Computación Gráfica 2019 - UNL-FICH

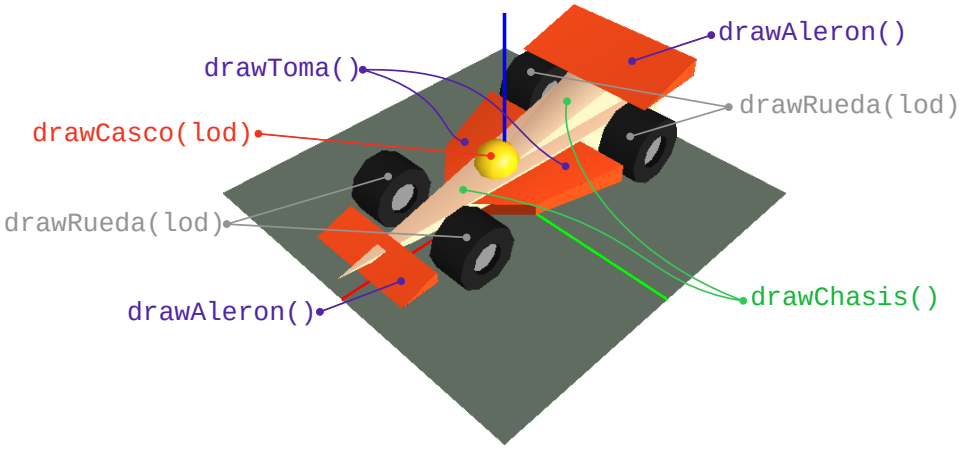
Práctica de Transformaciones: Conduzca su primer F1!

Última revisión: 26/08/2019

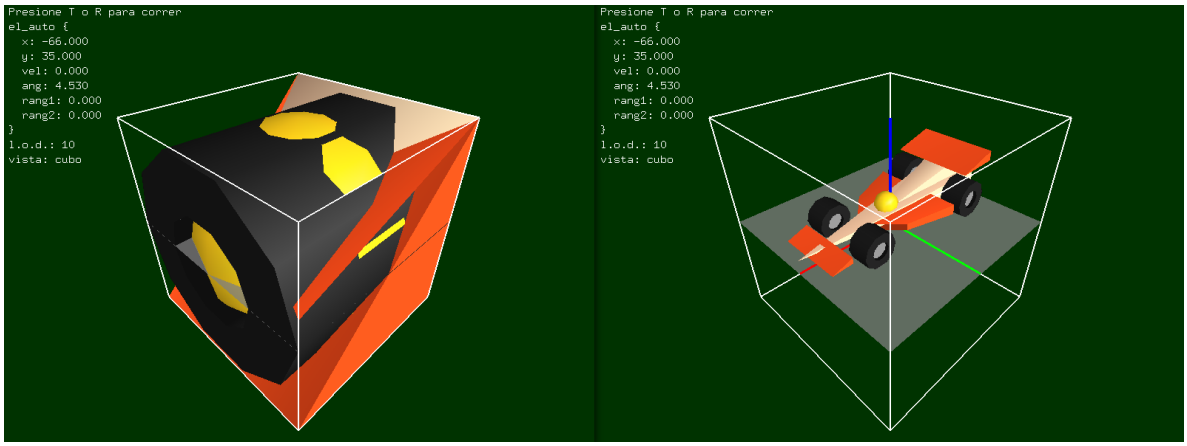
En este práctico se pretende dibujar y animar el movimiento de un auto de formula 1. El código base entregado tiene implementadas las funciones que dibujan cada una de las partes que lo componen, y el cálculo del movimiento del mismo. Aplicando lo aprendido en teoría, el alumno debe acomodar las piezas adecuadamente para conformar el modelo, ubicarlo en el mundo y seguirlo con la cámara mientras avanza.

Primera parte: Ensamblado del Modelo

Ya están implementadas las funciones que dibujan cada una de las partes que componen el auto.



Cada función dibuja una pieza ocupando todo el cubo `[-1;+1]x[-1;+1]x[-1;+1]` (dibujado en blanco, para usar de referencia), por lo que debe aplicar antes de cada llamada las transformaciones que correspondan para que el resultado tenga la posición, orientación y tamaño adecuado.



La imagen muestra el estado inicial, y el resultado final al que deberá llegar. Nótese que en la vista del cubo, los ejes x, y, z se representan con segmentos rojo, verde y azul respectivamente.

En el código, para aplicar las transformaciones deberá generar la matrices y aplicarlas con la función `glMultMatrix` [1].

$m = \begin{bmatrix} e_x^x & e_y^x & e_z^x & o_x \\ e_x^y & e_y^y & e_z^y & o_y \\ e_x^z & e_y^z & e_z^z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$	`float m[] = { ex_x, ex_y, ex_z, 0,`
	`ey_x, ey_y, ey_z, 0,`
	`ez_x, ez_y, ez_z, 0,`
	`o_x, o_y, o_z, 1 };`
	`glMultMatrixf(m);`
	`dibujarAlgo();`

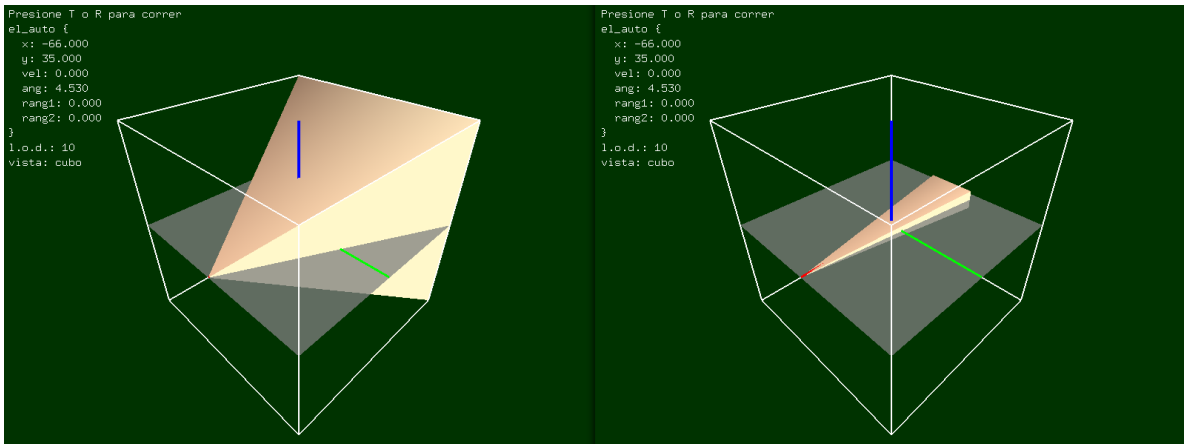
Matriz de transformación	*Aplicación con `glMultMatrix`*

Notar que la matriz, en el código, se escribe transpuesta respecto a cómo la escribimos habitualmente en teoría.

Ayuda: utilice además las funciones `glPushMatrix()` y `glPopMatrix()` para guardar y restaurar el estado de las matrices; es decir, para deshacer una transformación luego de aplicarla a una determinada pieza.

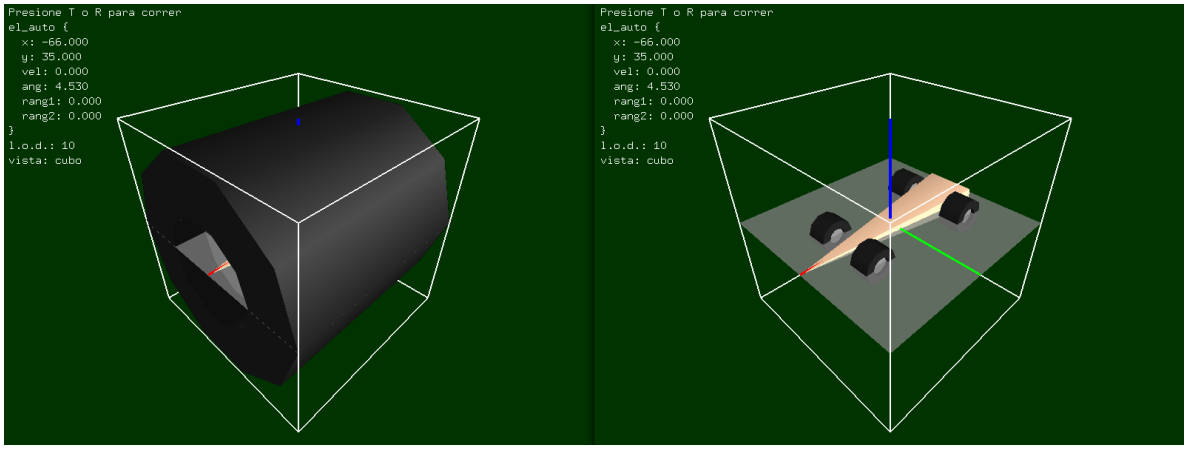
Paso 1: Dibujar base del chasis

Utilice la función `drawChasis`. Aplique una transformación antes de la llamada a la función para escalar esta pirámide. El auto debe formarse sobre el plano xy, mirando hacia +x. La nueva pirámide debe medir 1.9 unidades de largo (x), 0.5 unidades de ancho (y) y 0.2 unidades de profundidad (z) y permanecer centrada en (0;0;0).



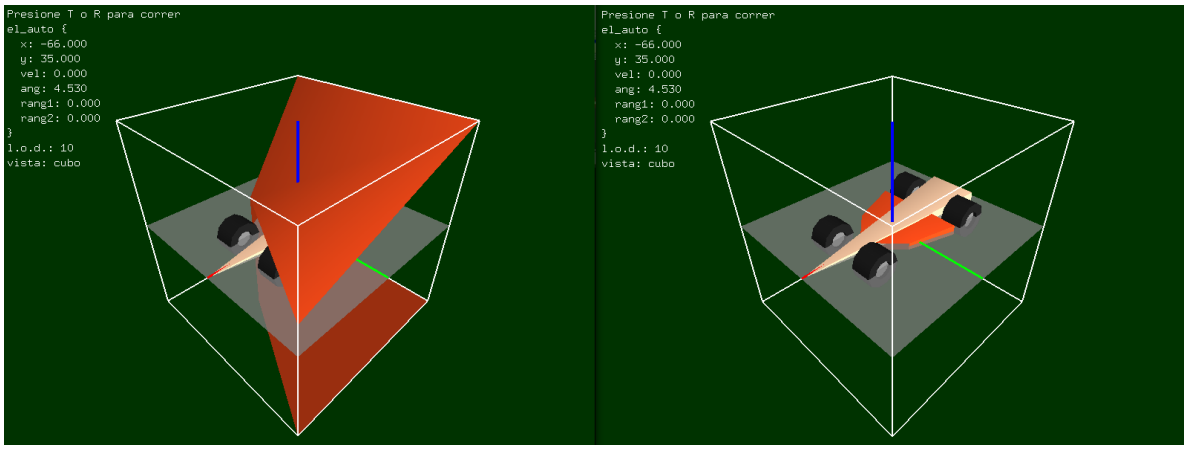
Paso 2: Dibujar las ruedas

Utilice la función `drawRueda`. Deberá escalarlas y rotarlas [2] adecuadamente. Las coordenadas de los centros de las ruedas, relativas a la ubicación del auto son: (+0.48;+0.28;0), (+0.48;-0.28;0), (-0.6;+0.35,0) y (-0.6;-0.35;0). Las ruedas traseras deben medir 0.22 x 0.36 x 0.36 las delanteras tienen el 90% del tamaño de las trasera



Paso 3: Dibujar las tomas de aire laterales

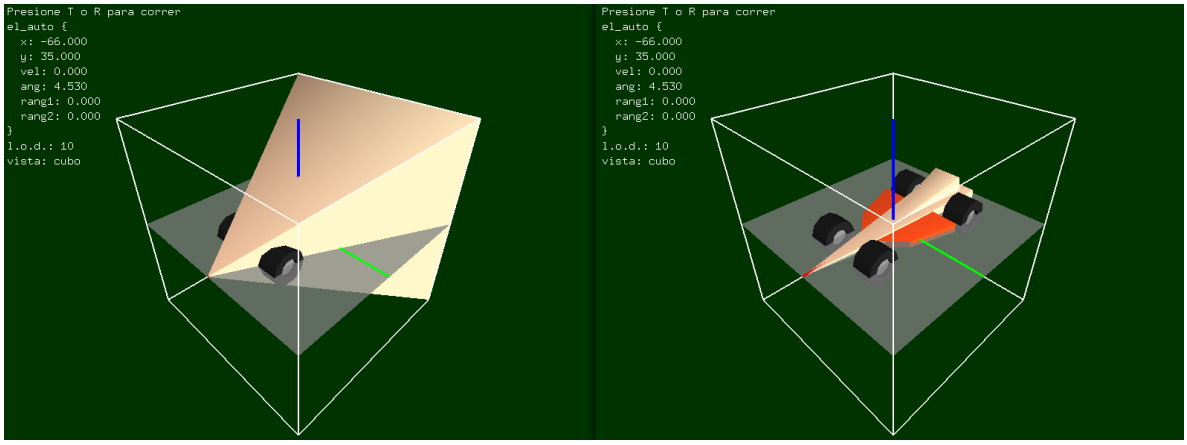
Utilice la función `drawToma`. Las tomas deben escalarse para medir 0.7 x 0.4 x 0.1. Para ubicarlas correctamente, debe desplazarlas 0.25 hacia un lateral del chasis (abajo en el dibujo, eje y) y 0.01 hacia arriba (eje z).



Debe dibujar las dos tomas con la misma función, aplicando una transformación de espejado `3` en una de ellas.

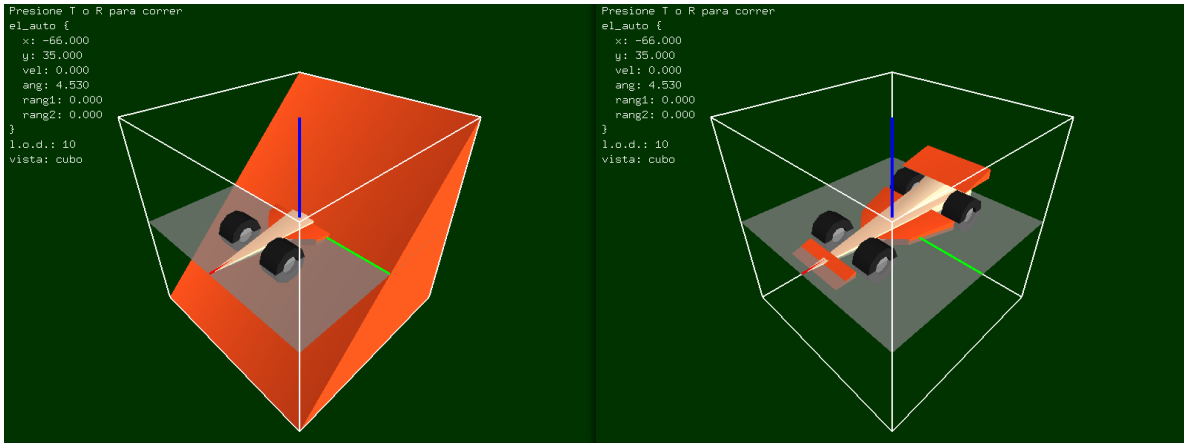
Paso 4: Dibujar la parte superior del chasis

Utilice nuevamente `drawChasis` para dibujar la parte superior del mismo (la que se ubica detrás del casco, y va hasta la base del alerón trasero). El tamaño final de la pieza debe ser 1.2 x .3 x .2, debe desplazarse 0.2 hacia atrás y 0.05 hacia arriba, y rotarse 10 grados para quedar como en la figura.



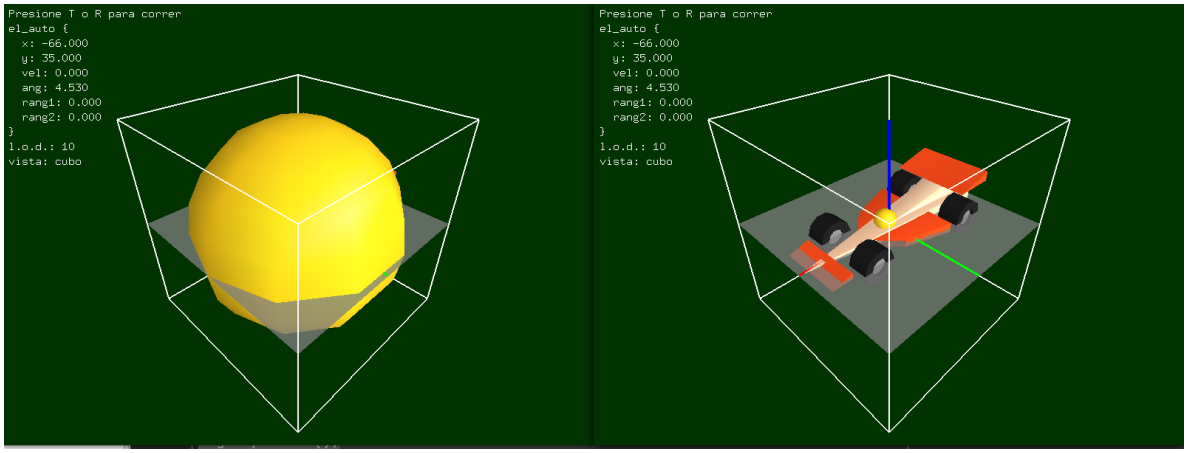
Paso 5: Dibujar alerones

Utilice dos llamadas a la función `drawAleron` para dibujarlos. El delantero mide 0.2;0.6;0.06 y va centrado en el punto 0.8;0;0, mientras que el trasero mide 0.4;0.8;0.1, y va centrado en -0.75;0;0.3



Paso 6: Dibujar el casco del piloto

Intente escalar y posicionar la esfera que simula el casco del piloto determinando usted un tamaño y una posición adecuados.

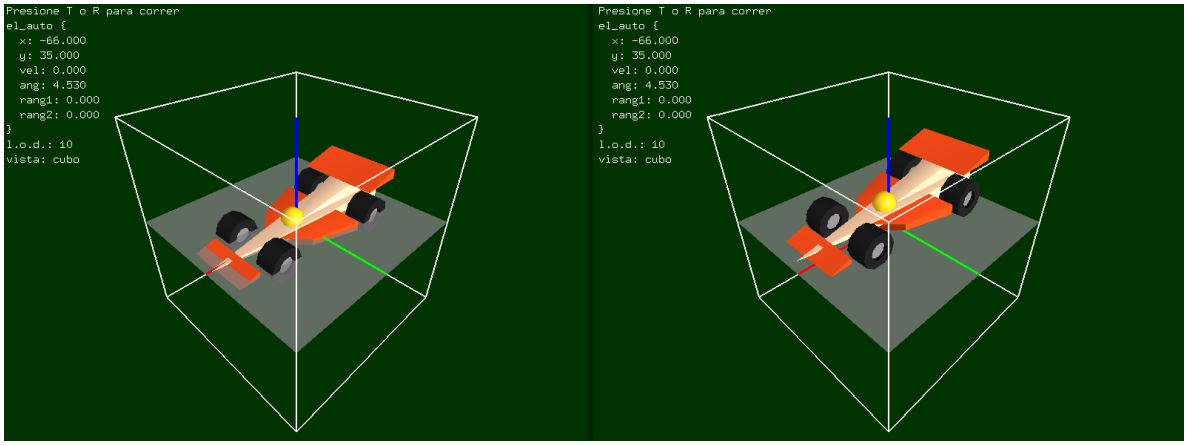


Segunda parte: Movimiento

El programa define una variable global denominada `el_auto`, que es en realidad un struct que contiene en sus atributos los datos respecto a la posición, orientación, etc. del auto. Puede ver su definición en el archivo `Auto.hpp`. A partir de ahora, cuando se diga "la variable", debe entenderse como "la variable *miembro* del struct".

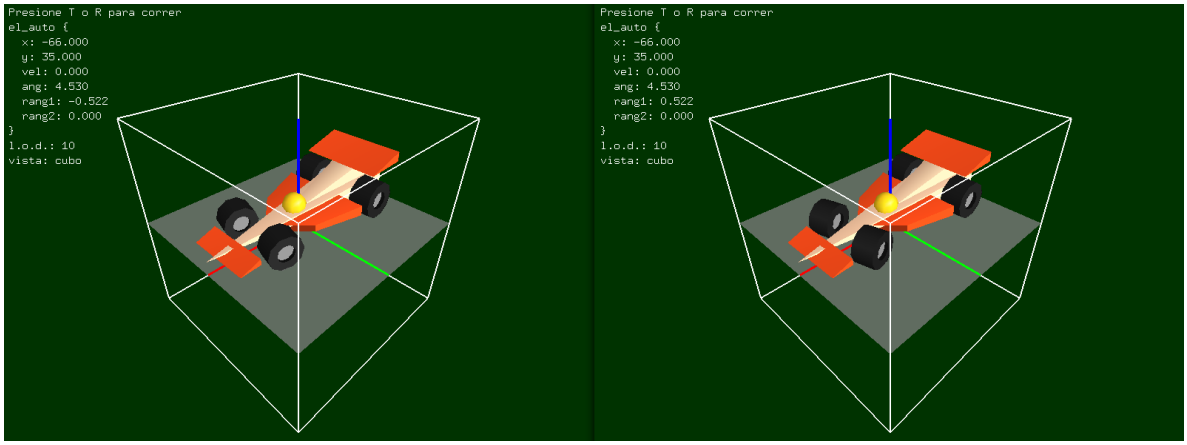
Paso 1: Acomodar el auto sobre la pista

Primero mueva y rote levemente el auto de forma que sus ruedas queden correctamente “apoyadas“ sobre la superficie que representa el suelo.



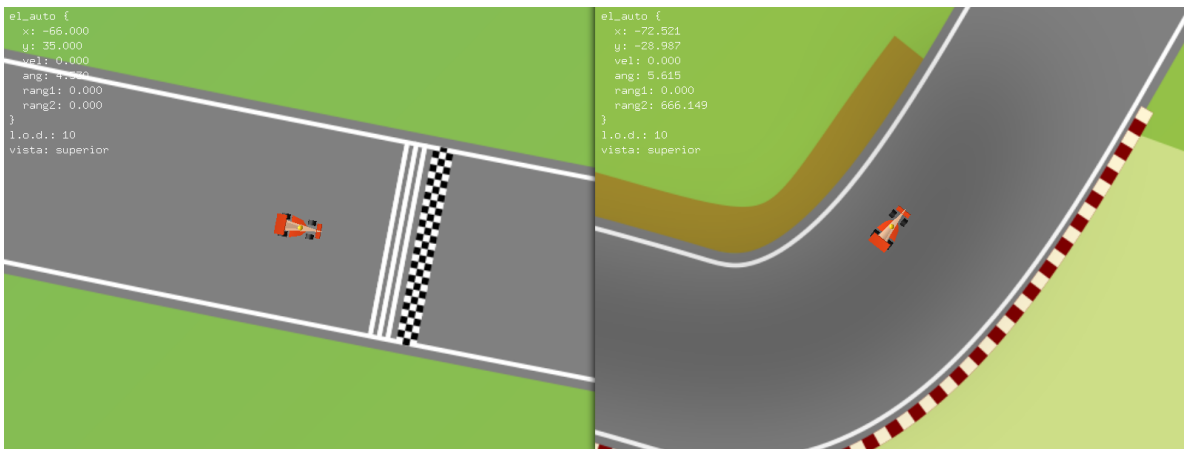
Paso 2: Dirección y ruedas delanteras

La variable `rang1` tiene el angulo de las ruedas respecto a la posición original en el auto (como giran cuando gira el volante). La variable `rang2` tiene el angulo de giro de la rueda sobre su propio eje (como "rueda" la rueda cuando el auto avanza). Agregue/modifique las transformaciones que corresponda para poder mover las ruedas cuando varían dichas variables.



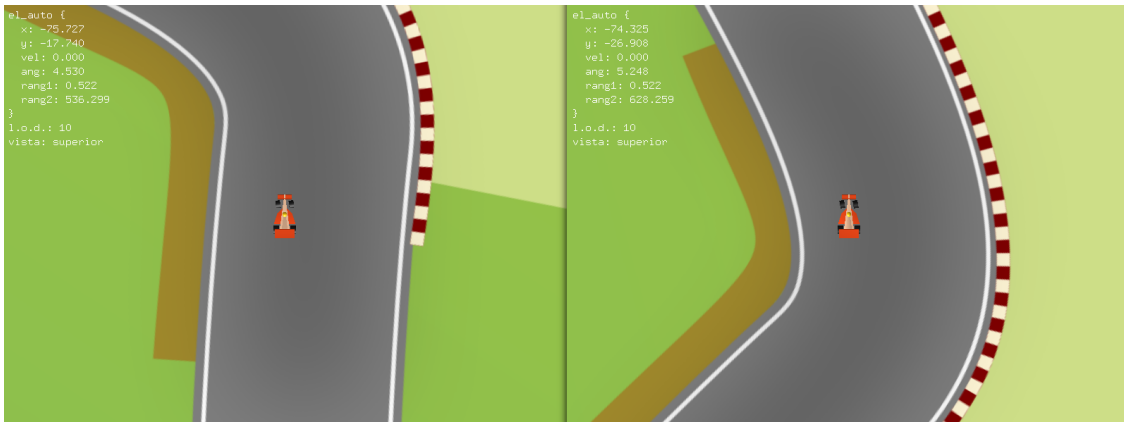
Paso 3: Movimiento

El cálculo del movimiento del auto ya está implementado. Las variables `x`, e `y` tienen las coordenadas de la posición del auto en la pista, y la variable `ang` el ángulo hacia donde se dirige el vehículo. Modifique el código para que cuando la variable animado sea true, se apliquen las transformaciones adecuadas de forma que todo el auto se dibuje donde corresponde en la pista y con la orientación que corresponde. Para mover el auto, presione la tecla `T` y luego utilice las teclas arriba para acelerar, abajo para frenar y derecha e izquierda para girar.

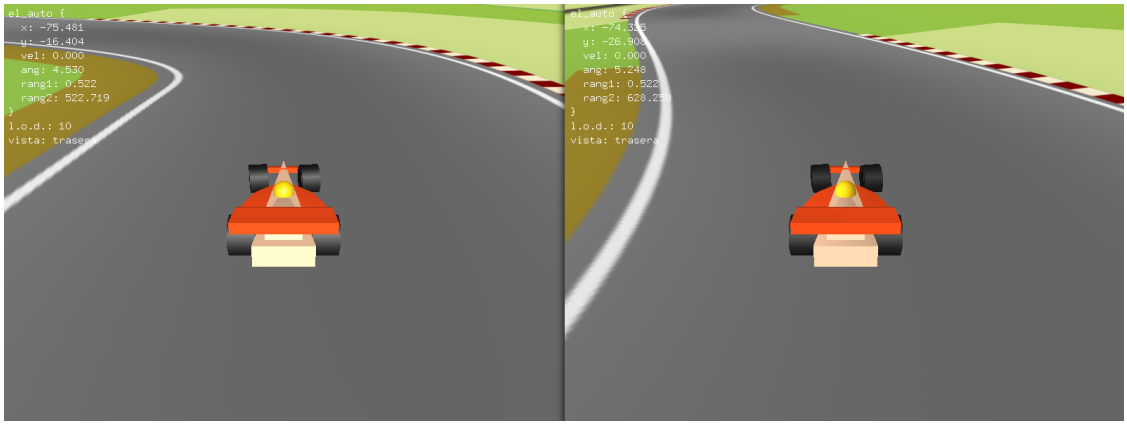


Paso 4: Cámara

1. Modifique la cámara de vista superior para que gire de forma que el auto siempre apunte hacia "arriba".

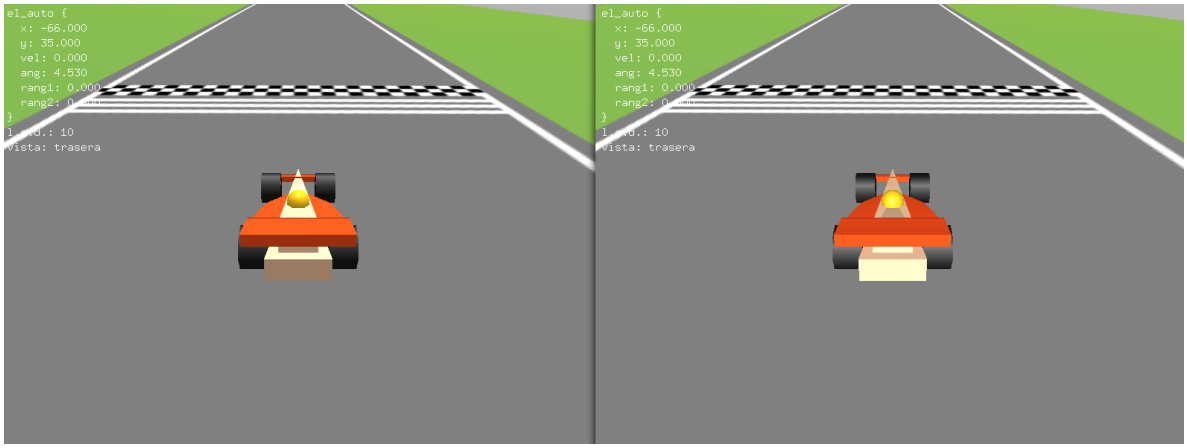


2. Utilice la función `gluLookAt` para implementar la segunda vista (se cambia la vista presionando nuevamente la tecla `T`, o directamente desde la vista del cubo con `R`) y hacer que la cámara siga al auto. Deberá calcular la posición de la misma en base a la posición y dirección del auto.



Paso 5: Luces

Verifique el comportamiento de la luz y compruebe si es afectada por las transformaciones y los desplazamientos de la cámara. La luz debe quedar fija respecta a la pista, y no moverse con el auto o con la cámara. De ser necesario, corrija la posición de la luz.



Ayuda: conduzco el auto en círculos; al girar debería cambiar la orientación del auto respecto del "sol", y por ende su iluminación/sombreado.

Paso 6: ¿A conducir?

Si llegó a este punto comprendiendo los pasos necesarios, ya es suficiente para conducir algunas vueltas y aprobar el TP. Sin embargo, si observa cualquier juego de carreras en 3D, verá que la posición y orientación del auto respecto a la cámara no es fija como en este práctico. Si le sobra tiempo y/o interés, como paso extra puede pensar: ¿cómo haría para "mejorar" el movimiento de la cámara?

---

1. No se permitirá el uso de las funciones `glScale`, `glTranslate` y `glRotate` para fomentar la comprensión de los componentes de una matriz de una transformación afín.↩

2. Notar que la rueda original tiene su eje de giro coincidente con el eje x, pero en el auto final debe estar alineado con el eje y. Recuerde que la función `drawEjes` utiliza los colores rojo, verde, azul para los ejes x, y, z respectivamente.↩

3. "Espejado" no es lo mismo que "rotado 180º". `\>` espejado horizontalmente es `</`, mientras que rotado 180º es `<\`.↩