

Solución Propuesta

La solución consta de 3 partes; Grafo de movimiento, clases controladoras y shaders.

Grafo de movimiento

Se utiliza Blender para modelar las parte móviles del articulado, modificando la función de lectura de *OBJ* para admitir coords. de textura. Para el movimiento se utilizó un grafo de escena con nodos que reciben sus movimientos. Destacar que la cabeza se dibuja con un pipeline distinto (no usa texturas). El siguiente esquema presenta al grafo utilizado:

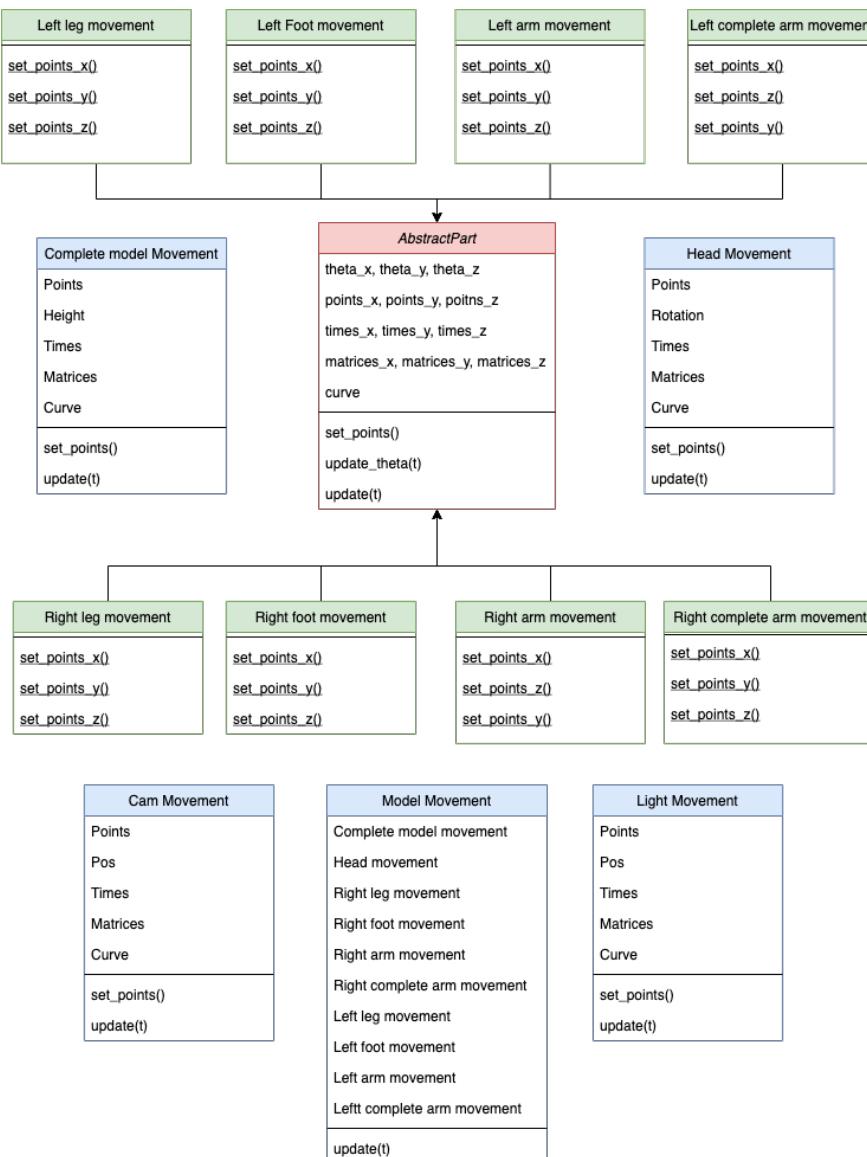


En azul las GPU's bases, en rojo a las piezas del cuerpo, verde a nodos con transformaciones y en amarillo nodos con el modelo completo o casi completo.

En total se realizan 10 transformaciones en tiempo de ejecución. Mencionar que se modeló un *skybox* con fondo de estadio y cancha de fútbol.

Clases controladoras y curvas

Para los movimientos se definieron clases controladoras, donde se almacenan puntos (ángulos de rotación o traslaciones) e internamente se generan curvas de Catmull–Rom para cada t que ingresa. Estas clases se utilizan para el modelo 3D, movimientos de la cámara y luces de fondo.



Las clases correspondientes a piezas móviles del modelo son enlazadas a una clase abstracta que engloba varias características y, por lo tanto, cada clase heredada solo se preocupa de *setear* sus propios puntos de la curva y actualizarlos.

La clase *Complete model movement* contiene todos los movimientos articulados y es invocada en la aplicación para luego extraer movimientos. También se llaman a las clases *Cam movement* y *Light movement* para la cámara y luces, respectivamente.

Por último, se definen nuevas funciones en el script *ex_curves* que evalúan puntos y tiempos (de tamaños arbitrarios) para generar curvas de Catmull–Rom, donde usualmente el eje x de las curvas representa el tiempo de los movimientos y el eje y ángulos de rotación o traslaciones (el eje z sólo es utilizado en la curva de la cámara, donde se especifican puntos alrededor del modelo bailarín). Se utilizan sólo estos tipos de curvas en todo el programa.

Shaders

Se implementan y utilizan 4 shaders para dibujar con colores y texturas e implementar el efecto *cel shading*.

Todos los shaders utilizan 4 luces ubicadas en las esquinas del skybox y que son de tipo spotlight, cuyos parámetros (posiciones, direcciones del spot y colores) son dados con *uniforms*. Las luces, como se mencionó antes, se mueven en horizontal o vertical, siguiendo curvas de Catmull–Rom.

El efecto de *cel shading* se logró introduciendo una constante *levels* que determina el nivel de discretización a utilizar (en este caso, 3 niveles). Con la función *floor* nativa se discretizan las componentes difusas y especular de las luces. Este efecto se utiliza tanto en shaders con colores, como en shaders de textura.

Por último, mencionar que el efecto de *slow motion* se implementó limitando los fps en que la aplicación dibuja, dándole un tope de 15 fps.

Instrucciones de Ejecución

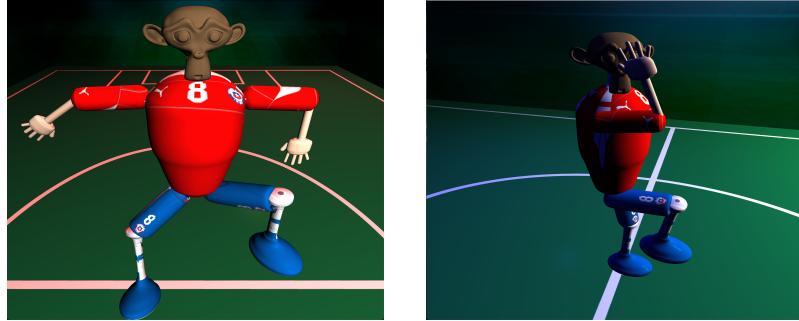
Se debe tener instalado el paquete *abstract method* y *ABC*. El programa se inicia al ejecutar el siguiente comando en la ventana de comandos:

```
python dance_celshading.py
```

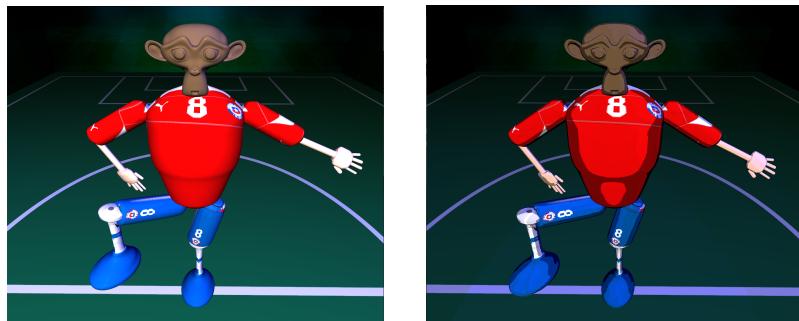
Con las teclas *left* y *right* se controla la posición de la cámara, con *tab* se cambia de iluminación de Phong a cel shading. Con la tecla 1 se produce el efecto *slow motion* y con 2 la cámara se mueve automáticamente alrededor del modelo articulado.

Resultados

Se obtienen los siguientes screenshots con características del programa:



Modelo 3D en distintas posiciones en su baile.



Diferentes shaders: Izquierda Phong y derecha cel shading.



Cel shading en las texturas de fondo y piso. Se puede notar la implementación de luz tipo spotlight.

Autoevaluación

Criterio-Puntaje	0	1	2	3
Evaluación general				x
OpenGL			x	
Shaders*				x
Modelos geométricos			x	
Transformaciones			x	
Texturas*				x
Modelación jerárquica*				x
Curvas*				x
Vistas y proyecciones			x	
Iluminación local*				x
Mallas de polígonos			x	
Funcionalidades mecánicas o lógica de juego				x
Entradas o Control de usuario*				x
Visualización de estado del programa			x	