



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA
DIVISIÓN DE INGENIERÍA ELÉCTRICA
INGENIERÍA EN COMPUTACIÓN
COMPUTACIÓN GRÁFICA e INTERACCIÓN HUMANO
COMPUTADORA



Reporte de práctica 3: modelado geométrico

NOMBRE COMPLETO: Gonzalez Villalba Bryan Jesus

Nº de Cuenta: 421530869

GRUPO DE LABORATORIO: 11

GRUPO DE TEORÍA: 4

SEMESTRE 2025-1

FECHA DE ENTREGA LÍMITE: 4 de septiembre del 2024

CALIFICACIÓN: _____

Introducción:

Para este reporte dibujaremos una pirámide triangular en la que dibujaremos las caras como de una pirámide en la cual una cara es roja, otra verde, otra azul y una celeste, para esto ocuparemos código de prácticas pasadas.

Desarrollo:

1. Primero dibujamos la pirámide color negra en un fondo blanco, imagen 1.0.

```
364 color = glm::vec3(1.0f, 0.0f, 1.0f);
365 //glUniform3fv(uniformColor, 1, glm::value_ptr(color)); //para cambiar el color del objetos
366 //meshList[0]->RenderMesh(); //dibuja cubo y pirámide triangular
367 //meshList[3]->RenderMeshGeometry(); //dibuja las figuras geométricas cilindro, cono, pirámide base cuadrangular
368 //sp.render(); //dibuja esfera
369
370
371
372 //ejercicio: Instanciar primitivas geométricas para recrear el dibujo de la práctica pasada en 3D,
373 //se requiere que exista piso y la casa tiene una ventana azul circular justo en medio de la pared trasera y solo 1 puerta front
374
375 // Pirámide (negro)
376 model = glm::mat4(1.0f);
377 color=glm::vec3(0.0f, 0.0f, 0.0f);
378 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
379 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
380 model = glm::translate(model, glm::vec3(0.0f, 0.1f, -4.0f));
381 model = glm::scale(model, glm::vec3(7.0f, 7.0f, 7.0f));
382 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
383 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
384 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
385 meshList[1]->RenderMesh();
```

Imagen 1.0

2. Dibujamos la cara color rojo, dibujando 9 pirámides triangulares de color rojo, con coordenadas y rotación diferentes, pero en todo lo demás igual, esta es la cara que más rápido me salió, imagen 2.0 a 2.8.

```
387 // Triangulo 1 cara (rojo)
388 model = glm::mat4(1.0f);
389 color = glm::vec3(1.0f, 0.0f, 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(0.0f, 0.0f, -3.0f));
392 model = glm::translate(model, glm::vec3(0.0f, 2.1f, -5.3f));
393 model = glm::rotate(model, glm::radians(-11.0f), glm::vec3(1.0f, 0.0f, 0.0f));
394 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
395 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
396 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
397 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
398 meshList[1]->RenderMesh();
```

Imagen 2.0

```
400 // Triangulo 2 cara (rojo)
401 model = glm::mat4(1.0f);
402 color = glm::vec3(1.0f, 0.0f, 0.0f);
403 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
404 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
405 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -4.8f));
406 model = glm::rotate(model, glm::radians(-16.0f), glm::vec3(1.0f, 0.0f, 0.0f));
407 model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
408 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
409 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
410 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
411 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
412 meshList[1]->RenderMesh();
```

Imagen 2.1

```
414 // Triangulo 3 cara (rojo)
415 model = glm::mat4(1.0f);
416 color = glm::vec3(1.0f, 0.0f, 0.0f);
417 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
418 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
419 model = glm::translate(model, glm::vec3(-1.1f, 0.0f, -4.8f));
420 model = glm::rotate(model, glm::radians(-12.0f), glm::vec3(1.0f, 0.0f, 0.0f));
421 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
422 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
423 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
424 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
425 meshList[1]->RenderMesh();
```

Imagen 2.2

```

427 // Triangulo 4 cara (rojo)
428 model = glm::mat4(1.0f);
429 color = glm::vec3(1.0f, 0.0f, 0.0f);
430 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
431 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
432 model = glm::translate(model, glm::vec3(1.1f, 0.0f, -4.8f));
433 model = glm::rotate(model, glm::radians(-12.0f), glm::vec3(1.0f, 0.0f, 0.0f));
434 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
435 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
436 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
437 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
438 meshList[1] -> RenderMesh();

```

Imagen 2.3

```

440 // Triangulo 5 cara (rojo)
441 model = glm::mat4(1.0f);
442 color = glm::vec3(1.0f, 0.0f, 0.0f);
443 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
444 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
445 model = glm::translate(model, glm::vec3(-2.2f, -2.1f, -4.25f));
446 model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
447 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
448 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
449 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
450 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
451 meshList[1] -> RenderMesh();

```

Imagen 2.4

```

453 // Triangulo 6 cara (rojo)
454 model = glm::mat4(1.0f);
455 color = glm::vec3(1.0f, 0.0f, 0.0f);
456 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
457 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
458 model = glm::translate(model, glm::vec3(-1.1f, -2.1f, -4.25f));
459 model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(1.0f, 0.0f, 0.0f));
460 model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
461 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
462 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
463 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
464 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
465 meshList[1] -> RenderMesh();

```

Imagen 2.5

```

467 // Triangulo 7 cara (rojo)
468 model = glm::mat4(1.0f);
469 color = glm::vec3(1.0f, 0.0f, 0.0f);
470 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
471 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
472 model = glm::translate(model, glm::vec3(0.0f, -2.1f, -4.25f));
473 model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
474 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
475 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
476 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
477 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
478 meshList[1] -> RenderMesh();

```

Imagen 2.6

```

480 // Triangulo 8 cara (rojo)
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(0.0f, 0.0f, -3.0f));
485 model = glm::translate(model, glm::vec3(1.1f, -2.1f, -4.25f));
486 model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(1.0f, 0.0f, 0.0f));
487 model = glm::rotate(model, glm::radians(180.0f), glm::vec3(0.0f, 0.0f, 1.0f));
488 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
489 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
490 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
491 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
492 meshList[1] -> RenderMesh();

```

Imagen 2.7

```

494 // Triangulo 9 cara (rojo)
495 model = glm::mat4(1.0f);
496 color = glm::vec3(1.0f, 0.0f, 0.0f);
497 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
498 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
499 model = glm::translate(model, glm::vec3(2.2f, -2.1f, -4.25f));
500 model = glm::rotate(model, glm::radians(-13.0f), glm::vec3(1.0f, 0.0f, 0.0f));
501 model = glm::scale(model, glm::vec3(2.0f, 2.0f, 0.3f));
502 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
503 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
504 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
505 meshList[1] -> RenderMesh();

```

Imagen 2.8

- Luego dibujamos 9 pirámides triangulares de color verde, fue una de las caras más difíciles de hacer y tuve que trabajar con rotación, imagen 3.0.

```

507 // Triangulo 1 cara (verde)
508 model = glm::mat4(1.0f);
509 color = glm::vec3(0.0f, 1.0f, 0.0f);
510 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
511 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
512 model = glm::translate(model, glm::vec3(0.45f, 2.1f, -5.84f));
513 model = glm::rotate(model, glm::radians(14.0f), glm::vec3(1.0f, 0.0f, 0.0f)); // orientacion de inclinacion
514 model = glm::rotate(model, glm::radians(135.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
515 model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(0.0f, 0.0f, 1.0f)); // orientacion de lado
516 model = glm::scale(model, glm::vec3(1.4f, 2.0f, 0.3f));
517 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
518 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
519 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
520 meshList[1]->RenderMesh();
521

```

Imagen 3.0

4. Dibujamos 9 pirámides triangulares color azul, trabajar en esta cara fue difícil y ocupe rotación, imagen 4.0.

```

642 // Triangulo 1 cara (azul)
643 model = glm::mat4(1.0f);
644 color = glm::vec3(0.0f, 0.0f, 1.0f);
645 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
646 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
647 model = glm::translate(model, glm::vec3(-0.32f, 2.1f, -5.84f));
648 model = glm::rotate(model, glm::radians(16.0f), glm::vec3(1.0f, 0.0f, 0.0f)); // orientacion de inclinacion
649 model = glm::rotate(model, glm::radians(40.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
650 model = glm::rotate(model, glm::radians(-18.0f), glm::vec3(0.0f, 0.0f, 1.0f)); // orientacion de lado
651 model = glm::scale(model, glm::vec3(1.4f, 2.0f, 0.3f));
652 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
653 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
654 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
655 meshList[1]->RenderMesh();

```

Imagen 4.0

5. Dibujamos 9 pirámides triangulares color celeste, en donde su dificultad fue muy sencilla, esta fue la segunda cara que más rápido me salió, imagen 5.0.

```

778 // Triangulo 1 cara (celeste)
779 model = glm::mat4(1.0f);
780 color = glm::vec3(0.0f, 1.0f, 1.0f);
781 //Opcional duplicar esta traslación inicial para posicionar en -Z a los objetos en el mismo punto
782 model = glm::translate(model, glm::vec3(0.0f, 0.0f, -3.0f));
783 model = glm::translate(model, glm::vec3(0.0f, -3.3f, -4.0));
784 model = glm::rotate(model, glm::radians(0.0f), glm::vec3(0.0f, 1.0f, 0.0f)); // orientacion de profundidad
785 model = glm::scale(model, glm::vec3(1.4f, 0.3f, 2.0f));
786 glUniformMatrix4fv(uniformModel, 1, GL_FALSE, glm::value_ptr(model)); //FALSE ES PARA QUE NO SEA TRANSPUESTA
787 glUniformMatrix4fv(uniformView, 1, GL_FALSE, glm::value_ptr(camera.calculateViewMatrix()));
788 glUniform3fv(uniformColor, 1, glm::value_ptr(color));
789 meshList[1]->RenderMesh();
790

```

Imagen 5.0

6. Resultado final, no es el esperado, pero trate de que estuviera lo más presentable, imagen 6.0, 6.1, 6.2 y 6.3.

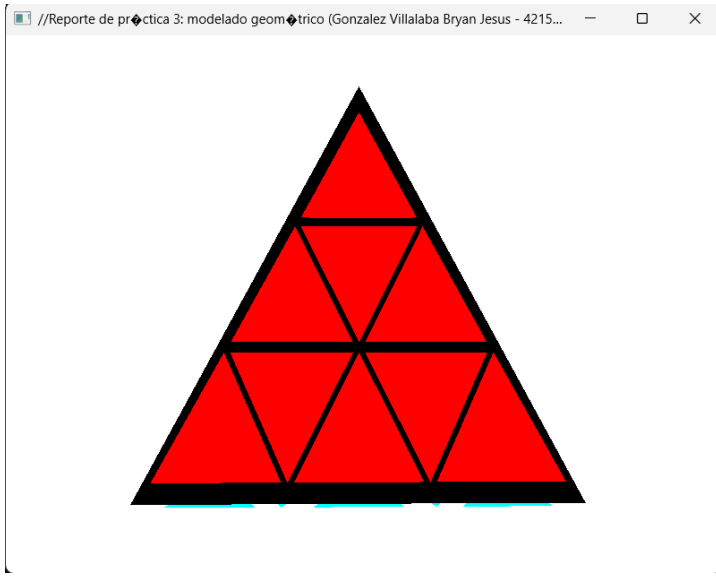


Imagen 6.0

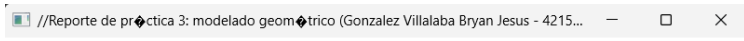


Imagen 6.1



Imagen 6.2

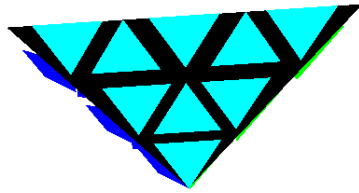


Imagen 6.3

Conclusión:

Fue una práctica muy difícil sobre todo en las caras de color verde y color azul, ya que a mi parecer la pirámide no media lo mismo y eso me dificulto mucho el hecho de realizar bien la rotación de las piezas, en el dibujo de las caras color rojo y celeste me lleve menos de 20 minutos por cada cara, ya que fueron sencillas por la forma en la que estaban, pero en las caras que se me dificultaron me rendir, me lleve más de 6 horas tratando de dibujar las caras bien, pero no me quedaron como esperaba, el problema es que tengo que modificar el eje Z y Y de forma que el triángulo quede alineado a la pirámide, y el hecho de que es color negro me dificulto mas el trabajo, no pensé en cambiar el color, hasta ahorita que estoy haciendo el reporte, la parte más compleja de todo son las rotaciones y la escala de los lados que no veo de manera uniforme.