PROYECTO PROGRAMACIÓN II (503212-1)

TEMA 5: Vehículo dirigido por teclado y moverse por pista sensible con bordes que detectan y cuentan colisiones.

Integrantes:	Daniel Ignacio Soto Salgado
	Gaspar Jimenez Cabezas
	Franchesca Mora Chehuan
Profesor:	Geoffrey Jean-Pierre Christophe Hecht
Referente:	Nicolás Eduardo Rojas Arévalo

INDÍCE

1. ENUNCIADO TEMA 5……PÁG. 3
2. AVANCES PROYECTO……PÁG. 4
   - 2.1) PRIMERA ITERACIÓN……PÁG. 4
   - 2.2) SEGUNDA ITERACIÓN……PÁG. 5
   - 2.3) TERCERA Y ÚLTIMA ITERACIÓN……PÁG. 6
3. VISTAS DEL PROGRAMA……PÁG. 7
   - 3.1) VISTA DEL MENU……PÁG.7
   - 3.2) VISTA DEL SIMULADOR……PÁG.7
4. PATRONES UTILIZADOS……PÁG.8
5. CASOS DE USO……PÁG.9
6. UML DEL CONTENIDO GRÁFICO……PÁG.10
7. UML DEL CONTENIDO LÓGICO……PÁG.11

1. ENUNCIADO TEMA 5:

El panel principal debe contener una ruta cerrada, con curvas y soleras creadas (representadas por Polygon). Las ruedas delanteras deben se visibles y deben reflejar la dirección si dobla. El usuario deberá controlar la dirección de las ruedas con las flechas del teclado. El móvil debe moverse siguiendo la dirección de las ruedas delanteras de manera similar a uno real. Las ruedas no tienen que girar sólo mostrar en la vista aérea su dirección. El control de velocidad se debe realizar mediante controles GUI. La ruta debe ser configurable por controles GUI: ancho y alto la pista y ancho de la calzada. Debe tener dos modos: configuración y conducción.

2. AVANCES PROYECTO

2.1 Primera iteración, 22 de Noviembre de 2022.

Nuestra primera propuesta consiste en crear una autopista junto con un vehiculo que se desplaze dentro de ésta. Dicha autopista estará compuesta por el camino en donde el auto debe moverse y las limitaciones de éste, esto quiere decir que el auto no podrá movilizarse fuera de las areas externas a la pista. De este modo si el auto en algun momenta sobrepasa la linea limite, la acción contará como una colisión o un “choque”, dichas colisiones serán contabilizadas. El movimiento del auto se ejecutará por medio del teclado, este permitirá darle dirección y sentido al desplazamiento del coche. Consideraremos tambíen que tanto el funcionamiento como las mecánicas implementadas en el proyecto, estarán descritas en una subventana a la hora de inicializar el programa.

Aspectos inconclusos por resolver:

- Geometrias del auto a la hora de moverlo. Esto va derechamente con la forma del auto cuando éste este en movimiento.

- Sistema de coliciones, decidir la forma de la pista, junto con implementar las colisiones dentro de esta para que se respeten los limites que el auto no puede pasar.

- Distintos limites de velocidad, posibilidad de que el auto vaya en aceleración o derechamente se detenga.

2.2 Segunda Iteración, 28 de Noviembre 2002.

Decidimos implementar aceleración al vehículo, de este modo el usuario podrá decidir que tan rápido o tan lento desea dirigir el coche para llegar correctamente a la meta y final de la autopista. También al reducir lo suficiente la velocidad, el vehículo podrá derechamente detenerse llegando a 0km/h.

Decidimos que en la ventana se presente el cambio de velocidad del vehículo en km/h determinado por el usuario mediante las flechas del teclado, junto a la cantidad de colisiones que le esta demorando llegar a la meta.

La pista la fijamos como una autopista simple y cerrada.

Problemas definiendo los limites de la autopista, ya que al momento de implementar las colisiones no esta considerando los limites correctos, entonces cuando el auto pasa por un aréa que aun no definimos el programa lo contará como colisión, una colisión incorrecta.

Como otra idea, pensamos en implementar un menu, este ayudara al usuario a entender como funciona el juego, como moverse dentro del juego entre otras características que se irán definiendo en el proceso de construcción.

Aspectos inconclusos por resolver:

- Colisiones que establezcan los limites de la autopista.
- Decidir la forma definitiva de la autopista, para implementar correctamente las colisiones, dado que pensamos en realizarla de modo que este compuesta por más lineas de pista y suba de cierto modo la “dificultad” del juego.
- Decidir si vamos a cambiar el fondo del juego para que se vea más estético.

2.3 Tercera Iteración(Final), 03 de Diciembre de 2022

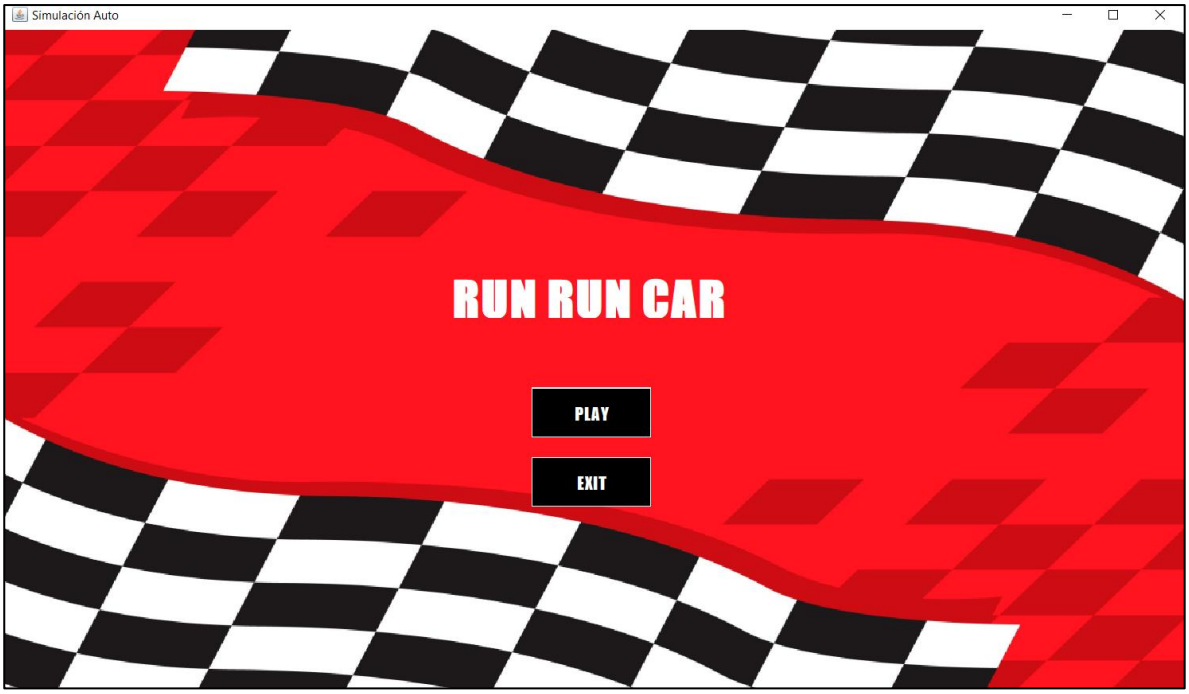
Como equipo junto con nuestro referente del proyecto, decidimos definir la pista con un circuito en fomar rectangular, las colisiones quedaron implementadas alrededor de los limites de la pista. El usuario puede iniciar correctamente el movimiento del vehículo con el teclado, este podrá acelerar, frenar y girar a la izquierda o derecha.

Dentro del panel donde se desarrolla el juego agregamos las indicaciones del movimiento del vehículo para mayor entendimiento del movimiento de este al usuario.

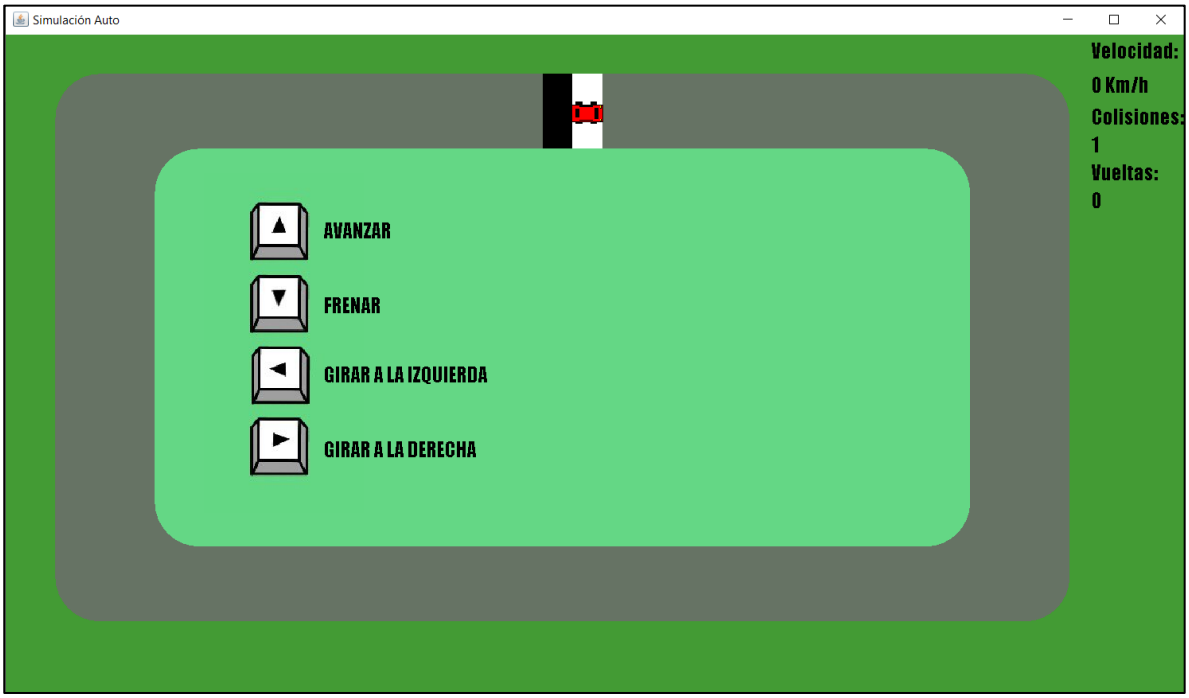
Finalmente, agregamos un Menu a la simulación del vehículo, este cuenta con botones de “Play” el cual le da inicio al funcionamiento del programa pasando al panel en donde se realizan las dinámicas del programa y un boton “Exit” que derechamente te cierra la ventana. Y como último decidimos agregar música de fondo a la simulación.

3. VISTAS DEL PROGRAMA.

3.1 VISTA DEL MENÚ



3.2 VISTA DEL SIMULADOR



4. PATRONES UTILIZADOS.

PATRON BUILDER: Permite construir objetos complejos paso a paso. Este patrón nos permite producir distintos tipos y representaciones de un objeto empleando el mismo código de construcción.

Este es usado a la hora de crear el movimiento del Automóvil con las clases Ruedas y Angulos, dado que estas, le dan el movimiento pedido al vehículo para que se traslade a lo largo del circuito creado.

PATRON SINGLETON: Permite asegurarnos de que una clase tenga una única instancia, a la vez que proporciona un punto de acceso global a dicha instancia.

Este patron está presente en las clases MenuStatus e InGame por ejemplo, dado que al instanciar más de una vez ambas clases el programa tendría problemas al ejecutarse por ende no logrará ser funcional.

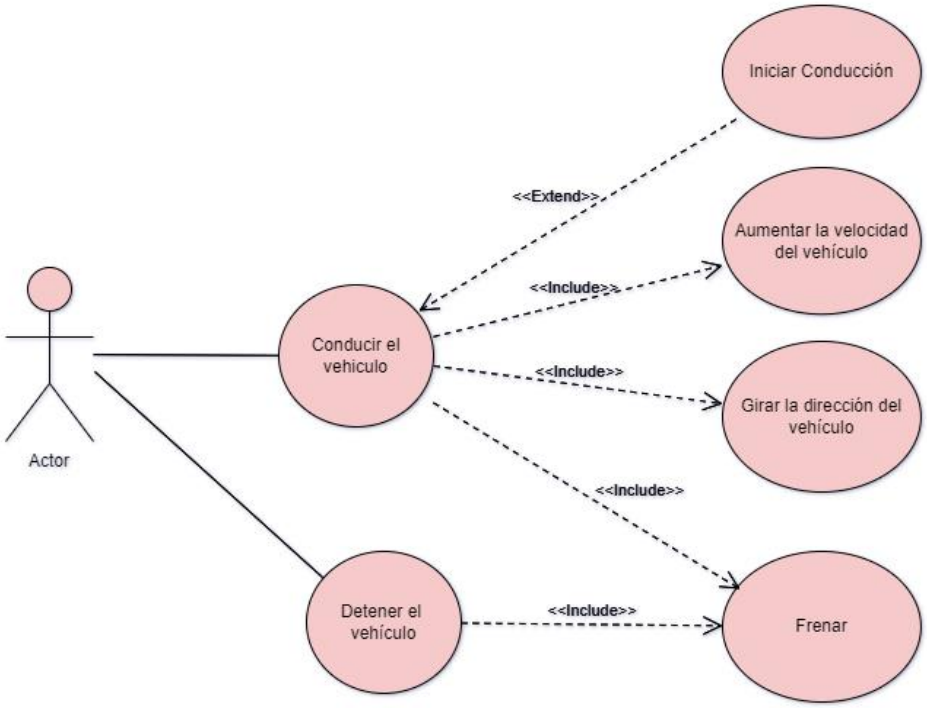
PATRON FACADE: Proporciona una interfaz simplificada a una biblioteca, un framework o cualquier otro grupo complejo de clases.

Este patron es usado a la hora de separar clases en en 3 Packages, Físicas, GUI y Objetos, de este modo es más cómodo y ordenado orgainzar las clases y usar un package dentro de otro, por ejemplo en la clase Ruedas, se importa el package Físicas.Angulos, por ende se logró desarrollar de forma más eficiente el comportamiento de las ruedas del Automovil dentro de la ventana.

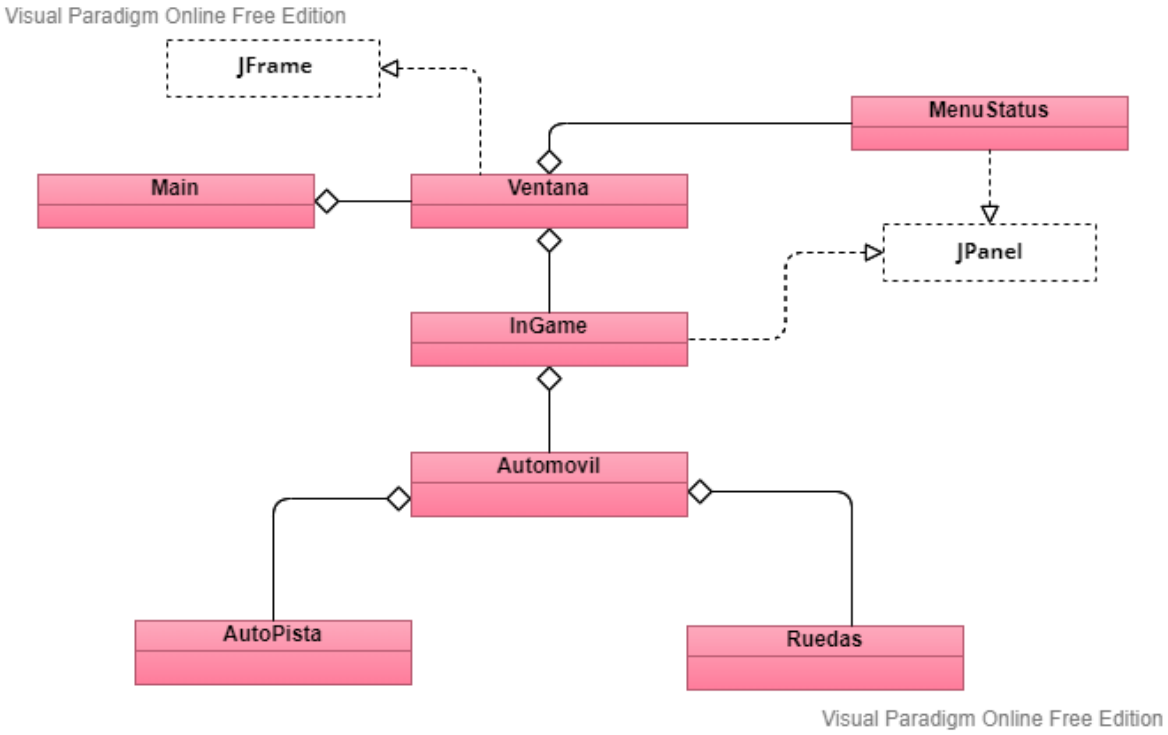
PATRON COMPOSITE: Permite componer objetos en estructuras de árbol y trabajar con esas estructuras como si fueran objetos individuales.

Este patron es usado en la mayor parte del programa realizado, un claro ejemplo es como la clase MenuStatus contiene a la clase InGame y en como InGame contiene a otras clases, y asi hasta lograr que todas conectadas se logre el funcionamiento correcto del programa realizado.

5. CASOS DE USO



6. UML DEL CONTENIDO GRÁFICO



7. UML DE CONTENIDO LÓGICO