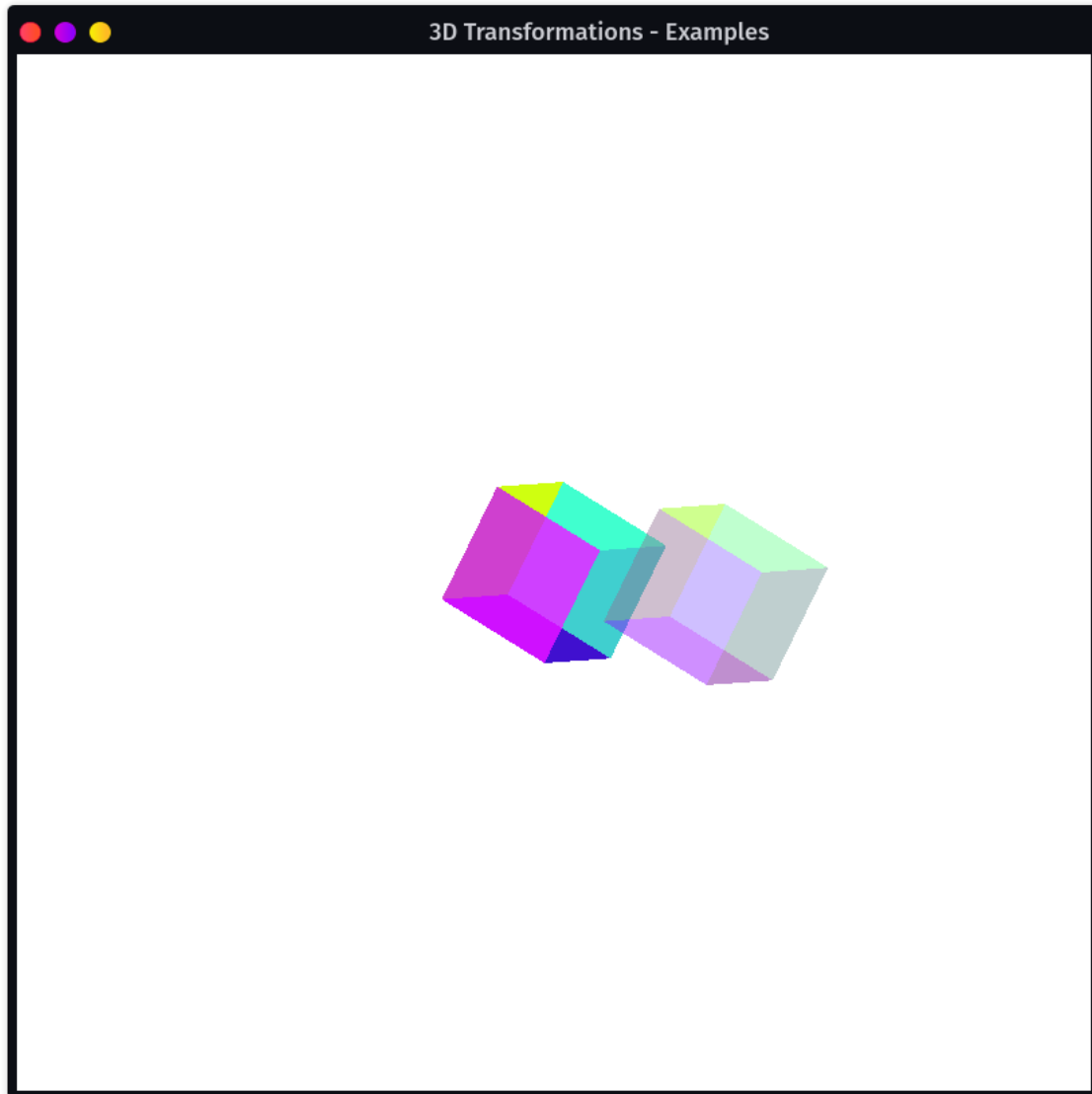
Enter Z: 400

-----TRANSLATION DONE-----

Choose Transformation:
    1 for Translation
    2 for Rotation
    3 for Scaling
```

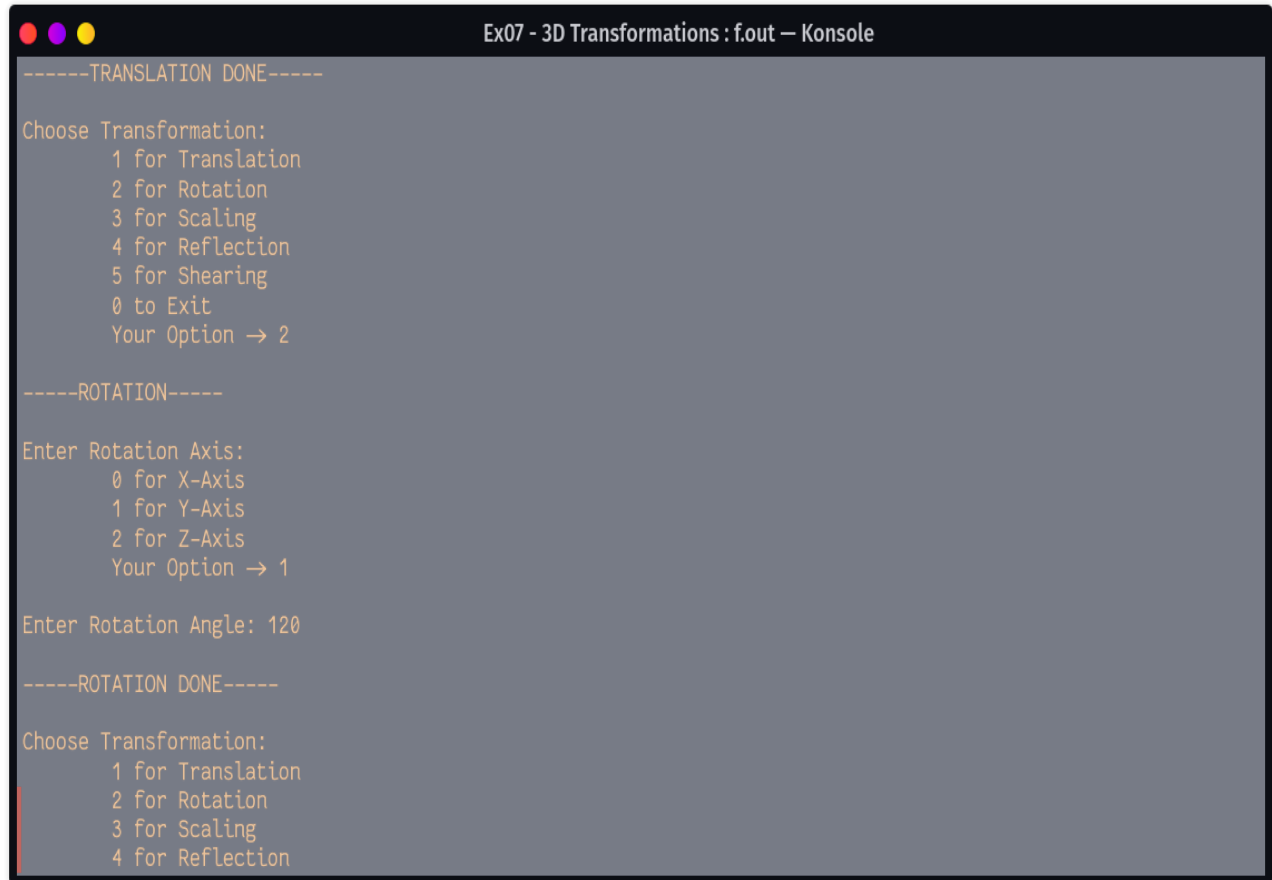
**Output: Translated 3D Object, (200, 300, 400)**

Figure 3: Translated 3D Object, (200, 300, 400).



## Output: Console - Rotation, Y-Axis, 120°

Figure 4: Output: Console - Rotation, Y-Axis, 120°.

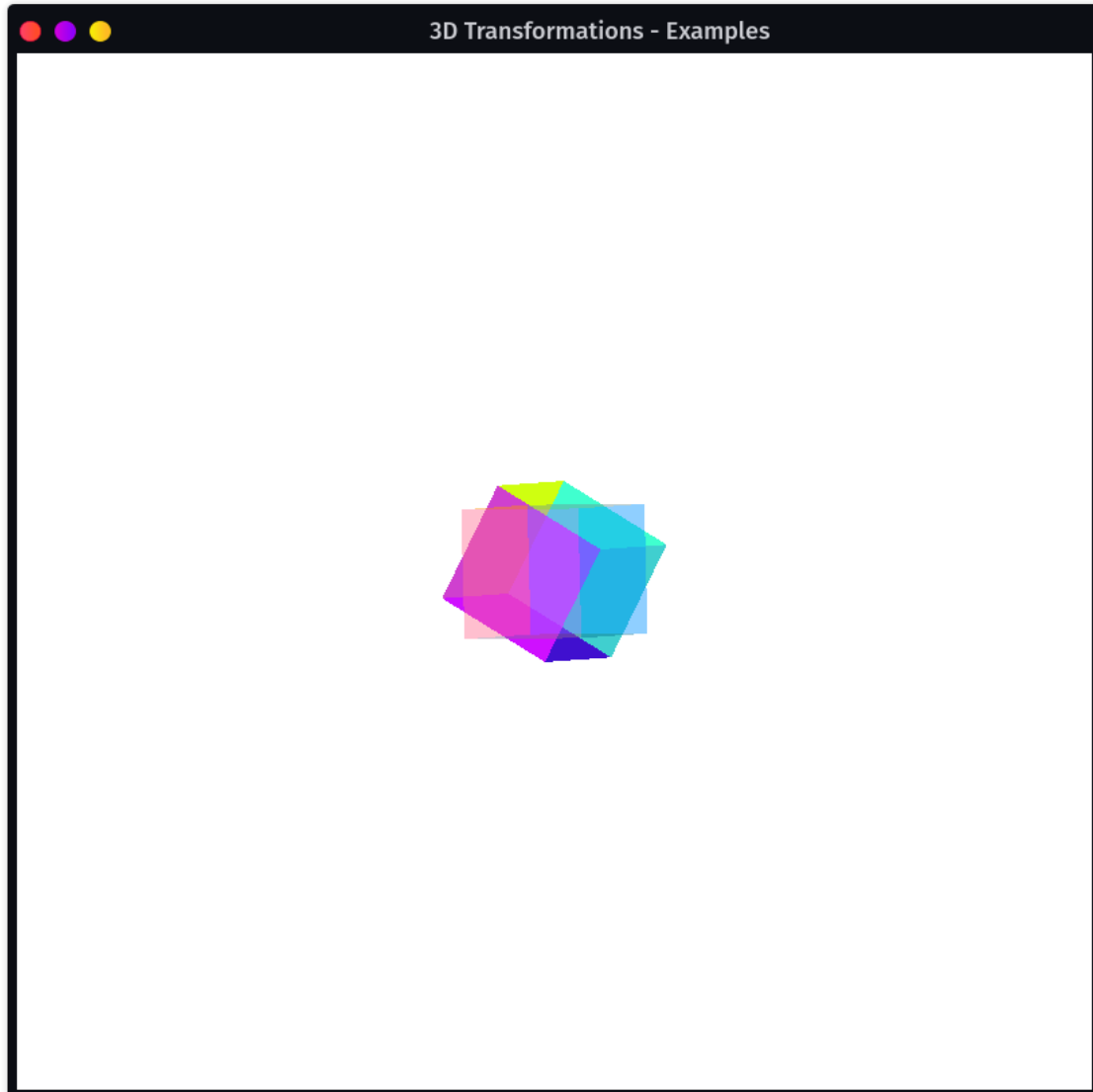


```
-----TRANSLATION DONE-----  
Choose Transformation:  
    1 for Translation  
    2 for Rotation  
    3 for Scaling  
    4 for Reflection  
    5 for Shearing  
    0 to Exit  
Your Option → 2  
  
-----ROTATION-----  
Enter Rotation Axis:  
    0 for X-Axis  
    1 for Y-Axis  
    2 for Z-Axis  
Your Option → 1  
  
Enter Rotation Angle: 120  
  
-----ROTATION DONE-----  
Choose Transformation:  
    1 for Translation  
    2 for Rotation  
    3 for Scaling  
    4 for Reflection
```



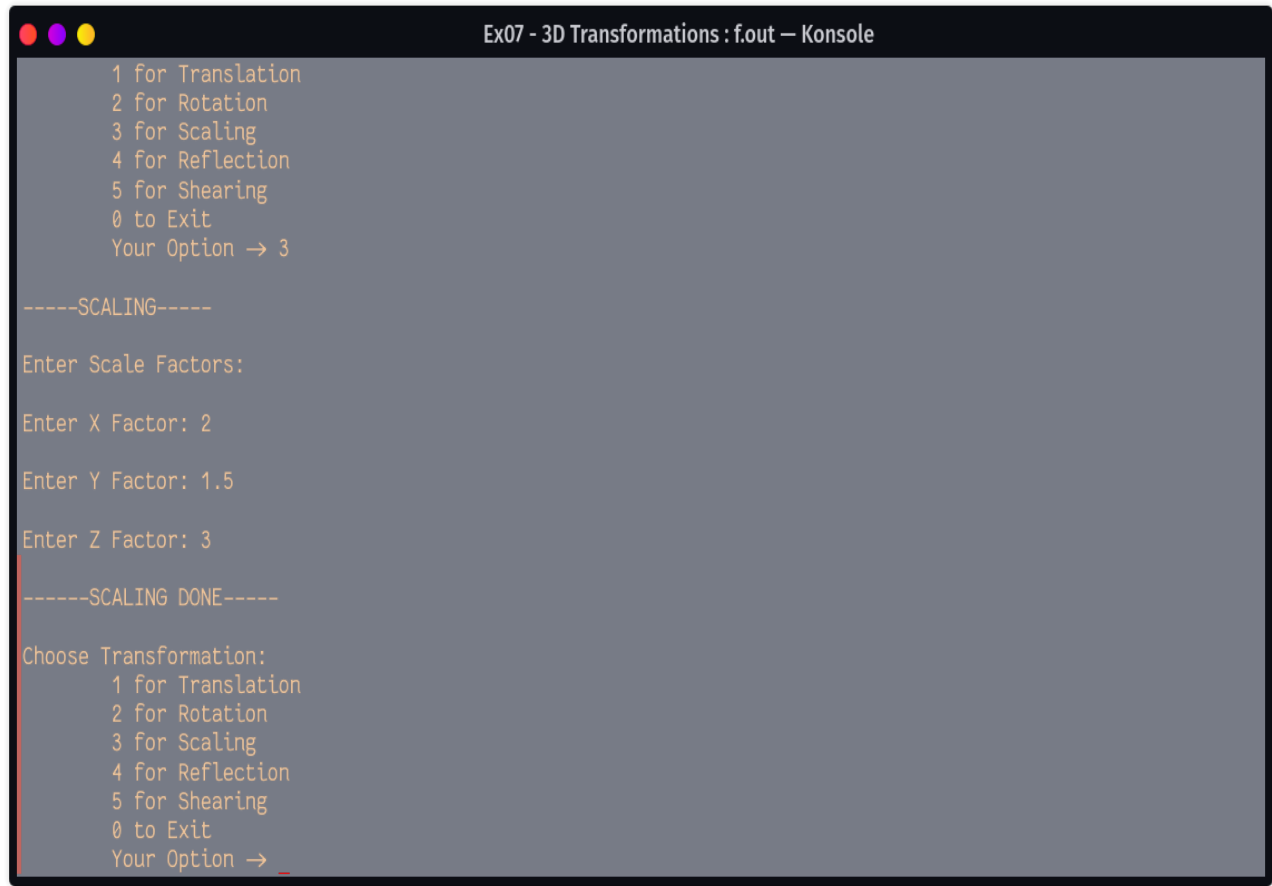
**Output: Rotated 3D Object, Y-Axis,  $120^\circ$**

Figure 5: Rotated 3D Object, Y-Axis,  $120^\circ$ .



## Output: Console - Scaling, (2, 1.5, 3)

Figure 6: Output: Console - Scaling, (2, 1.5, 3).



```
1 for Translation
2 for Rotation
3 for Scaling
4 for Reflection
5 for Shearing
0 to Exit
Your Option → 3

-----SCALING-----

Enter Scale Factors:

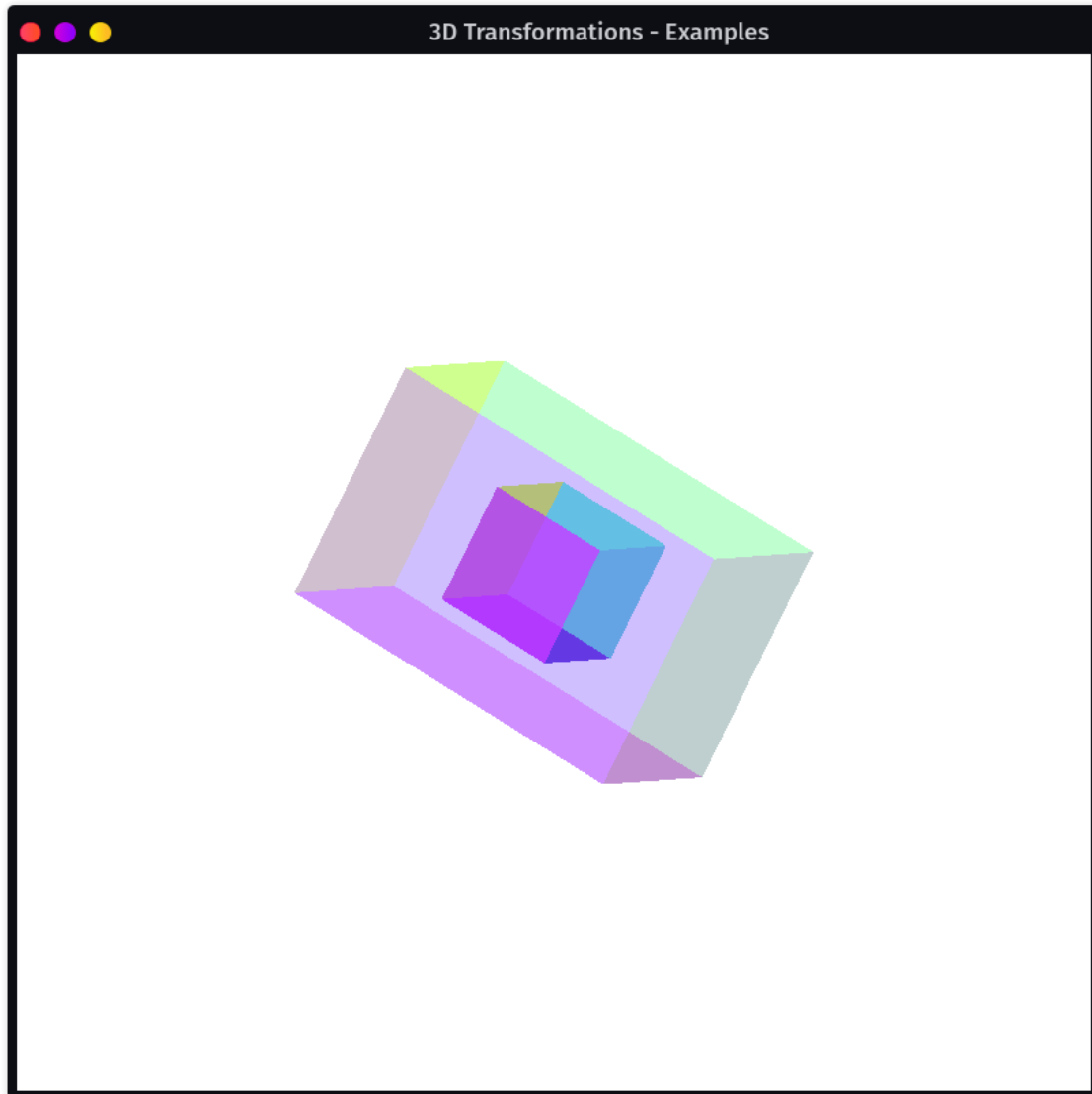
Enter X Factor: 2
Enter Y Factor: 1.5
Enter Z Factor: 3

-----SCALING DONE-----

Choose Transformation:
1 for Translation
2 for Rotation
3 for Scaling
4 for Reflection
5 for Shearing
0 to Exit
Your Option → _
```

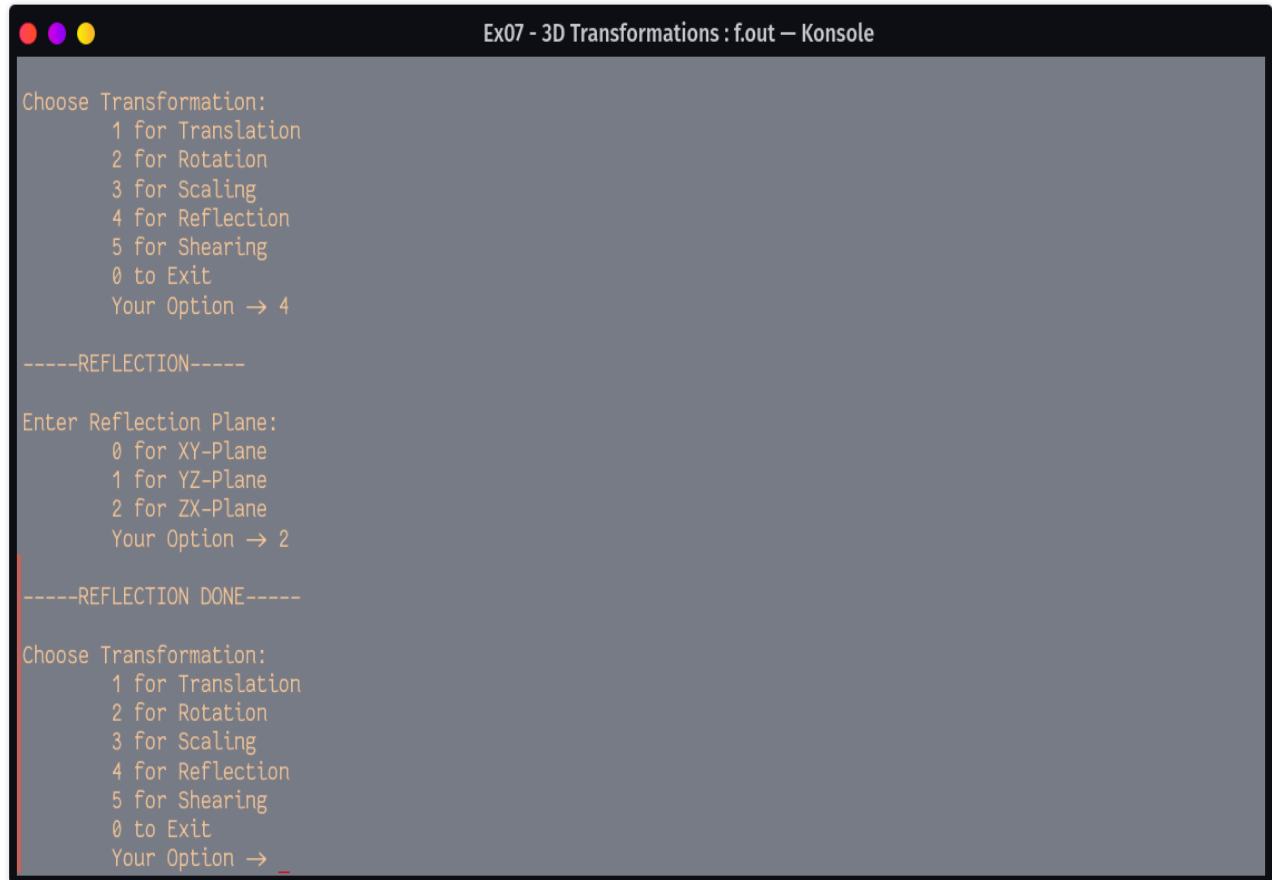
**Output: Scaled 3D Object, (2, 1.5, 3)**

Figure 7: Scaled 3D Object, (2, 1.5, 3).



## Output: Console - Reflection, ZX Plane

Figure 8: Output: Console - Reflection, ZX Plane.



```
Ex07 - 3D Transformations : f.out - Konsole

Choose Transformation:
  1 for Translation
  2 for Rotation
  3 for Scaling
  4 for Reflection
  5 for Shearing
  0 to Exit
Your Option → 4

----REFLECTION----

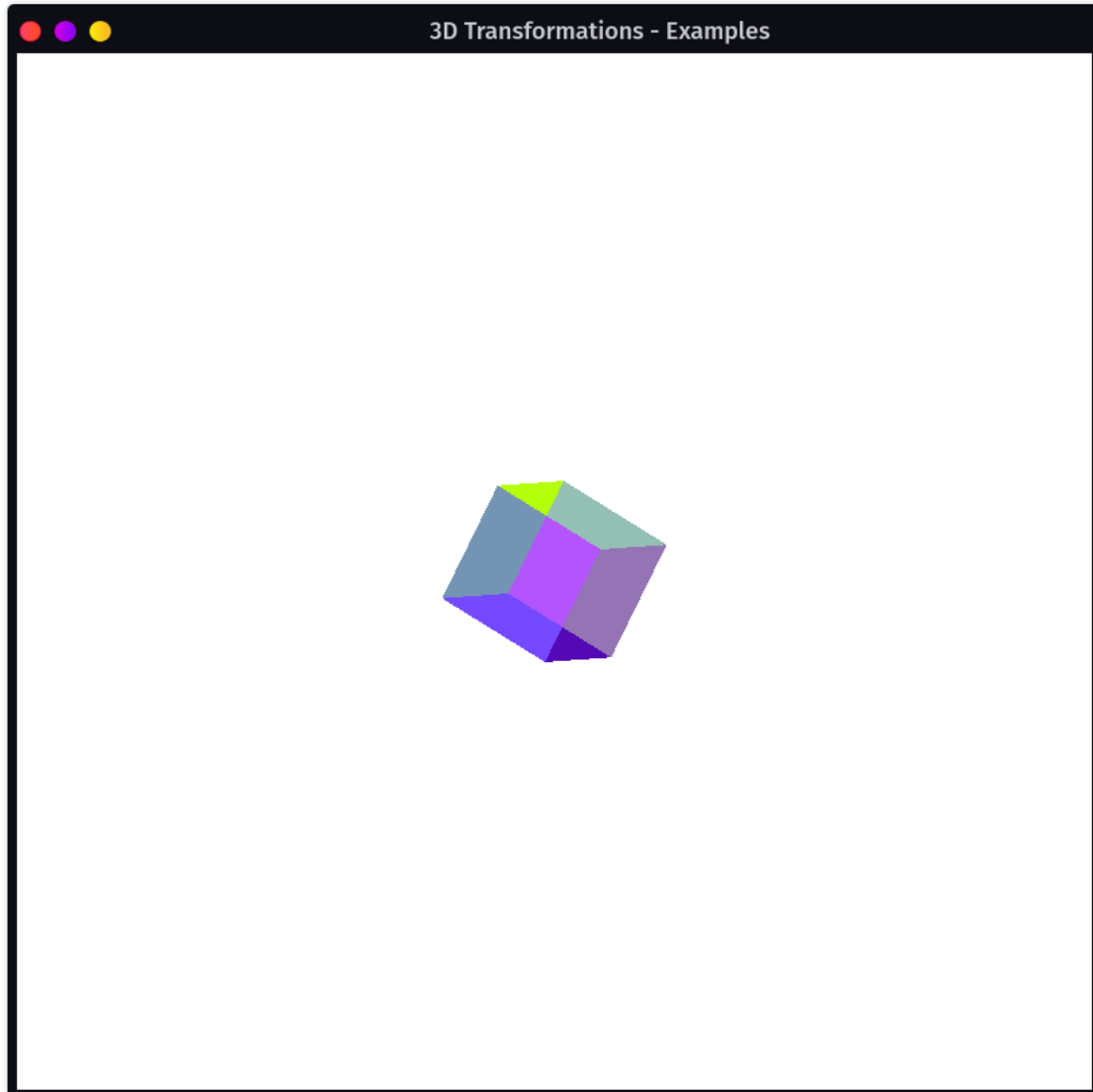
Enter Reflection Plane:
  0 for XY-Plane
  1 for YZ-Plane
  2 for ZX-Plane
Your Option → 2

----REFLECTION DONE----

Choose Transformation:
  1 for Translation
  2 for Rotation
  3 for Scaling
  4 for Reflection
  5 for Shearing
  0 to Exit
Your Option → _
```

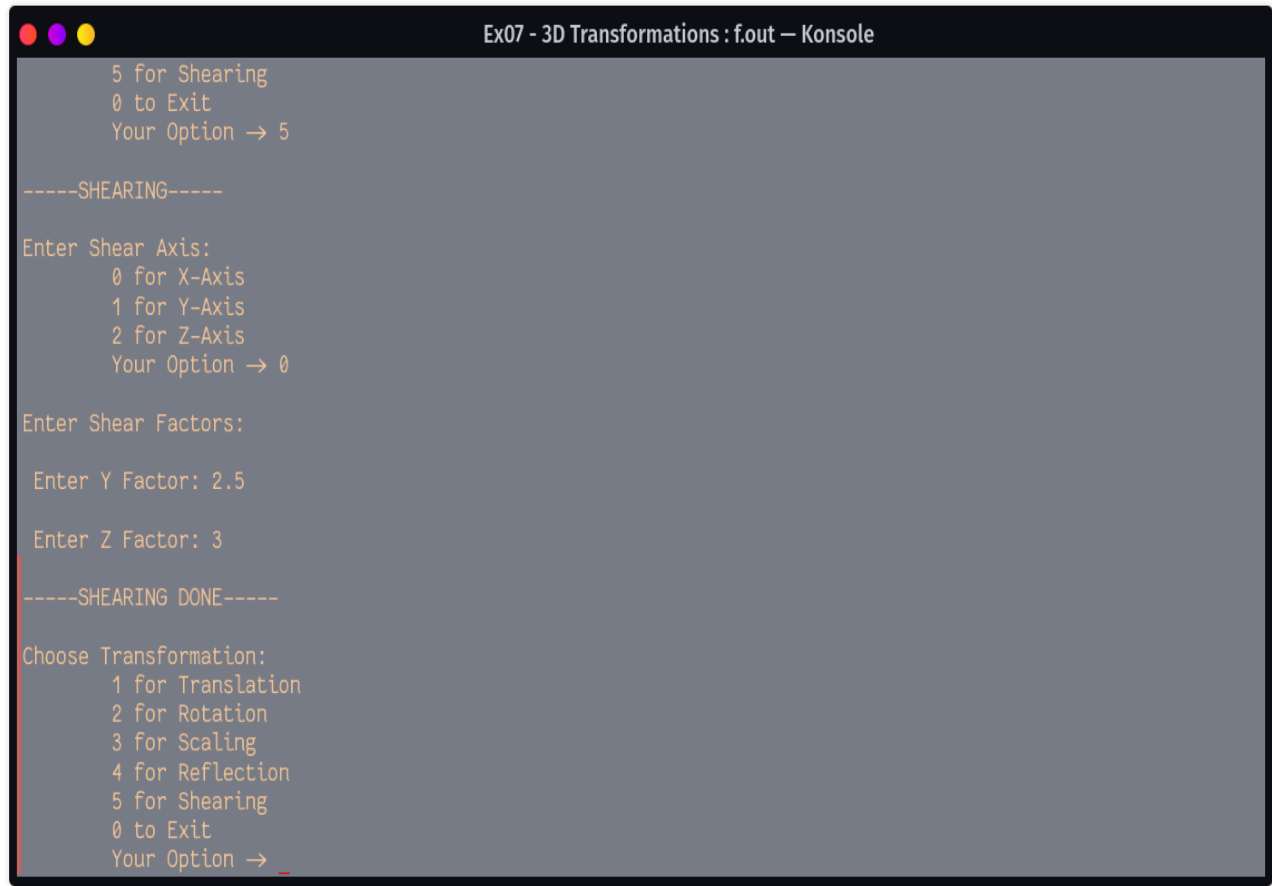
## Output: Reflected 3D Object, ZX Plane

Figure 9: Reflected 3D Object, ZX Plane.



## Output: Console - Shearing, X Axis, (2.5, 3)

Figure 10: Output: Console - Shearing, X Axis, (2.5, 3).



```
Ex07 - 3D Transformations : f.out - Konsole
5 for Shearing
0 to Exit
Your Option → 5

-----SHEARING-----

Enter Shear Axis:
0 for X-Axis
1 for Y-Axis
2 for Z-Axis
Your Option → 0

Enter Shear Factors:

Enter Y Factor: 2.5

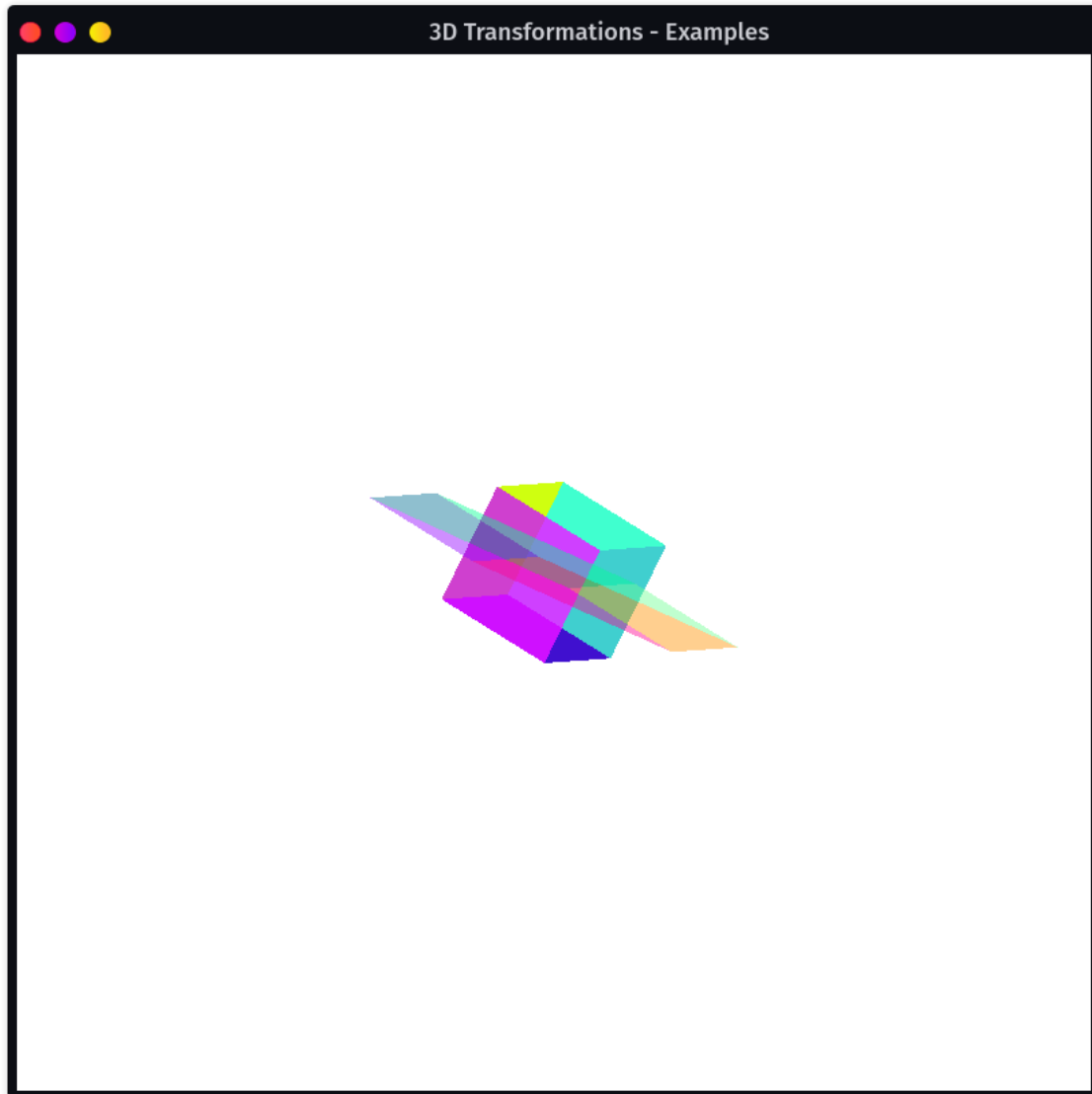
Enter Z Factor: 3

-----SHEARING DONE-----

Choose Transformation:
1 for Translation
2 for Rotation
3 for Scaling
4 for Reflection
5 for Shearing
0 to Exit
Your Option → _
```

**Output: Sheared 3D Object, X Axis, (2.5, 3)**

Figure 11: Sheared 3D Object, X Axis, (2.5, 3).



## Learning Outcome:

- I learnt how to represent a **3D Object** in terms of its **2D Faces** as planar polygons.
- I learnt how to perform **Translation, Rotation, Scaling, Reflection and Shearing** upon a given 3D Object with the use of 3D transformation matrices.
- I learnt about the different transformation matrices and how to calculate them.
- I learnt how to set **depth flags** to enable the **Z-Axis** projections.
- I came to know about inbuilt functions that perform the same transformations like **glRotate3f()** in OpenGL.
- I learnt how to do **orthographic projection** in 3D with OpenGL, using **glOrtho()**.
- I learnt to use parametrized and default constructors in C++ and to call them from other constructors.