



UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

LABORATORIO DE COMPUTACIÓN

GRÁFICA e INTERACCIÓN HUMANO COMPUTADORA



REPORTE DE PRÁCTICA N° 03

NOMBRE COMPLETO: Miranda González José Francisco

N° de Cuenta: 318222327

GRUPO DE LABORATORIO: 03

GRUPO DE TEORÍA: 04

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 01/09/24

CALIFICACIÓN: _____

REPORTE DE PRÁCTICA:

Reporte de práctica 3: modelado geométrico

Instrucciones:

1.- Generar una pirámide rubik (pyraminx) de 9 pirámides por cara.

Cada cara de la pyraminx que se vea de un color diferente y que se vean las separaciones entre instancias (las líneas oscuras son las que permiten diferenciar cada pirámide pequeña)

Agregar en su documento escrito las capturas de pantalla necesarias para que se vean las 4 caras de toda la pyraminx o un video en el cual muestra las 4 caras

Actividad realizada

Para completar la actividad solicitada realice los siguiente:

En la función CrearPiramideTriangular(), cambie los vértices para que todas las caras fueran del mismo tamaño.

```
84 // Pirámide triangular regular
85 void CrearPiramideTriangular()
86 {
87     unsigned int indices_piramide_triangular[] = {
88         0,1,2,
89         1,3,2,
90         3,0,2,
91         1,0,3
92     };
93
94     //GLfloat vertices_piramide_triangular[] = {
95     //    -0.5f, -0.5f, 0.0f, //0
96     //    0.5f, -0.5f, 0.0f, //1
97     //    0.0f, 0.5f, -0.25f, //2
98     //    0.0f, -0.5f, -0.5f, //3
99
100    //};
101    GLfloat vertices_piramide_triangular[] = {
102        -0.5f, -0.5f, 0.0f, //0
103        0.5f, -0.5f, 0.0f, //1
104        0.0f, 0.5f, 0.0f, //2
105        0.0f, 0.0f, 1.0f, //3
106    };
107
108    Mesh* obj1 = new Mesh();
109    obj1->CreateMesh(vertices_piramide_triangular, indices_piramide_triangular, 12, 12);
110    meshList.push_back(obj1);
111
112 }
```

Dentro de main(), cambie el tamaño de la ventana para que cada lado fuera de la misma medida.

```
293      mainWindow = Window(800, 800);
```

Dentro del while de main() se realizaron los cambios mas importantes, pues fue ahí donde construí las pirámides.

Pirámide negra:

Esta es la base en la que coloque todas las demás pirámides

```
355      // PIRAMIDE PRINCIPAL (NEGRO)
356
357      model = glm::mat4(1.0f);
358      //Traslación inicial para posicionar en -Z a los objetos
359      model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.0f));
360      model = glm::scale(model, glm::vec3(1.0f, 1.0f, 1.0f));
361      //otras transformaciones para el objeto
362      //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
363      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model));
364      //la línea de proyección solo se manda una vez a menos que en tiempo de ejecución
365      //se programe cambio entre proyección ortogonal y perspectiva
366      glUniformMatrix4fv(uniformProjection, 1, GL_FALSE, glm::value_ptr(projection));
367      glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
368      color = glm::vec3(0.0f, 0.0f, 0.0f);
369      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
370      meshList[1]-->RenderMesh(); //dibuja cubo y pirámide triangular
371      //meshList[3]-->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
372      //sp.render(); //dibuja esfera
```

Pirámides verdes (9):

```
375      // CARA (VERDE)
376
377      model = glm::mat4(1.0f);
378      color = glm::vec3(0.0f, 1.0f, 0.0f);
379      //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
380      model = glm::translate(model, glm::vec3(0.0f, 0.30f, -4.005f));
381      model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
382      //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
383      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
384      glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
385      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
386      meshList[1]-->RenderMesh();
387
388      model = glm::mat4(1.0f);
389      color = glm::vec3(0.0f, 1.0f, 0.0f);
390      //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
391      model = glm::translate(model, glm::vec3(0.165f, -0.02f, -4.005f));
392      model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
393      //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
394      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
395      glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
396      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
397      meshList[1]-->RenderMesh();
398
399      model = glm::mat4(1.0f);
400      color = glm::vec3(0.0f, 1.0f, 0.0f);
401      //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
402      model = glm::translate(model, glm::vec3(-0.165f, -0.02f, -4.005f));
403      model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
404      //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
405      glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
406      glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
407      glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
408      meshList[1]-->RenderMesh();
```

```

410     model = glm::mat4(1.0f);
411     color = glm::vec3(0.0f, 1.0f, 0.0f);
412     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
413     model = glm::translate(model, glm::vec3(0.0f, -0.02f, -4.005f));
414     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
415     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
416     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
417     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
418     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
419     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
420     meshList[1]-->RenderMesh();
421
422     model = glm::mat4(1.0f);
423     color = glm::vec3(0.0f, 1.0f, 0.0f);
424     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
425     model = glm::translate(model, glm::vec3(-0.32f, -0.33f, -4.005f));
426     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
427     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
428     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
429     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
430     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
431     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
432     meshList[1]-->RenderMesh();
433
434     model = glm::mat4(1.0f);
435     color = glm::vec3(0.0f, 1.0f, 0.0f);
436     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
437     model = glm::translate(model, glm::vec3(-0.16f, -0.33f, -4.005f));
438     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
439     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
440     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
441     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
442     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
443     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
444     meshList[1]-->RenderMesh();

```

```

445     model = glm::mat4(1.0f);
446     color = glm::vec3(0.0f, 1.0f, 0.0f);
447     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
448     model = glm::translate(model, glm::vec3(0.001f, -0.33f, -4.005f));
449     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
450     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
451     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
452     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
453     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
454     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
455     meshList[1]-->RenderMesh();
456
457     model = glm::mat4(1.0f);
458     color = glm::vec3(0.0f, 1.0f, 0.0f);
459     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
460     model = glm::translate(model, glm::vec3(0.16f, -0.33f, -4.005f));
461     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
462     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
463     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
464     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
465     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
466     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
467     meshList[1]-->RenderMesh();
468
469     model = glm::mat4(1.0f);
470     color = glm::vec3(0.0f, 1.0f, 0.0f);
471     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
472     model = glm::translate(model, glm::vec3(0.32f, -0.33f, -4.005f));
473     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.1f));
474     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
475     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
476     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
477     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
478     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
479     meshList[1]-->RenderMesh();

```

Pirámides rojas (9):

```

479     // CARA (ROJO)
480
481     model = glm::mat4(1.0f);
482     color = glm::vec3(1.0f, 0.0f, 0.0f);
483     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
484     model = glm::translate(model, glm::vec3(-0.02f, 0.31f, -3.96f));
485     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
486     //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
487     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
488     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
489     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
490     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
491     meshList[1]-->RenderMesh();
492
493     model = glm::mat4(1.0f);
494     color = glm::vec3(1.0f, 0.0f, 0.0f);
495     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
496     model = glm::translate(model, glm::vec3(-0.078f, 0.16f, -3.715f));
497     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
498     model = glm::rotate(model, 210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
499     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
500     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
501     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
502     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
503     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
504     meshList[1]-->RenderMesh();

```

```

506     model = glm::mat4(1.0f);
507     color = glm::vec3(1.0f, 0.0f, 0.0f);
508     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
509     model = glm::translate(model, glm::vec3(-0.21f, -0.15f, -3.72f));
510     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
511     model = glm::rotate(model, 210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
512     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
513     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
514     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
515     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
516     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
517     meshList[1]-->RenderMesh();
518
519     model = glm::mat4(1.0f);
520     color = glm::vec3(1.0f, 0.0f, 0.0f);
521     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
522     model = glm::translate(model, glm::vec3(-0.04f, 0.01f, -3.39f));
523     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
524     model = glm::rotate(model, 210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
525     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
526     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
527     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
528     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
529     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
530     meshList[1]-->RenderMesh();
531
532     model = glm::mat4(1.0f);
533     color = glm::vec3(1.0f, 0.0f, 0.0f);
534     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
535     model = glm::translate(model, glm::vec3(-0.17f, 0.0f, -3.96f));
536     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
537     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
538     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
539     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
540     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
541     meshList[1]-->RenderMesh();

```

```

543     model = glm::mat4(1.0f);
544     color = glm::vec3(1.0f, 0.0f, 0.0f);
545     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
546     model = glm::translate(model, glm::vec3(-0.02f, 0.165f, -3.645f));
547     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
548     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
549     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
550     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
551     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
552     meshList[1]-->RenderMesh();
553
554     model = glm::mat4(1.0f);
555     color = glm::vec3(1.0f, 0.0f, 0.0f);
556     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
557     model = glm::translate(model, glm::vec3(-0.167f, -0.155f, -3.65f));
558     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
559     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
560     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
561     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
562     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
563     meshList[1]-->RenderMesh();
564
565     model = glm::mat4(1.0f);
566     color = glm::vec3(1.0f, 0.0f, 0.0f);
567     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
568     model = glm::translate(model, glm::vec3(-0.015f, 0.005f, -3.33f));
569     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
570     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
571     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
572     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
573     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
574     meshList[1]-->RenderMesh();

```

```

576     model = glm::mat4(1.0f);
577     color = glm::vec3(1.0f, 0.0f, 0.0f);
578     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
579     model = glm::translate(model, glm::vec3(-0.33f, -0.32f, -3.96f));
580     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
581     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
582     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
583     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
584     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
585     meshList[1]-->RenderMesh();

```

Pirámides azules (9):

```

587     // CARA (AZUL)
588
589     model = glm::mat4(1.0f);
590     color = glm::vec3(0.0f, 0.0f, 1.0f);
591     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
592     model = glm::translate(model, glm::vec3(0.02f, 0.31f, -3.96f));
593     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
594     //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
595     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
596     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
597     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
598     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
599     meshList[1]-->RenderMesh();

```

```

601     model = glm::mat4(1.0f);
602     color = glm::vec3(0.0f, 0.0f, 1.0f);
603     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
604     model = glm::translate(model, glm::vec3(0.078f, 0.16f, -3.715f));
605     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
606     model = glm::rotate(model, -210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
607     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
608     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
609     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
610     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
611     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
612     meshList[1]-->RenderMesh();
613
614     model = glm::mat4(1.0f);
615     color = glm::vec3(0.0f, 0.0f, 1.0f);
616     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
617     model = glm::translate(model, glm::vec3(0.21f, -0.15f, -3.72f));
618     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
619     model = glm::rotate(model, -210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
620     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
621     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
622     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
623     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
624     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
625     meshList[1]-->RenderMesh();
626
627     model = glm::mat4(1.0f);
628     color = glm::vec3(0.0f, 0.0f, 1.0f);
629     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
630     model = glm::translate(model, glm::vec3(0.04f, 0.81f, -3.39f));
631     model = glm::scale(model, glm::vec3(0.22f, 0.22f, 0.22f));
632     model = glm::rotate(model, -210 * toRadians, glm::vec3(0.0f, 1.0f, 0.0f));
633     model = glm::rotate(model, -5 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
634     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
635     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
636     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
637     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
638     meshList[1]-->RenderMesh();

```

```

640     model = glm::mat4(1.0f);
641     color = glm::vec3(0.0f, 0.0f, 1.0f);
642     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
643     model = glm::translate(model, glm::vec3(0.17f, 0.0f, -3.96f));
644     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
645     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
646     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
647     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
648     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
649     meshList[1]-->RenderMesh();
650
651     model = glm::mat4(1.0f);
652     color = glm::vec3(0.0f, 0.0f, 1.0f);
653     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
654     model = glm::translate(model, glm::vec3(0.02f, 0.165f, -3.645f));
655     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
656     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
657     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
658     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
659     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
660     meshList[1]-->RenderMesh();
661
662     model = glm::mat4(1.0f);
663     color = glm::vec3(0.0f, 0.0f, 1.0f);
664     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
665     model = glm::translate(model, glm::vec3(0.167f, -0.155f, -3.65f));
666     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
667     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
668     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
669     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
670     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
671     meshList[1]-->RenderMesh();

```

```

673     model = glm::mat4(1.0f);
674     color = glm::vec3(0.0f, 0.0f, 1.0f);
675     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
676     model = glm::translate(model, glm::vec3(0.015f, 0.085f, -3.33f));
677     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
678     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
679     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
680     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
681     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
682     meshList[1]-->RenderMesh();
683
684     model = glm::mat4(1.0f);
685     color = glm::vec3(0.0f, 0.0f, 1.0f);
686     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
687     model = glm::translate(model, glm::vec3(0.33f, -0.32f, -3.96f));
688     model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
689     //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
690     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
691     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
692     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
693     meshList[1]-->RenderMesh();

```


Pirámides amarillas (9):

```
695 // CARA (AMARILLA)
696
697 model = glm::mat4(1.0f);
698 color = glm::vec3(1.0f, 1.0f, 0.0f);
699 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
700 model = glm::translate(model, glm::vec3(0.0f, -0.013f, -3.31f));
701 model = glm::scale(model, glm::vec3(0.29f, 0.29f, 0.29f));
702 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
703 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
704 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
705 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
706 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
707 meshList[1] -> RenderMesh();
708
709 model = glm::mat4(1.0f);
710 color = glm::vec3(1.0f, 1.0f, 0.0f);
711 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
712 model = glm::translate(model, glm::vec3(-0.33f, -0.35f, -3.99f));
713 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
714 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
715 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
716 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
717 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
718 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
719 meshList[1] -> RenderMesh();
720
721 model = glm::mat4(1.0f);
722 color = glm::vec3(1.0f, 1.0f, 0.0f);
723 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
724 model = glm::translate(model, glm::vec3(0.0f, -0.35f, -3.99f));
725 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
726 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
727 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
728 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
729 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
730 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
731 meshList[1] -> RenderMesh();
```

```
733 model = glm::mat4(1.0f);
734 color = glm::vec3(1.0f, 1.0f, 0.0f);
735 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
736 model = glm::translate(model, glm::vec3(0.33f, -0.35f, -3.99f));
737 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
738 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
739 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
740 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
741 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
742 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
743 meshList[1] -> RenderMesh();
744
745 model = glm::mat4(1.0f);
746 color = glm::vec3(1.0f, 1.0f, 0.0f);
747 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
748 model = glm::translate(model, glm::vec3(-0.16f, -0.18f, -3.65f));
749 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
750 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
751 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 1.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
752 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
753 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
754 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
755 meshList[1] -> RenderMesh();
756
757 model = glm::mat4(1.0f);
758 color = glm::vec3(1.0f, 1.0f, 0.0f);
759 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
760 model = glm::translate(model, glm::vec3(0.16f, -0.18f, -3.65f));
761 model = glm::scale(model, glm::vec3(0.3f, 0.3f, 0.3f));
762 //model = glm::rotate(model, 45 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
763 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(0.0f, 0.0f, 1.0f)); //al presionar la tecla Y se rota sobre el eje y
764 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
765 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
766 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
767 meshList[1] -> RenderMesh();
```

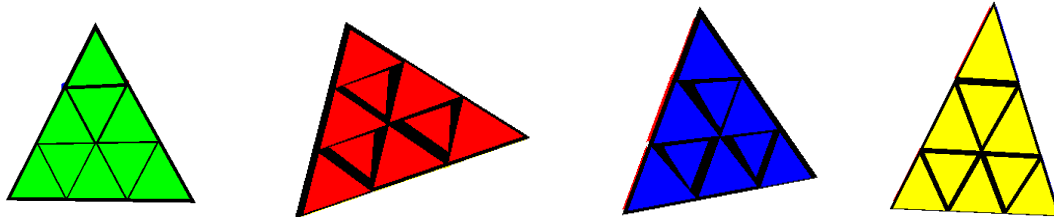
```
769 model = glm::mat4(1.0f);
770 color = glm::vec3(1.0f, 1.0f, 0.0f);
771 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
772 model = glm::translate(model, glm::vec3(0.0f, -0.099f, -3.45f));
773 model = glm::scale(model, glm::vec3(0.28f, 0.28f, 0.28f));
774 model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
775 model = glm::rotate(model, 235 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
776 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
777 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
778 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
779 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
780 meshList[1] -> RenderMesh();
781
782 model = glm::mat4(1.0f);
783 color = glm::vec3(1.0f, 1.0f, 0.0f);
784 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
785 model = glm::translate(model, glm::vec3(-0.165f, -0.26f, -3.795f));
786 model = glm::scale(model, glm::vec3(0.28f, 0.28f, 0.28f));
787 model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
788 model = glm::rotate(model, 235 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
789 //model = glm::rotate(model, glm::radians(mainWindow.getrotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
790 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
791 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
792 glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
793 meshList[1] -> RenderMesh();
```

```

795     model = glm::mat4(1.0f);
796     color = glm::vec3(1.0f, 1.0f, 0.0f);
797     //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
798     model = glm::translate(model, glm::vec3(0.17f, -0.26f, -3.795f));
799     model = glm::scale(model, glm::vec3(0.28f, 0.28f, 0.28f));
800     model = glm::rotate(model, 180 * toRadians, glm::vec3(0.0f, 0.0f, 1.0f));
801     model = glm::rotate(model, 235 * toRadians, glm::vec3(1.0f, 0.0f, 0.0f));
802     //model = glm::rotate(model, glm::radians(mainWindow.getRotay()), glm::vec3(1.0f, 0.0f, 0.0f)); //al presionar la tecla Y se rota sobre el eje y
803     glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
804     glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
805     glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
806     meshList[1] -> RenderMesh();

```

Ejecución del programa:



Problemas presentados

1. Cara roja y azul:
El problema que tuve fue colocar las tres pirámides invertidas en la cara roja y azul, pues como se puede ver en las imágenes no quedaron tan bien como las que están a sus lados.
2. Cámara :
Aunque ya me acostumbré más a la cámara aun me resulta difícil conseguir ciertos ángulos

Conclusión

La practica en si no era tan difícil, pero a mi parecer lo que la complicaba eran las pirámides invertidas.

En las caras verde y amarillo no tuve ningún problema en rotar las pirámides, pues solo era sobre un eje. En cuanto a las caras rojo y azul, no solo tenia que rotar las pirámides sobre un solo eje y esto complicaba mucho acomodarlas de la manera correcta, además que no lograba pegarlas por completo a la pirámide negra.

En general creo que lo solicitado en las instrucciones se cumplió casi en su totalidad, ya que las pirámides invertidas fueron el problema.

Bibliografía

Stack Overflow. (última edición: hace 3 meses). glm rotate usage in Opengl. Consultado el 31 de agosto del 2024 de <https://stackoverflow.com/questions/8844585/glm-rotate-usage-in-opengl>