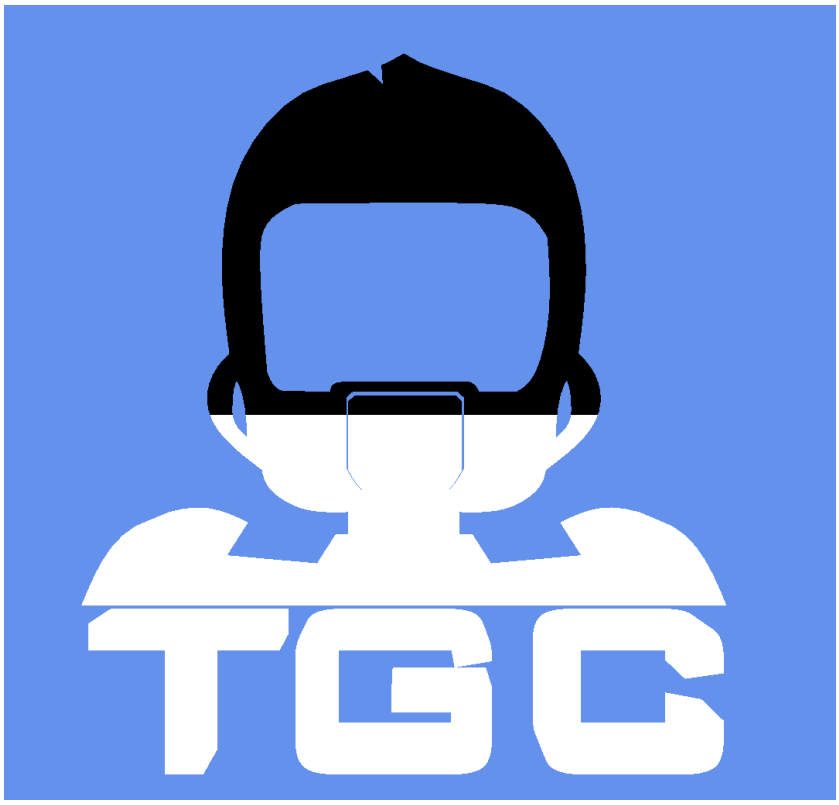
Ejercicios de Shaders Simples

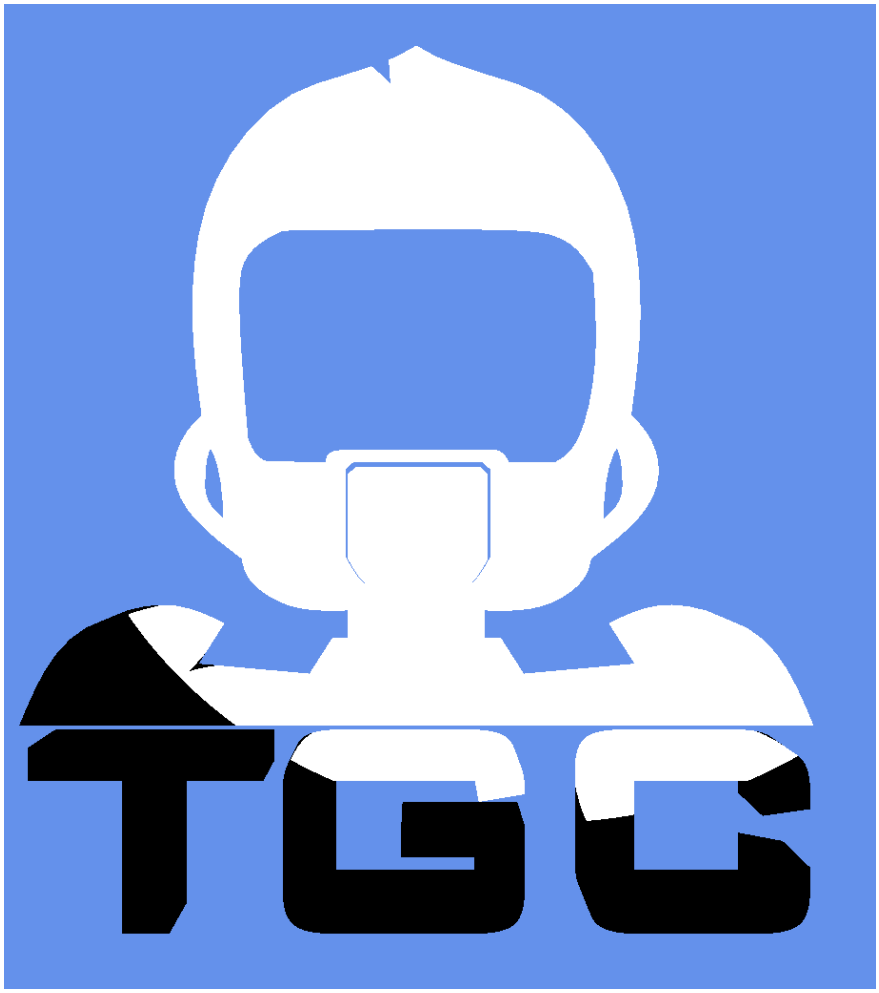
Ejercicio 1

Escribir un fragment shader que dibuje blanco si la posición en el eje Y del espacio de modelo es menor o igual que un valor arbitrario, o negro en el caso contrario.
Bonus: Lograrlo sin usar if ni el operador condicional ternario (?:).



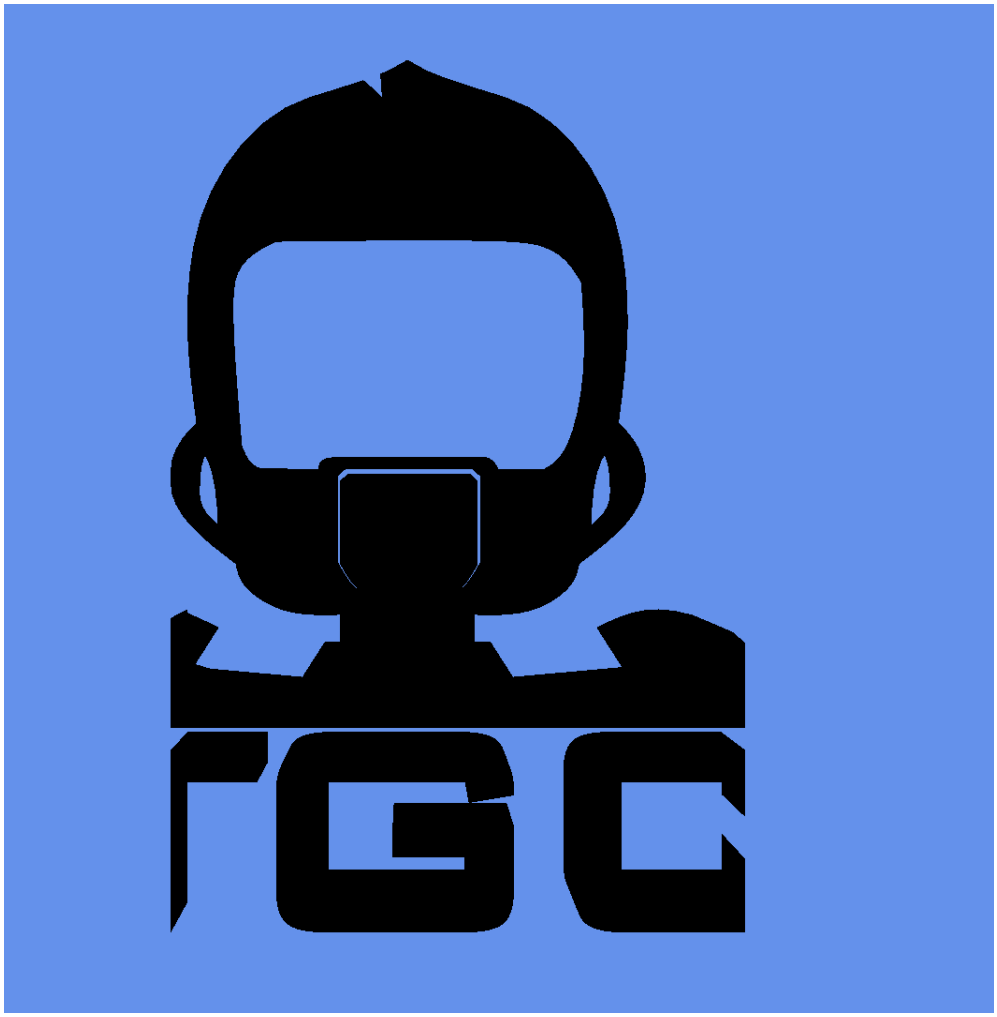
Ejercicio 2

Escribir un fragment shader que dibuje blanco si la posición en espacio de mundo está dentro de una esfera de radio 40 unidades centrada en el punto (10.0, 10.0, 10.0), o negro en el caso contrario.
Bonus: Lograrlo sin usar if ni el operador condicional ternario (?:).



Ejercicio 3

Escribir un vertex shader que haga que los vértices en el eje X se compriman entre los valores [Min, Max] en espacio local siendo Min y Max dos variables float uniformes que se pasan desde la aplicación.



Ejercicio 4

Escribir un fragment shader que tome una textura (cualquiera) y descarte los fragmentos cuyo color de textura tengan valores menores que 0.4 en el canal rojo. En el caso contrario, muestra el color de la textura.

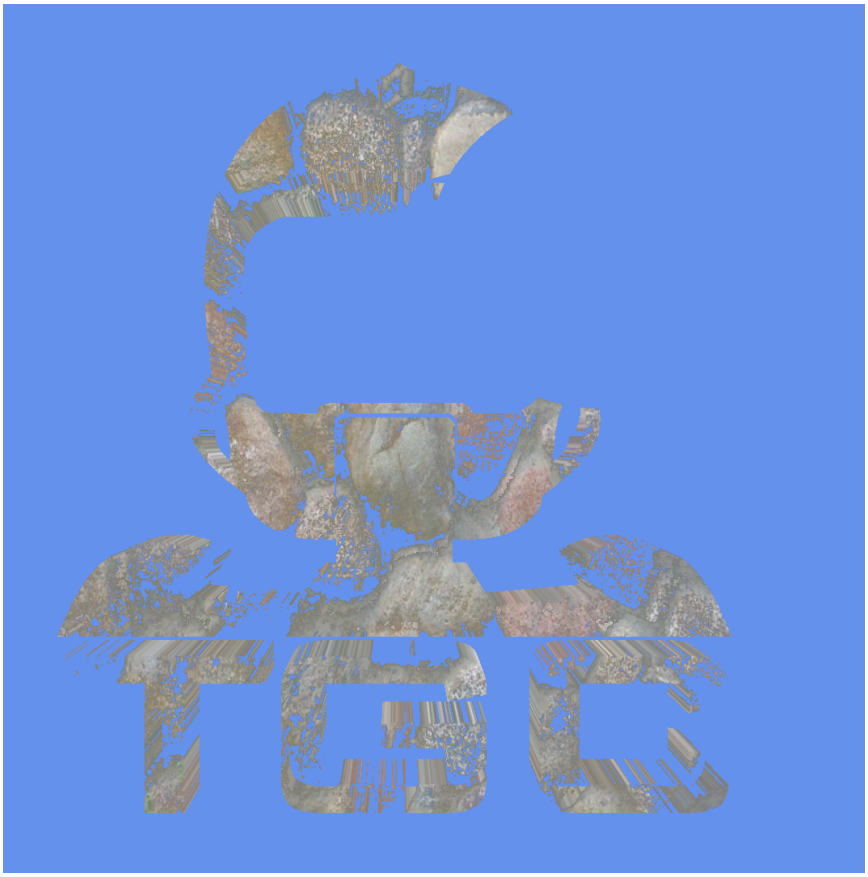
Ejercicio 5

Escribir un fragment shader que tome una textura (cualquiera) y descarte los fragmentos que cumplan al menos una de estas condiciones:

- El color de textura tiene un valor menor que 0.5 en el canal verde.
- El color de textura tiene un valor menor que 0.1 en el canal azul.
- Los fragmentos se encuentran dentro de una esfera de radio 20 con centro en el punto (20.0, 20.0, 0.0) en espacio de mundo.

En el caso contrario, muestra el color de la textura.

Bonus: Lograrlo usando la función `clip`, que descarta los fragmentos si el valor especificado es menor que cero, y sin usar if ni valores booleanos ni el operador condicional ternario (?:).



Ejercicio 6

Escribir un fragment shader que tome una textura (cualquiera) y haga "scrolling" en el eje Y de las coordenadas de textura, sumando el tiempo que crece infinitamente.

Bonus: Razonar y determinar con cuales Addressing Modes este efecto funciona y con cuales Addressing Modes no.

Ejercicio 7

Escribir un fragment shader que tome un `float4` como variable uniforme y lo interprete como un plano (Normal en xyz y Desplazamiento en w) para descartar todos los fragmentos que estén de un lado de este plano, y dibujar todos los que estén del otro lado de color rojo.