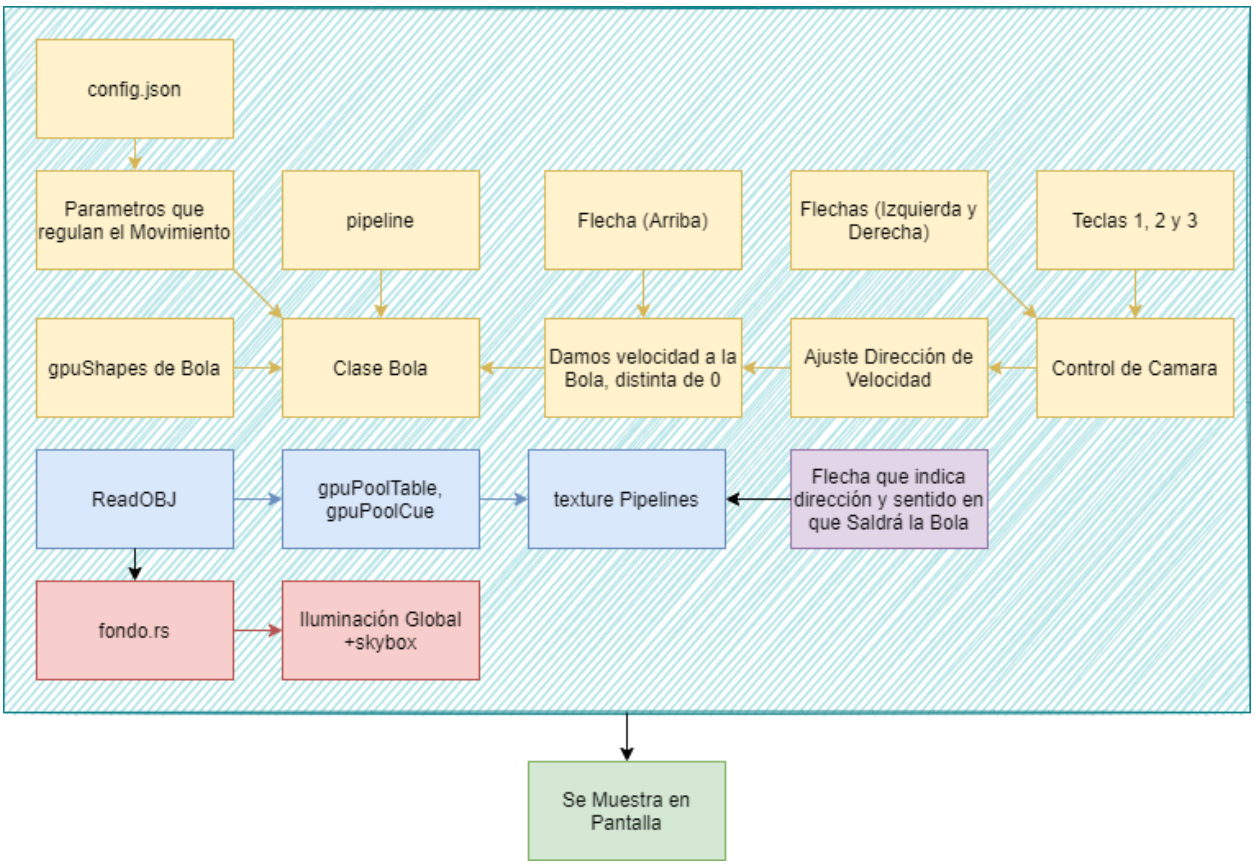
Solución Propuesta

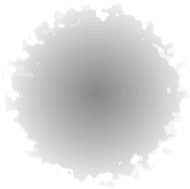
Se quiere Realizar un Juego de Pool utilizando OpenGL 3D, donde tengamos distintas vistas que nos permitan visualizar la mesa junto a las pelotas, es decir, el juego en general implementando colisiones y física en este.

Para realizar lo anterior, se comienza creando una clase para las Bolas de billar, las cuales serán de color Blanco, Negro, Rojo y Amarillo, en donde nosotros debemos "pegarle" a la bola blanca. La clase Bola o "Ball" nos permite a través de la función "action" desplazar las bolas, para simular el golpe utilizamos una velocidad inicial de magnitud constante, el control de su dirección y sentido puede observarse en el siguiente diagrama resumen.

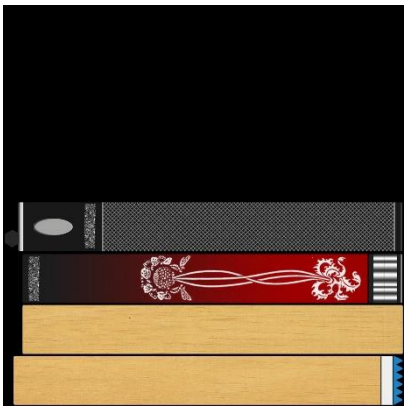


Las bolas, una vez obteniendo su dirección y sentido a partir de los controles de cámara, se ven afectados por coeficientes de Fricción y Restitución por el movimiento, sin embargo, dada la complejidad y cantidad de errores generados con el uso del coeficiente de Restitución se prefirió no usarlo, no obstante existe un roce cinético dado por el coeficiente de Fricción cuando las Bolas se mueven, lo anterior puede ser configurado en el archivo *config.json* previo a iniciar el programa con el comando *python pool_party.py config.json* .

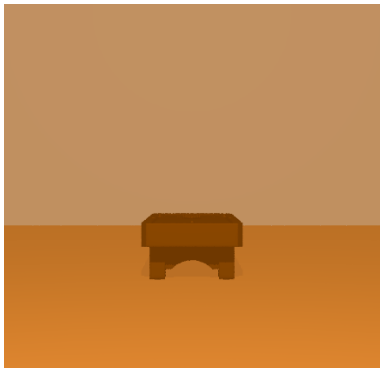
Utilizamos la función *draw()* definida en la clase Bola para dibujar cada una de estas, las cuales son modelos geométricos de Bolas con valores RGB, además, también para facilitar la implementación de sombras dentro de la misma clase definimos la función *draw_shadow*(), la cual a través de una Textura de Sombras nos permite ponerlas en escena para cada bola como la que se muestra a continuación.



Para dibujar la mesa y el taco, hacemos uso de Blender para modificar OBJS y texturas, luego a partir del archivo *readOBJ.py* definido, podemos implementar objs con texturas, las cuales iluminamos a través del método de iluminación de Phong para la iluminación local. Lo anterior será puesto en pantalla solo cuando no haya movimiento de bolas. Las texturas utilizadas son las siguientes:



Para la iluminación Global, se utilizó Rust colocando objs de mesas de pool para simular un local de Pool, colocando una luz central que ilumine a las mesas adyacentes y seleccionando colores en la gama de los cafés para mantener estética, generando las siguientes imágenes:



Las colisiones son implementadas en el archivo *collisions.py* junto a la clase Bola, donde cada Bola tiene su propia hitbox, una Fricción común, definimos además una “hitbox” para los hoyos de la mesa de pool, donde al estar las bolas cerca de sus posiciones estas dejan de interactuar con el resto.

Instrucciones de Ejecución

Para ejecutar el Programa se utiliza el commando *python pool_party.py config.json*, luego el usuario puede utilizar las siguientes teclas para controlar nuestro programa:

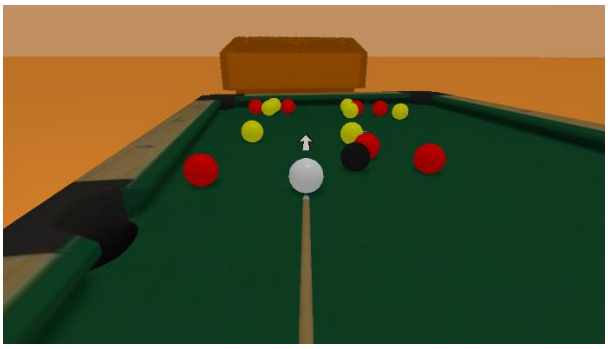
- Teclas 1, 2 y 3 cambian la visualización de la cámara, para que esta sea aérea, alrededor de la mesa o detrás de la bola respectivamente.
- Teclas ← y → nos permiten ajustar el angulo de la cámara horizontalmente, junto con el control de la dirección y sentido en que saldrá disparada la bola.
- Tecla ↑ para lanzar.

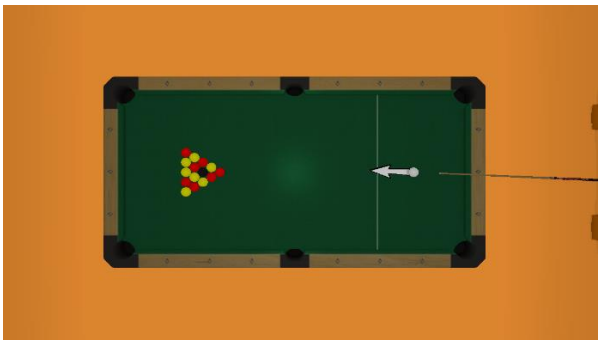
Para hacer uso de nuestro pool party hacen falta las librerías de OpenGL, sys, numpy, repositorio del curso y los archivos adjuntos en nuestro repositorio.

Resultados

Se obtiene como resultado una mesa de pool funcional con estética de un Pool Bar, simulando el juego de “Black Ball”, que sin embargo presenta deficiencia al momento de simular el movimiento de bolas debido a la física del problema.

A continuación, podemos ver las 3 distintas vistas de la mesa de Pool:





Autoevaluación

Criterio-Puntaje	0	1	2	3
OpenGL			x	
Shaders		x		
Modelos geométricos			x	
Transformaciones				x
Texturas				x
Modelación jerárquica		x		
Curvas	x			
Funcionalidades mecánicas o lógica de juego				x
Entradas o control de usuario				x
Visualización de estado del programa		x		