

---

# Computer Graphics HW04

---

소프트웨어학부  
소프트웨어학과  
2017012197  
여채린

---

문영식 교수님

---

## 1. Source code for the main part of the program

```
5   GLfloat xRotated_1, yRotated_1, zRotated_1;
6   GLfloat xRotated_2, yRotated_2, zRotated_2;
7   GLfloat xRotated_3, yRotated_3, zRotated_3;
8   GLfloat xRotated_4, yRotated_4, zRotated_4;
9   GLfloat xRotated_5, yRotated_5, zRotated_5;
10  GLuint texture[15];
11  GLfloat obj_size = 0.5f;
12  GLfloat obj_size0 = 0.0f;
13  GLfloat a = 0.5;
14  GLfloat b = 1/(2*(1 + sqrt(5)) / 2);
25  void loadTextureFromFile(char *filename1, char *filename2, char *filename3, char *filename4, char *filename5, char *filename6,
26                          char *filename7, char *filename8, char *filename9, char *filename10, char *filename11, char *filename12,
27                          char *filename13, char *filename14){
28      glClearColor(0,0,0,0);
29      glEnable(GL_DEPTH_TEST);
30      RgbImage theTexMap1(filename1);
31      RgbImage theTexMap2(filename2);
32      RgbImage theTexMap3(filename3);
33      RgbImage theTexMap4(filename4);
34      RgbImage theTexMap5(filename5);
35      RgbImage theTexMap6(filename6);
36      RgbImage theTexMap7(filename7);
37      RgbImage theTexMap8(filename8);
38      RgbImage theTexMap9(filename9);
39      RgbImage theTexMap10(filename10);
40      RgbImage theTexMap11(filename11);
41      RgbImage theTexMap12(filename12);
42      RgbImage theTexMap13(filename13);
43      RgbImage theTexMap14(filename14);

void drawScene(void){
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_TEXTURE_2D);
    glLoadIdentity();

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    GLfloat lightpos[] = {1,1,1,1};
    glLightfv(GL_LIGHT0, GL_POSITION, lightpos);
    GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0};
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
    GLfloat specular[] = {1, 1, 1, 1};
    glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
    GLfloat diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f};
    glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);

170    //Cube
171    glPushMatrix();
172    glTranslatef(-4.0, 0.0, -5);
173    glRotatef(xRotated_1, 1, 0, 0);
174    glRotatef(yRotated_1, 0, 1, 0);
175    glRotatef(zRotated_1, 0, 0, 1);
176    // Making cube (3 quads with texture 6)
177    glBindTexture(GL_TEXTURE_2D, texture[0]);
178    glBegin(GL_QUADS);
179    //Front Face
180    glTexCoord2f(0.0f,0.0f); glVertex3f(-obj_size,-obj_size, obj_size);
181    glTexCoord2f(1.0f,0.0f); glVertex3f(obj_size,-obj_size, obj_size);
182    glTexCoord2f(1.0f,1.0f); glVertex3f(obj_size, obj_size, obj_size);
183    glTexCoord2f(0.0f,1.0f); glVertex3f(-obj_size,obj_size, obj_size);
184    glEnd();
```

```

185 glBindTexture(GL_TEXTURE_2D, texture[1]);
186 glBegin(GL_QUADS);
187     //Back Face
188     glTexCoord2f(1.0f,0.0f); glVertex3f(-obj_size,-obj_size, -obj_size);
189     glTexCoord2f(1.0f,1.0f); glVertex3f(-obj_size,obj_size, -obj_size);
190     glTexCoord2f(0.0f,1.0f); glVertex3f(obj_size,obj_size, -obj_size);
191     glTexCoord2f(0.0f,0.0f); glVertex3f(obj_size,-obj_size, -obj_size);
192 glEnd();
193 glBindTexture(GL_TEXTURE_2D, texture[2]);
194 glBegin(GL_QUADS);
195     //Top Face
196     glTexCoord2f(0.0f,1.0f); glVertex3f(-obj_size,obj_size, -obj_size);
197     glTexCoord2f(0.0f,0.0f); glVertex3f(-obj_size,obj_size, obj_size);
198     glTexCoord2f(1.0f,0.0f); glVertex3f(obj_size,obj_size, obj_size);
199     glTexCoord2f(1.0f,1.0f); glVertex3f(obj_size,obj_size, -obj_size);
200 glEnd();
201 glBindTexture(GL_TEXTURE_2D, texture[3]);
202 glBegin(GL_QUADS);
203     // Bottom Face
204     glTexCoord2f(1.0f, 1.0f); glVertex3f(-obj_size, -obj_size, -obj_size);
205     glTexCoord2f(0.0f, 1.0f); glVertex3f( obj_size, -obj_size, -obj_size);
206     glTexCoord2f(0.0f, 0.0f); glVertex3f( obj_size, -obj_size, obj_size);
207     glTexCoord2f(1.0f, 0.0f); glVertex3f(-obj_size, -obj_size, obj_size);
208 glEnd();
209 glBindTexture(GL_TEXTURE_2D, texture[4]);
210 glBegin(GL_QUADS);
211     //Right face
212     glTexCoord2f(1.0f, 0.0f); glVertex3f( obj_size, -obj_size, -obj_size);
213     glTexCoord2f(1.0f, 1.0f); glVertex3f( obj_size, obj_size, -obj_size);
214     glTexCoord2f(0.0f, 1.0f); glVertex3f( obj_size, obj_size, obj_size);
215     glTexCoord2f(0.0f, 0.0f); glVertex3f( obj_size, -obj_size, obj_size);
216 glEnd();
217 glBindTexture(GL_TEXTURE_2D, texture[5]);
218 glBegin(GL_QUADS);
219     //Left Face
220     glTexCoord2f(0.0f, 0.0f); glVertex3f(-obj_size, -obj_size, -obj_size);
221     glTexCoord2f(1.0f, 0.0f); glVertex3f(-obj_size, -obj_size, obj_size);
222     glTexCoord2f(1.0f, 1.0f); glVertex3f(-obj_size, obj_size, obj_size);
223     glTexCoord2f(0.0f, 1.0f); glVertex3f(-obj_size, obj_size, -obj_size);
224 glEnd();
225 glPopMatrix();
226
227 //Tetrahedron
228 glPushMatrix();
229 glTranslatef(-2.0, 0.0, -5);
230 glRotatef(xRotated_2, 1, 0, 0);
231 glRotatef(yRotated_2, 0, 1, 0);
232 glRotatef(zRotated_2, 0, 0, 1);
233 glBindTexture(GL_TEXTURE_2D, texture[6]);
234 glBegin(GL_TRIANGLES);
235     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
236     glTexCoord2f(1.0f, 0.0f); glVertex3f(obj_size, -obj_size, obj_size); //
237     glTexCoord2f(0.0f, 1.0f);glVertex3f( -obj_size,-obj_size, obj_size); //
238 glEnd();
239 glBindTexture(GL_TEXTURE_2D, texture[7]);
240 glBegin(GL_TRIANGLES);
241     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
242     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0,-obj_size, -obj_size); /
243     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,obj_size); // {
244 glEnd();
245 glBindTexture(GL_TEXTURE_2D, texture[8]);
246 glBegin(GL_TRIANGLES);
247     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); /
248     glTexCoord2f(0.0f, 0.0f);glVertex3f( -obj_size,-obj_size,obj_size); //
249     glTexCoord2f(0.0f, 1.0f);glVertex3f(obj_size0,-obj_size,-obj_size); //
250 glEnd();

```

```

251 glBindTexture(GL_TEXTURE_2D, texture[9]);
252 glBegin(GL_TRIANGLES);
253     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size0, -obj_size, -obj_size);
254     glTexCoord2f(0.0f, 0.0f);glVertex3f( -obj_size,-obj_size,obj_size); //
255     glTexCoord2f(0.0f, 1.0f);glVertex3f(obj_size,-obj_size,obj_size); // {
256 glEnd();
257 glPopMatrix();
258
259 //Pyramid
260 glPushMatrix();
261 glTranslatef(0.0, 0.0, -5);
262 glRotatef(xRotated_3, 1, 0, 0);
263 glRotatef(yRotated_3, 0, 1, 0);
264 glRotatef(zRotated_3, 0, 0, 1);
265 glBindTexture(GL_TEXTURE_2D, texture[10]);
266 glBegin( GL_TRIANGLES );
267     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); //
268     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
269     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); //
270 glEnd();
271 glBindTexture(GL_TEXTURE_2D, texture[13]);
272 glBegin( GL_TRIANGLES );
273     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); //
274     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
275     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
276 glEnd();
277 glBindTexture(GL_TEXTURE_2D, texture[12]);
278 glBegin( GL_TRIANGLES );
279     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
280     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); //
281     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
282 glEnd();
283 glBindTexture(GL_TEXTURE_2D, texture[3]);
284 glBegin(GL_TRIANGLES);
285     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
286     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
287     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
288 glEnd();
289 glBindTexture(GL_TEXTURE_2D, texture[1]);
290 glBegin(GL_QUADS);
291     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
292     glTexCoord2f(1.0f, 1.0f);glVertex3f(obj_size,-obj_size,-obj_size);
293     glTexCoord2f(1.0f, 0.0f);glVertex3f(obj_size,-obj_size,obj_size);
294     glTexCoord2f(0.0f, 0.0f);glVertex3f(-obj_size,-obj_size,obj_size);
295 glEnd();
296 glPopMatrix();
297
298 //Octahedron
299 glPushMatrix();
300 glTranslatef(2.0, 0.0, -5);
301 glRotatef(xRotated_4, 1, 0, 0);
302 glRotatef(yRotated_4, 0, 1, 0);
303 glRotatef(zRotated_4, 0, 0, 1);
304 glBindTexture(GL_TEXTURE_2D, texture[0]);
305 glBegin( GL_TRIANGLES );
306     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
307     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); //
308     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); //
309 glEnd();
310 glBindTexture(GL_TEXTURE_2D, texture[2]);
311 glBegin( GL_TRIANGLES );
312     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
313     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); //
314     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
315 glEnd();
316 glBindTexture(GL_TEXTURE_2D, texture[4]);
317 glBegin( GL_TRIANGLES );
318     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0);
319     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); //
320     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); //
321 glEnd();

```

```

322 glBindTexture(GL_TEXTURE_2D, texture[6]);
323 glBegin(GL_TRIANGLES);
324     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, obj_size, obj_size0); //
325     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
326     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
327 glEnd();
328 glBindTexture(GL_TEXTURE_2D, texture[8]);
329 glBegin( GL_TRIANGLES );
330     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
331     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
332     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size, obj_size); // {
333 glEnd();
334 glBindTexture(GL_TEXTURE_2D, texture[10]);
335 glBegin( GL_TRIANGLES );
336     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
337     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size, obj_size); // {
338     glTexCoord2f(0.0f, 1.0f);glVertex3f( obj_size,-obj_size,-obj_size); // {
339 glEnd();
340 glBindTexture(GL_TEXTURE_2D, texture[12]);
341 glBegin( GL_TRIANGLES );
342     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
343     glTexCoord2f(1.0f, 0.0f);glVertex3f( obj_size,-obj_size,-obj_size); // {
344     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
345 glEnd();
346 glBindTexture(GL_TEXTURE_2D, texture[13]);
347 glBegin(GL_TRIANGLES);
348     glTexCoord2f(0.0f, 0.0f);glVertex3f( obj_size0, -3*obj_size, obj_size0);
349     glTexCoord2f(1.0f, 0.0f);glVertex3f(-obj_size,-obj_size,-obj_size); // {
350     glTexCoord2f(0.0f, 1.0f);glVertex3f(-obj_size,-obj_size, obj_size); // {
351 glEnd();
352 glPopMatrix();
353
354
355 //Icosahedron
356 glPushMatrix();
357 glTranslatef(4.0, 0.0, -5);
358 glRotatef(xRotated_5, 1, 0, 0);
359 glRotatef(yRotated_5, 0, 1, 0);
360 glRotatef(zRotated_5, 0, 0, 1);
361 //A
362 glBindTexture(GL_TEXTURE_2D, texture[7]);
363 glBegin( GL_POLYGON );
364     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
365     glTexCoord2f(1.0f, 0.0f);glVertex3f(b,a,0);
366     glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,a,0);
367 glEnd();
368 //B
369 glBindTexture(GL_TEXTURE_2D, texture[8]);
370 glBegin( GL_POLYGON );
371     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
372     glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,a,0);
373     glTexCoord2f(0.0f, 1.0f);glVertex3f(b,a,0);
374 glEnd();
375 //C
376 glBindTexture(GL_TEXTURE_2D, texture[9]);
377 glBegin( GL_POLYGON );
378     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
379     glTexCoord2f(1.0f, 0.0f);glVertex3f(0,-b,a);
380     glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
381 glEnd();
382 //D
383 glBindTexture(GL_TEXTURE_2D, texture[10]);
384 glBegin( GL_POLYGON );
385     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
386     glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
387     glTexCoord2f(0.0f, 1.0f);glVertex3f(0,-b,a);
388 glEnd();

```

```

389 //E
390 glBindTexture(GL_TEXTURE_2D, texture[0]);
391 glBegin( GL_POLYGON );
392     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
393     glTexCoord2f(1.0f, 0.0f);glVertex3f(0,-b,-a);
394     glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
395 glEnd();
396 //F
397 glBindTexture(GL_TEXTURE_2D, texture[1]);
398 glBegin( GL_POLYGON );
399     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
400     glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
401     glTexCoord2f(0.0f, 1.0f);glVertex3f(0,-b,-a);
402 glEnd();
403 //G
404 glBindTexture(GL_TEXTURE_2D, texture[2]);
405 glBegin( GL_POLYGON );
406     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
407     glTexCoord2f(1.0f, 0.0f);glVertex3f(b,-a,0);
408     glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,-a,0);
409 glEnd();
410 //H
411 glBindTexture(GL_TEXTURE_2D, texture[3]);
412 glBegin( GL_POLYGON );
413     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
414     glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,-a,0);
415     glTexCoord2f(0.0f, 1.0f);glVertex3f(b,-a,0);
416 glEnd();
417 //I
418 glBindTexture(GL_TEXTURE_2D, texture[4]);
419 glBegin( GL_POLYGON );
420     glTexCoord2f(0.0f, 0.0f);glVertex3f(-b,a,0);
421     glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,b);
422     glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,-b);
423 glEnd();
424 //J
425 glBindTexture(GL_TEXTURE_2D, texture[5]);
426 glBegin( GL_POLYGON );
427     glTexCoord2f(0.0f, 0.0f);glVertex3f(-b,-a,0);
428     glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
429     glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
430 glEnd();
431 //K
432 glBindTexture(GL_TEXTURE_2D, texture[6]);
433 glBegin( GL_POLYGON );
434     glTexCoord2f(0.0f, 0.0f);glVertex3f(b,a,0);
435     glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,-b);
436     glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,b);
437 glEnd();
438 //L
439 glBindTexture(GL_TEXTURE_2D, texture[7]);
440 glBegin( GL_POLYGON );
441     glTexCoord2f(0.0f, 0.0f);glVertex3f(b,-a,0);
442     glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
443     glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
444 glEnd();
445 //M
446 glBindTexture(GL_TEXTURE_2D, texture[8]);
447 glBegin( GL_POLYGON );
448     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
449     glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,b);
450     glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,a,0);
451 glEnd();

```



```

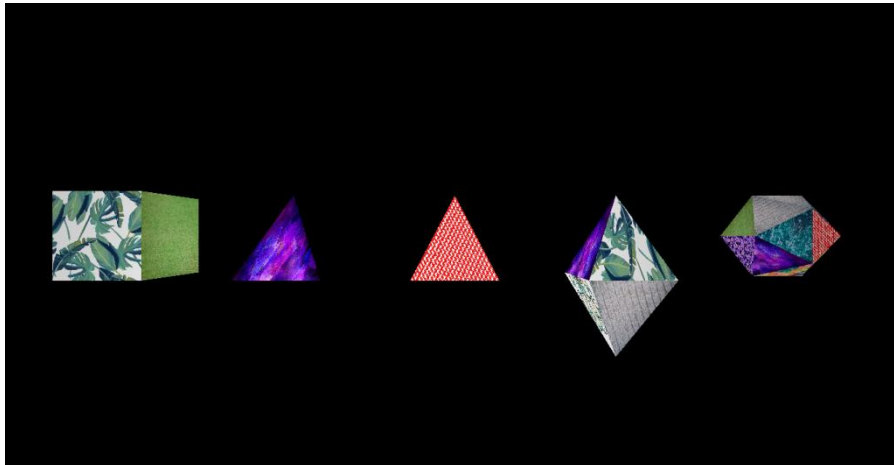
452 //N
453 glBindTexture(GL_TEXTURE_2D, texture[9]);
454 glBegin( GL_POLYGON );
455     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,a);
456     glTexCoord2f(1.0f, 0.0f);glVertex3f(b,a,0);
457     glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,b);
458 glEnd();
459 //O
460 glBindTexture(GL_TEXTURE_2D, texture[10]);
461 glBegin( GL_POLYGON );
462     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
463     glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,a,0);
464     glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,-b);
465 glEnd();
466 //P
467 glBindTexture(GL_TEXTURE_2D, texture[11]);
468 glBegin( GL_POLYGON );
469     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,b,-a);
470     glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,-b);
471     glTexCoord2f(0.0f, 1.0f);glVertex3f(b,a,0);
472 glEnd();
473 //Q
474 glBindTexture(GL_TEXTURE_2D, texture[12]);
475 glBegin( GL_POLYGON );
476     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
477     glTexCoord2f(1.0f, 0.0f);glVertex3f(-a,0,-b);
478     glTexCoord2f(0.0f, 1.0f);glVertex3f(-b,-a,0);
479 glEnd();
480 //R
481 glBindTexture(GL_TEXTURE_2D, texture[13]);
482 glBegin( GL_POLYGON );
483     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,-a);
484     glTexCoord2f(1.0f, 0.0f);glVertex3f(b,-a,0);
485     glTexCoord2f(0.0f, 1.0f);glVertex3f(a,0,-b);
486 glEnd();
487 //S
488 glBindTexture(GL_TEXTURE_2D, texture[6]);
489 glBegin( GL_POLYGON );
490     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
491     glTexCoord2f(1.0f, 0.0f);glVertex3f(-b,-a,0);
492     glTexCoord2f(0.0f, 1.0f);glVertex3f(-a,0,b);
493 glEnd();
494 //T
495 glBindTexture(GL_TEXTURE_2D, texture[2]);
496 glBegin( GL_POLYGON );
497     glTexCoord2f(0.0f, 0.0f);glVertex3f(0,-b,a);
498     glTexCoord2f(1.0f, 0.0f);glVertex3f(a,0,b);
499     glTexCoord2f(0.0f, 1.0f);glVertex3f(b,-a,0);
500 glEnd();
501 glPopMatrix();
538 glFlush();
539 glDisable(GL_TEXTURE_2D);
540 }
541
542 void resizeWindow(int x, int y)
543 {
544     //Set a new projection matrix
545     glMatrixMode(GL_PROJECTION);
546     glLoadIdentity();
547     glOrtho(-1, 1, -1, 1, 1, -1);
548     //Angle of view : 60 degrees
549     //Near clipping plane distance :0.5
550     //Far clipping plane distance : 20.0
551     gluPerspective(60.0, (GLdouble)x/(GLdouble)y, 0.5, 20.0);
552
553     glMatrixMode(GL_MODELVIEW);
554     glViewport(0,0,x,y); //Use the whole window for rendering
555 }

```

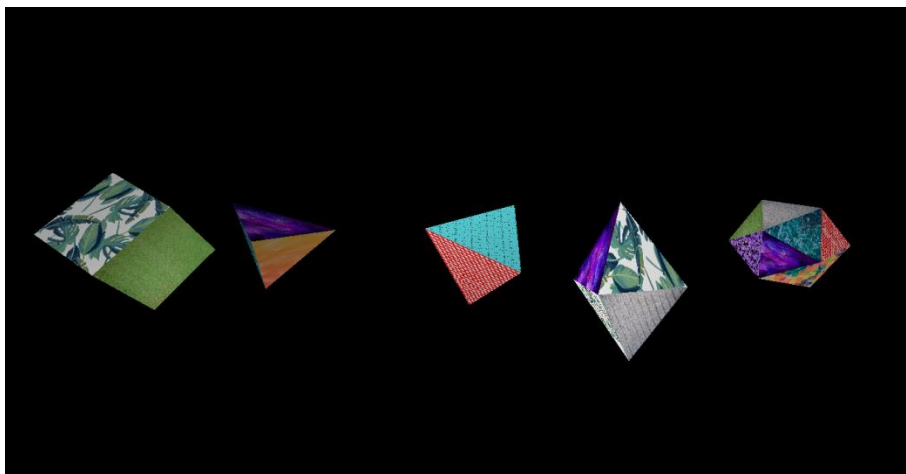
## 2. Screen capture to see the execution result

In order, hexagon, tetrahedron, pyramid, icosahedron in 3D with different textures.

- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {1,1,1,0}**

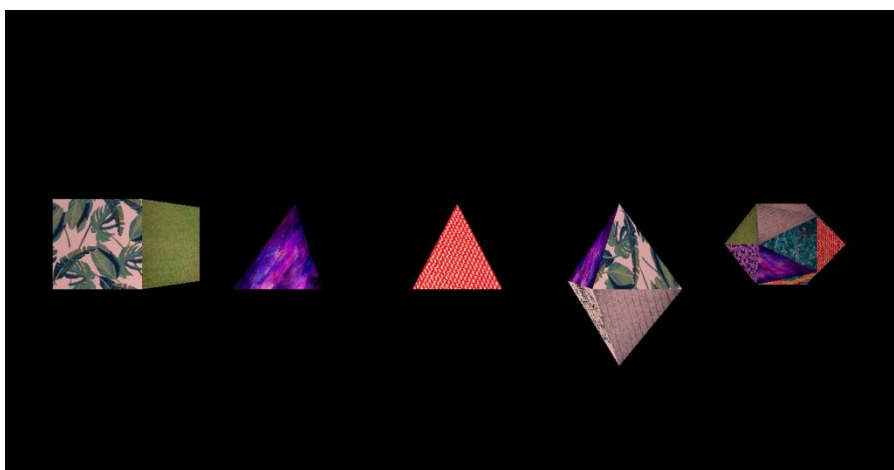


do not rotate



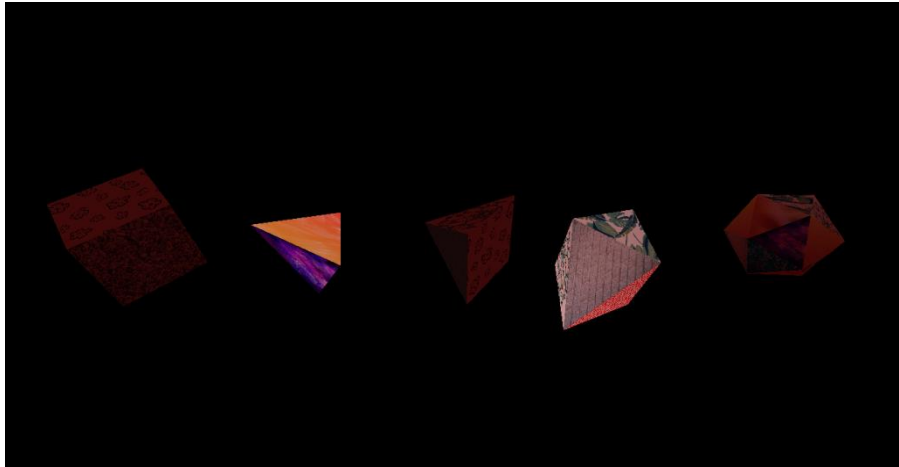
rotate individually

- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {1,0,0,0} (Red Light)**



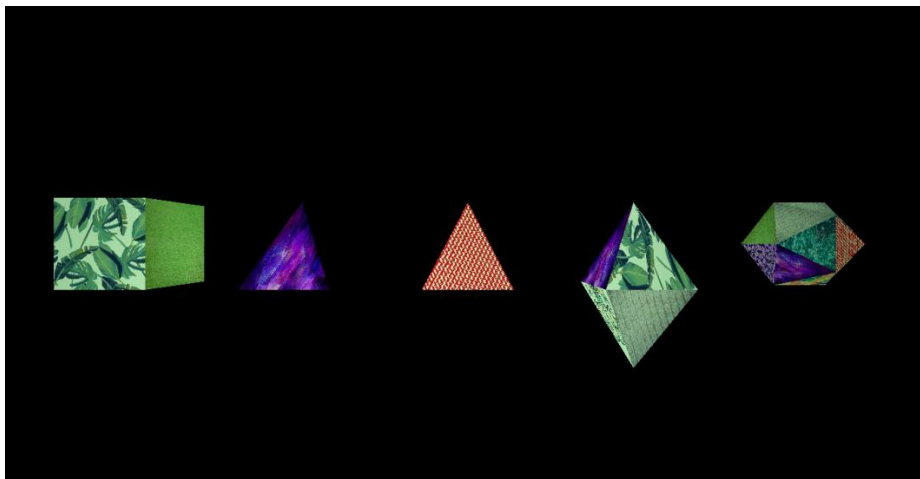
do not rotate



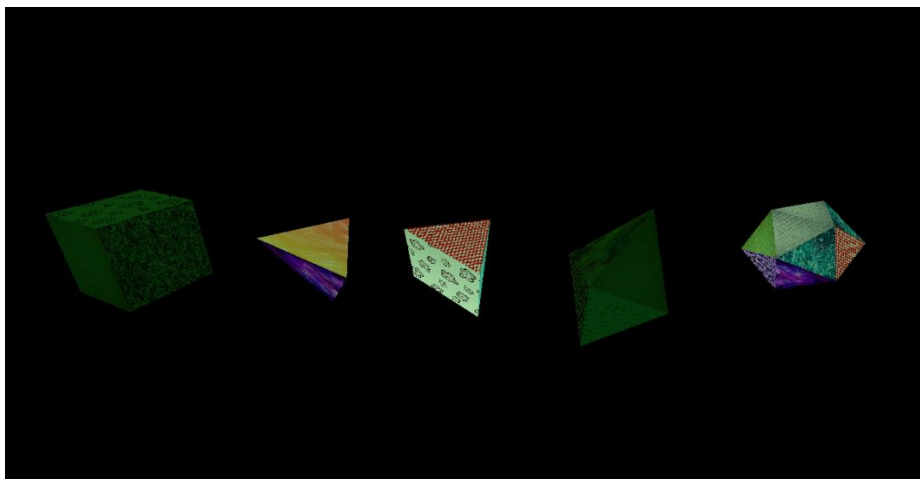


rotate individually

- When use `GLfloat lightpos[] = {-1,-1,-1,1}`, `GLfloat ambient[] = {0,1,0,0}` (Green Light)

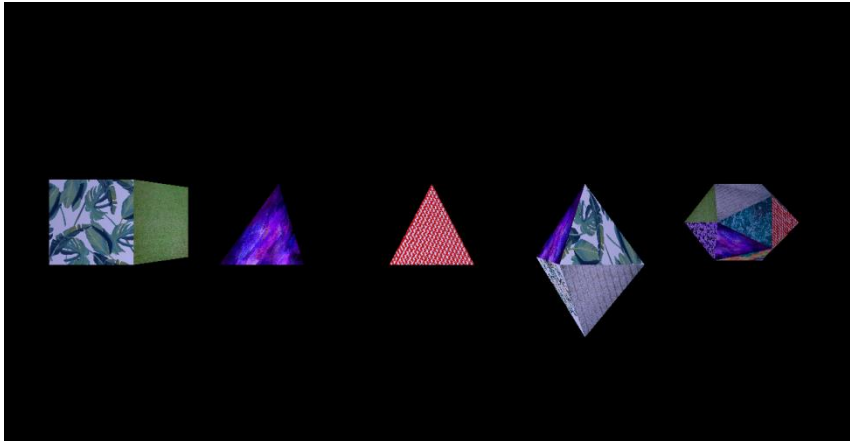


do not rotate

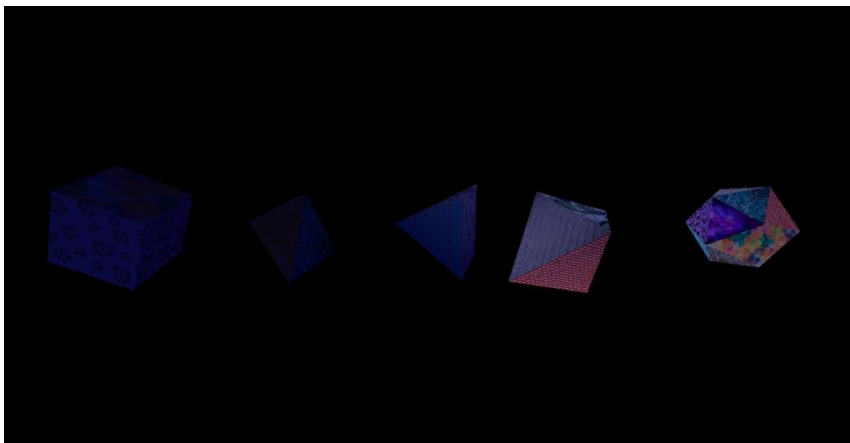


rotate individually

- When use **GLfloat lightpos[] = {-1,-1,-1,1}, GLfloat ambient[] = {0,1,0,0}** (Blue Light)

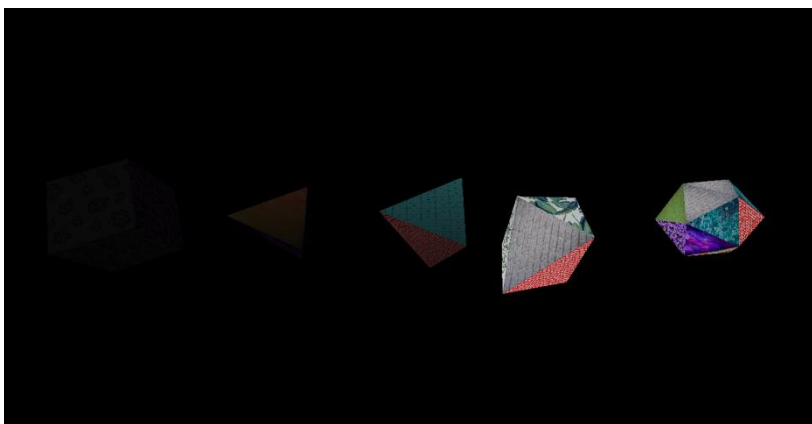


do not rotate



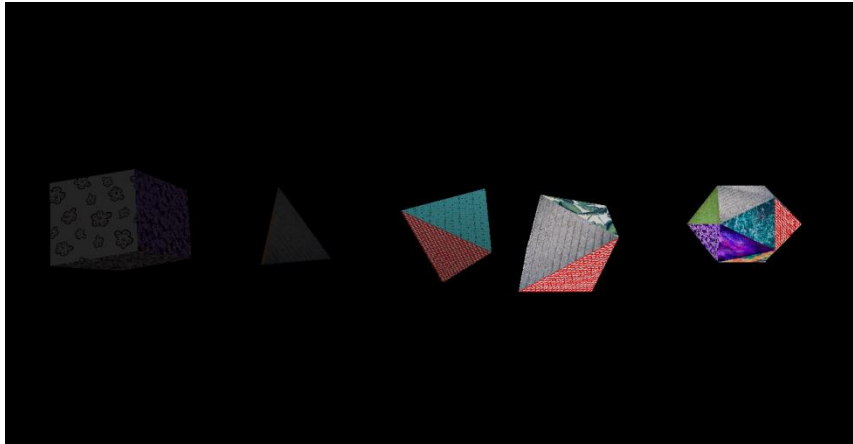
rotate individually

- When use **GLfloat lightpos[] = {1,1,1,1}, GLfloat specular[] = {1, 1, 1, 1 }**



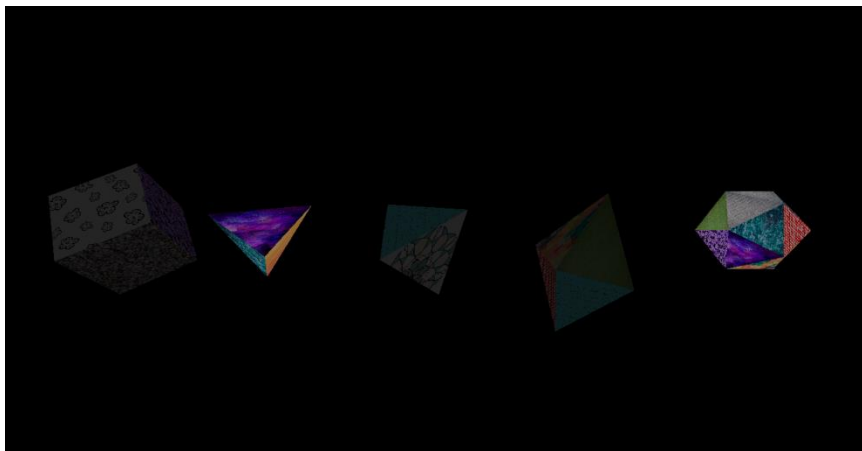
rotate individually

- When use **GLfloat lightpos[] = {1, 1, 1, 1}**, **GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0 }**,  
**GLfloat specular[] = {1, 1, 1, 1 }**



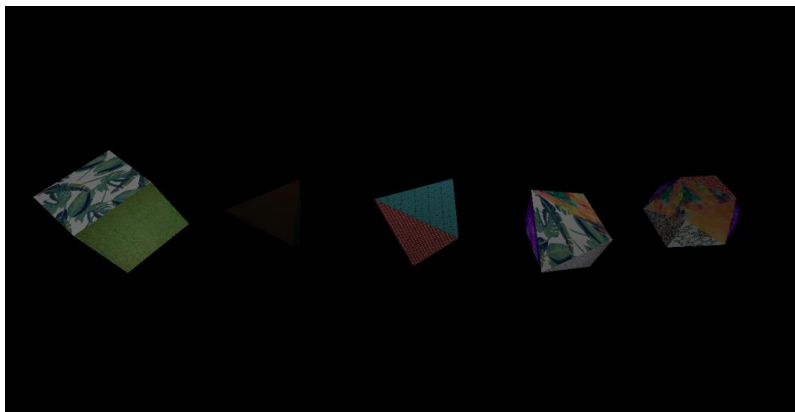
rotate individually

- When use **GLfloat lightpos[] = {0, 0, 0.5, 1}**, **GLfloat ambient[] = {0.5, 0.5, 0.5, 1.0 }**,  
**GLfloat specular[] = {1, 1, 1, 1 }**, **GLfloat diffuse[] = {0.7, 0.7, 0.7, 1.0}**



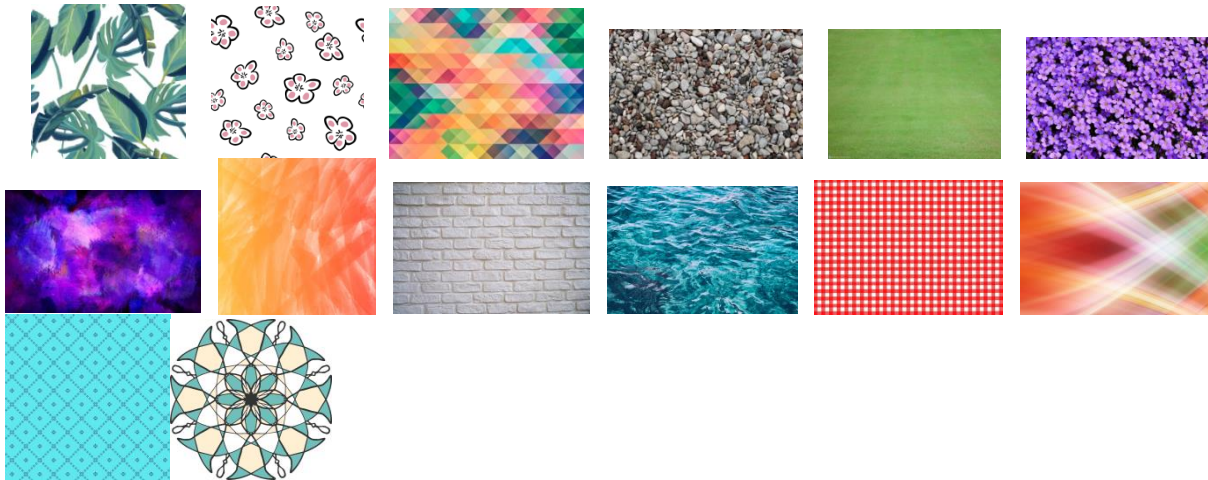
rotate individually

- When use **GLfloat lightpos[] = {1,1,1,1}**, **GLfloat diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f}**



rotate individually

### 3. Analysis and Discussion of Execution Result



In this assignment, I applied a total of 14 textures to a 3D polygon as follows: All textures were downloaded from a free texture download site. I have made a cube, a tetrahedron, a pyramid, an octahedron and an icosahedron. Since the coordinates of the cube, the tetrahedron, the pyramid, and the octahedron are not very complicated, the size of the shape can be changed by two variables, `obj_size` and `obj_size0`. In the case of icosahedron, the coordinate values were calculated by setting a separate variable called `a` and `b`. A detailed description of these variables will follow later.

Here's how to draw each shape in openGL: Between `glPushMatrix ()` and `glPopMatrix ()`, set the position of the figure with `glTranslatef`, and rotate each figure individually using individual `glRotatef`. After selecting the texture to apply with `glBindTexture`, you can use `glVertex3f` to print the vertices of the shape you want to draw between `glBegin ()` and `glEnd ()`. At this time, use `glTexCoord2f` to paste the texture and the drawn shapes together. This will bring the texture you want to the screen.

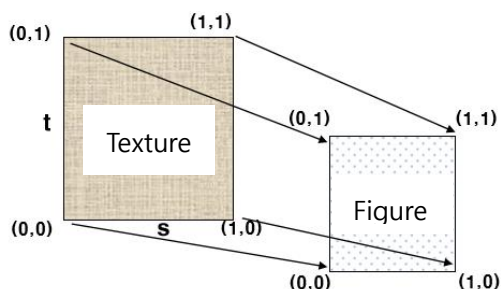


Image from: [https://m.blog.naver.com/PostView.nhn?blogId=enter\\_maintanance&logNo=220945341294&proxyReferer=https%3A%2F%2Fwww.google.com%2F](https://m.blog.naver.com/PostView.nhn?blogId=enter_maintanance&logNo=220945341294&proxyReferer=https%3A%2F%2Fwww.google.com%2F)

For lightning, `GL_AMBIENT`, `GL_SPECULAR`, `GL_DIFFUSE` are used. I also adjusted the position of the light with `GL_POSITION`.

`GL_AMBIENT` is the degree of reflection of ambient light. Specifies the intensity of each color

element in the (R, G, B, A) array. R is Red, G is Green, B is Blue, and A is transparency. Ambient light is light that covers the surroundings without a specific direction. It is expressed in constant brightness and color.

GL\_SPECULAR is the intensity of reflected light. R, G, B, and A values are used. You can adjust the intensity of the reflected light with a value between. Reflected light is light that flows in a specific direction and is completely reflected in one direction. Strong reflected light creates a highlight on the object.

GL\_DIFFUSE is the degree of reflection of diffused light. R, G, B, and A values are used. Diffuse light is light that enters a certain direction and is distributed in various directions on the surface of an object. Although scattered, the surface that receives this light appears brighter than it does. Similar to fluorescent light or sunlight.

In fact, I tried these three lightning effects, and adjusted the colors of the lights by adjusting the R, G, and B values. If there were a curved object, the difference in each lightning effects would be more apparent. Nevertheless, it was confirmed that the contrast of the object becomes more obvious when using specular alone than when using ambient and diffuse.

Now I will explain how I made each polygon. First of all, a cube has the following characteristics: Vertices: 8, Edges: 12, Faces: 6, Edges per face: 4, Edges per vertex: 3, Sin of angle at edge: 1, Surface area:  $6 * \text{edgelenlength}^2$ , Volume:  $\text{edgelenlength}^3$ , Circumscribed radius:  $\sqrt{3} / 2 * \text{edgelenlength}$ , Inscribed radius:  $1/2 * \text{edgelenlength}$ . When calculating Coordinates,

(-1, -1, -1), (1, -1, -1), (1, -1, 1), (-1, -1, 1) / (-1, -1, -1), (-1, -1, 1), (-1, 1, 1), (-1, 1, -1) / (-1, -1, 1), (1, -1, 1), (1, 1, 1), (-1, 1, 1) / (-1, 1, -1), (-1, 1, 1), (1, 1, 1), (1, 1, -1) / (1, -1, -1), (1, 1, -1), (1, 1, 1), (1, -1, 1) / (-1, -1, -1), (-1, 1, -1), (1, 1, -1), (1, -1, -1)

The tetrahedron has the following characteristics : Vertices: 4, Edges: 6, Faces: 4, Edges per face: 3, Edges per vertex: 3, Sin of angle at edge:  $2 * \sqrt{2} / 3$ , Surface area:  $\sqrt{3} * \text{edgelenlength}^2$ , Volume:  $\sqrt{2} / 12 * \text{edgelenlength}^3$ , Circumscribed radius:  $\sqrt{6} / 4 * \text{edgelenlength}$ , Inscribed radius:  $\sqrt{6} / 12 * \text{edgelenlength}$ . When calculating Coordinates,

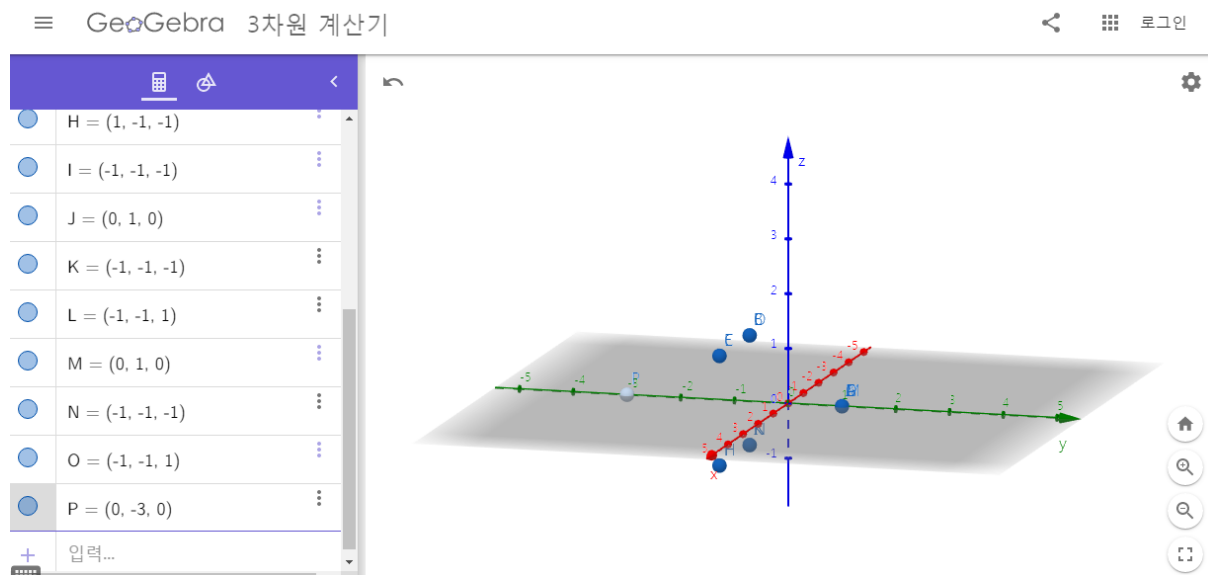
(1, 1, 1), (-1, 1, -1), (1, -1, -1) / (-1, 1, -1), (-1, -1, 1), (1, -1, -1) / (1, 1, 1), (1, -1, -1), (-1, -1, 1) / (1, 1, 1), (-1, -1, 1), (-1, 1, -1)

Octahedron has the following characteristics: Vertices: 6, Edges: 12, Faces: 8, Edges per face: 3, Edges per vertex: 4, Sin of angle at edge:  $2 * \sqrt{2} / 3$ , Surface area:  $2 * \sqrt{3} * \text{edgelenlength}^2$ , Volume:  $\sqrt{2} / 3 * \text{edgelenlength}^3$ , Circumscribed radius:  $\sqrt{2} / 2 * \text{edgelenlength}$ , Inscribed radius:  $\sqrt{6} / 6 * \text{edgelenlength}$ . When calculating Coordinates,

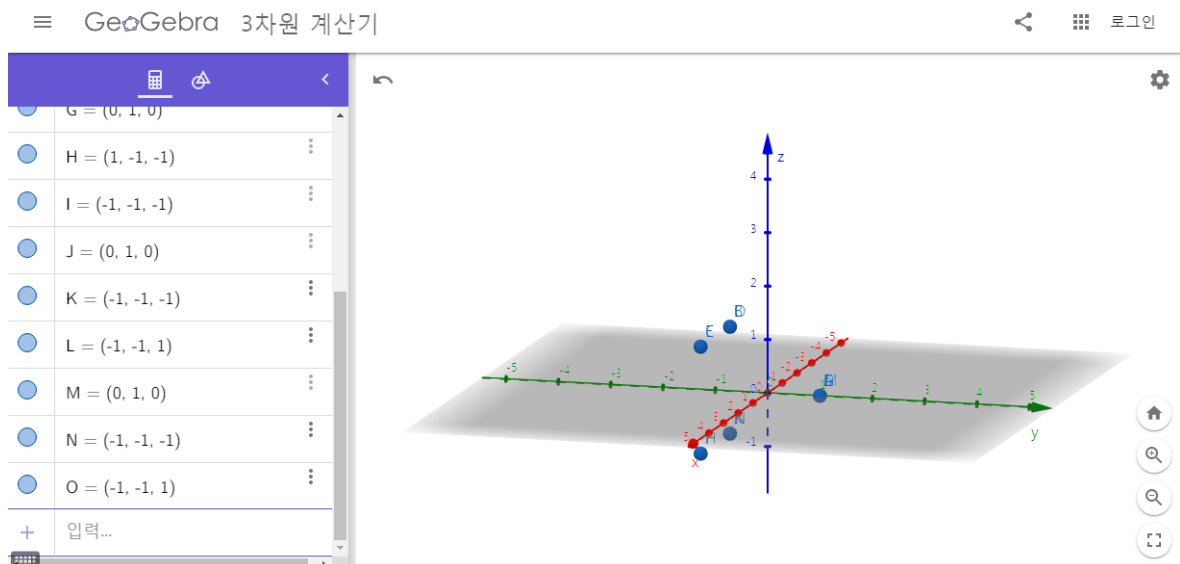
(-a, 0, a), (-a, 0, -a), (0, b, 0) / (-a, 0, -a), (a, 0, -a), (0, b, 0) / (a, 0, -a), (a, 0, a), (0, b, 0) / (a, 0, a), (-a, 0, a), (0, b, 0) / (a, 0, -a), (-a, 0, -a), (0, -b, 0) / (-a, 0, -a), (-a, 0, a), (0, -b, 0) / (a, 0, a), (a, 0, -a),

$(0, -b, 0) / (-a, 0, a), (a, 0, a), (0, -b, 0)$

Or, Coordinates can be created with only 0 and 1. The coordinates of an octahedron created by taking direct coordinates from GeoGebra are as follows.



The pyramid is an application of octahedron. Removing one of the following coordinates from an octahedron creates a pyramid. Note that removing one of the coordinates below will result in a perforated pyramid. Therefore, you must map the texture by specifying the coordinates of the four points on the bottom. I tried to plot the pyramid using only 0,1, like the octahedron.



Icosahedron has the following characteristics: Vertices: 12, Edges: 30, Faces: 20, Edges per face: 3, Edges per vertex: 5, Sin of angle at edge:  $2/3$ , Surface area:  $5 * \sqrt{3} * \text{edgelenh}^2$ , Volume:  $5 * (3 + \sqrt{5}) / 12 * \text{edgelenh}^3$ , Circumscribed radius:  $\sqrt{10 + 2 * \sqrt{5}} / 4 * \text{edgelenh}$ , Inscribed radius:  $\sqrt{42 + 18 * \sqrt{5}} / 12 * \text{edgelenh}$ , When calculating



Coordinates,  $a = 1 / (2 * \sqrt{2})$  and  $b = 1/2$ .  $a = 1/2$  and  $b = 1 / (2 * (1 + \sqrt{5}) / 2)$ . . When calculating Coordinates,

(0, b, -a), (b, a, 0), (-b, a, 0) / (0, b, a), (-b, a, 0), (b, a, 0) / (0, b, a), (0, -b, a), (-a, 0, b) / (0, b, a), (a, 0, b), (0, -b, a) / (0, b, -a), (0, -b, -a), (a, 0, -b) / (0, b, -a), (-a, 0, -b), (0, -b, -a) / (0, -b, a), (b, -a, 0), (-b, -a, 0) / (0, -b, -a), (-b, -a, 0), (b, -a, 0) / (-b, a, 0), (-a, 0, b), (-a, 0, -b) / (-b, -a, 0), (-a, 0, -b), (-a, 0, b) / (b, a, 0), (a, 0, -b), (a, 0, b) / (b, -a, 0), (a, 0, b), (a, 0, -b) / (0, b, a), (-a, 0, b), (-b, a, 0) / (0, b, a), (b, a, 0), (a, 0, b) / (0, b, -a), (-b, a, 0), (-a, 0, -b) / (0, b, -a), (a, 0, -b), (b, a, 0) / (0, -b, -a), (-a, 0, -b), (-b, -a, 0) / (0, -b, -a), (b, -a, 0), (a, 0, -b) / (0, -b, a), (-b, -a, 0), (-a, 0, b) / (0, -b, a), (a, 0, b), (b, -a, 0)

The calculated coordinate points are taken to complete the figure, and different textures are applied to each face. You can reduce or increase the size of each figure by adjusting the size of the coordinates.