Universidad de Chile
FCFM - DCC
CC3501

Reporte Tarea 3

Sebastián Tapia
26 de julio de 2021

Solución Propuesta

La solución consta de 3 partes: Fisica y colisiones, clases controladoras y shaders.

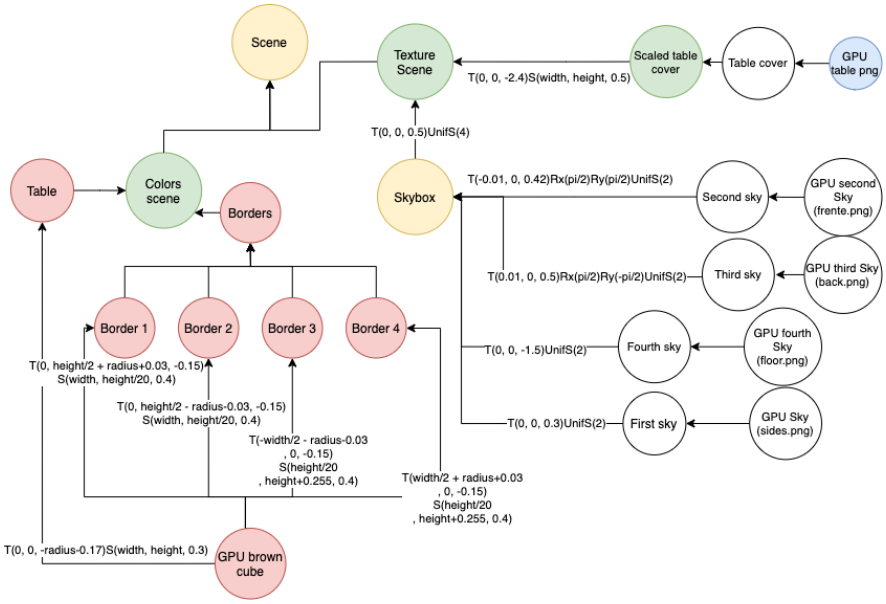
Fisica y colisiones

Las colisiones entre bolas son inspiradas del ejemplo *ex_collisions*, donde se implementan colisiones inelásticas entre las esferas, con un factor de restitución $C = 0{,}9999$.

Por otro lado, la física correspondía a agregar roce ($mu = 0{,}5$), donde se modificaron las velocidades en cada eje. Como estas magnitudes eran vectoriales, se calcularon de distintas formas según el signo del vector, donde además se utilizaron distintos métodos numéricos (todos los vistos en clases) para aproximar las velocidades. Por simplicidad, se omite la rotación de las esferas en la mesa.

El golpe a la bola blanca se efectúa asignando una velocidad inicial con misma dirección que el vector *forward* de la cámara. La intensidad del golpe es siempre la misma. Las colisiones de las bolas con los agujeros de la mesa se modelan usando colisión entre circunferencias y provocando que la bola en cuestión desaparezca de la escena.

Para modelar el fondo se utilizó la librería *rpt* con *raytracing*, generando imágenes para un skybox que simule una sala con varias mesas y sillas. Por último la mesa y el skybox fueron modelados utilizando el siguiente grafo de escena:



Reporte Tarea 3

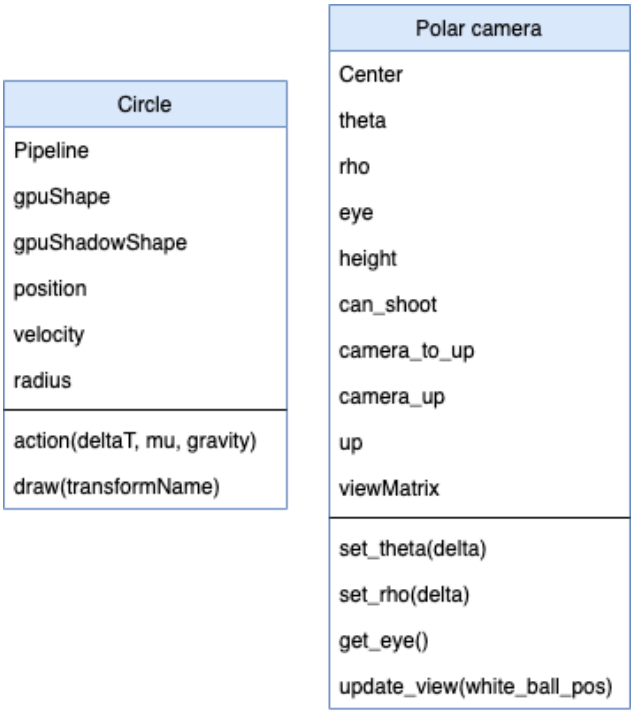
Sebastián Tapia

Clases controladoras

Para los movimientos se definieron clases controladoras tanto para las bolas como para el movimiento de la cámara.

En el caso de las esferas, se almacenan las velocidades y posiciones de cada una, donde internamente son aplicadas las ecuaciones físicas y de colisiones dichas en la sección anterior.

En tanto la cámara almacena su propia posición, vector *at* y vector *up*, siguiendo una trayectoria circular alrededor de la bola blanca y con la posibilidad de alternar a una vista desde arriba (sin posibilidad de desplazarse). Las clases se resumen en el siguiente diagrama UML:



Shaders

Se implementan y utilizan 3 shaders para dibujar con colores y texturas e implementar un efecto de *Heat map* según una de las luces de la escena.
Todos los shaders utilizan 2 luces ubicadas en los bordes de la mesa y que son de tipo spotlight, cuyos parámetros (posiciones, direcciones del spot y colores) son dados con *uniforms*.

El Heat map se logró introduciendo condiciones según la distancia en que se encuentra un punto de una de las fuentes de luz, asignando los colores rojo (más cercano), amarillo, verde,

cyan y azul (más lejano). Este efecto se activa con la tecla *tab* y se utiliza sobre cuerpos con textura.

Instrucciones de Ejecución

El programa se inicia al ejecutar el siguiente comando en la ventana de comandos:

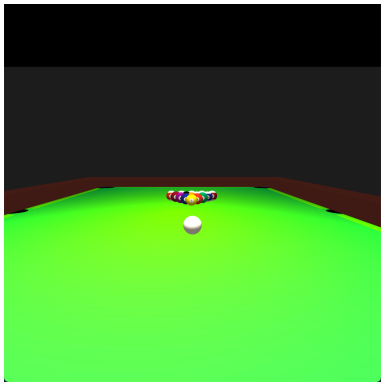
```
python pool_party.py config.json
```

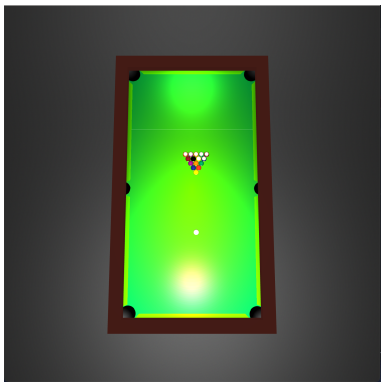
Donde el archivo *config.json* contiene los parámetros para el coeficiente de restitución y roce sobre la mesa.

Con las teclas *left* y *right* se controla la posición de la cámara alrededor de la bola blanca, con *tab* se cambia de iluminación de Phong al Heat Map. Con la tecla 1 se activa la vista desde arriba y con la tecla enter se dispara la bola blanca en la dirección donde se esté viendo. Esta última tecla queda desactivada hasta que todas las bolas queden nuevamente quietas.

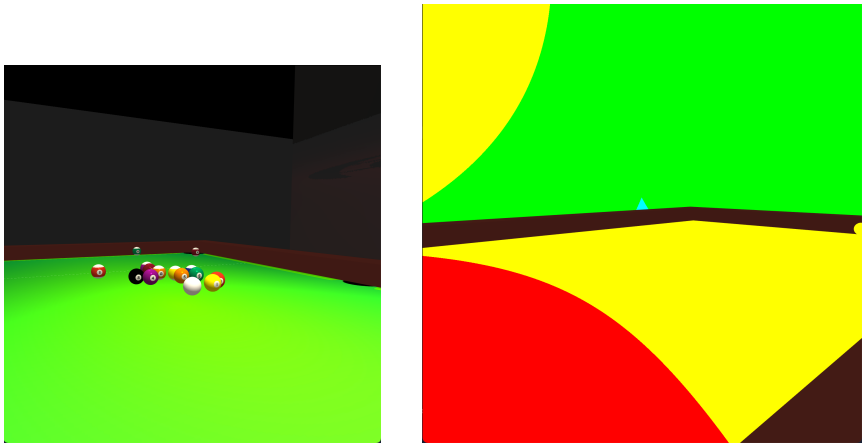
Resultados

Se obtienen los siguientes screenshots con características del programa:





Distintas vistas de la cámara.



Diferentes shaders: Izquierda Phong y derecha mapa de calor. Se puede notar la implementación de luz tipo spotlight en el mapa de calor.

Autoevaluación

Criterio-Puntaje	0	1	2	3
Modelación jerárquica				x
Vistas y proyecciones				x
Iluminación local				x
Iluminación global			x	
Visualización científica		x		
Diferencias finitas				x
Colisiones y física			x	
Funcionalidades mecánicas o lógica de juego			x	
Entradas o Control de usuario*			x	
Visualización de estado del programa			x	