Instituto Tecnológico de Costa Rica	Primer Proyecto Programado
Área Académica de Ingeniería en Computadores	**Collision simulator**
CE-1102 Taller de Programación	Prof: Jeff Schmidt Peralta Referencia: Ing. Leonardo Araya
I Semestre 2020 29 de mayo de 2020	Consultas: grupo Telegram correo: jschmidtcr@gmail.com

1. Introducción.

Se va a desarrollar un simulador que recrea el sistema de alerta de colisiones y frenado automático, implementando múltiples funciones utilizando los conceptos de listas, archivos de texto, animaciones básicas y otros.

2. Collision Avoidance System Simulator

La industria automovilística ha visto incrementada su capacidad gracias al constante avance de la tecnología, la cual se pone al servicio de otras áreas que anteriormente eran ajenas a su área de influencia. Esta sinergia entre la ingeniería mecánica y la tecnología, ha redundado en vehículos más seguros, tanto para sus ocupantes como para el resto de usuarios de las vías. Adicionalmente, se han hecho importantes avances en simplificar la forma en la que se conducen los vehículos, de manera que muchas funciones que antes eran responsabilidad del piloto atender, ahora son asumidas por el vehículo. Un ejemplo muy claro, es la incorporación de transmisiones automáticas o asistentes de parqueo.

De acuerdo con la Organización Mundial de la Salud (World Health Organization o WHO por sus siglas en inglés), alrededor de 1.3 millones de personas mueren cada año como consecuencia de accidentes de tránsito. Esto tiene implicaciones sociales, como que es la principal causa de muerte en personas de 5 a 29 años de edad; aparte de las consecuencias económicas que implica para cada país atender dichos accidentes. La WHO estima en 3% del PIB atender los accidentes de tránsito en general.

Volvo (oficialmente AB Volvo, fundada en 1927 en Gotemburgo, Suecia) ha presumido por muchos años ser uno de los fabricantes de automóviles más seguros del mundo; y entre sus mayores aportes destaca la creación del cinturón de seguridad de tres puntos, tal como se conoce ahora.

Una subsidiaria del Grupo Volvo es Volvo Trucks, que se encarga del diseño y fabricación de los vehículos industriales de la marca. Dentro de los sistemas de seguridad que equipan los camiones, destaca el Collision Warning with Emergency Brake (CW-EB), el cual consiste en un sistema que detecta la proximidad de los vehículos circundantes y le genera una alerta al conductor. Si el conductor no toma ninguna acción que evite el impacto, el vehículo automáticamente genera una frenada de emergencia.

Los siguientes videos explican mejor el funcionamiento del CW-EB.

https://www.youtube.com/watch?v=LBT3tB_AQQA&feature=youtu.be

https://www.youtube.com/watch?v=ridS396W2BY&feature=youtu.be

https://www.youtube.com/watch?v=vI9EIjUx20I&feature=youtu.be

3. Descripción Funcional Programa.

La presentación de la interfaz con el usuario es **totalmente libre** y será un elemento importante dentro de la calificación del proyecto.

Al inicio del juego debe mostrarse una pantalla de presentación (splash animado) del desarrollador por cinco segundos (tiempo configurable).

Suponga que Volvo Trucks lo contrata a usted, como ingeniero en Computadores, para realizarle una mejora al sistema CW-EB. Utilizando su conocimiento en la interacción entre hardware y software, la empresa le pide que ahora el sistema se configure, para que no solamente frene de emergencia, si no que intente cambiar de carril cuando las condiciones así lo permitan. La única condición que permitiría un cambio de carril es que no exista un vehículo al lado. Sino se produciría un choque lateral.

Debido a la complejidad del proyecto, se le pide que como primera etapa cree un simulador en Python que simule cómo se comportaría el camión en la vida real. Este proyecto consiste en crear dicho simulador.

El simulador consiste en una calle de dos carriles, en la cual se generan vehículos aleatoriamente en ambos carriles. El camión debe mantenerse a una velocidad constante y debe encargarse de frenar (detenerse) o de realizar los cambios de carril automáticamente cuando el vehículo frente a él esté muy cerca (la cercanía para cambiar de carril estará definida por un parámetro de proximidad). El parámetro de proximidad debe ser configurable por el usuario sin tener que modificar el código al inicio de la ejecución, al igual que otros parámetros. La forma en la que esto se configure queda a su discreción. La figura 1 ilustra cómo se vería el simulador en un modo de conducción normal (el camión simplemente transita por su carril).

El programa debe generar vehículos aleatoriamente en cada carril cada cierto tiempo aleatorio. Si se genera un vehículo en el mismo carril en el que se encuentra el camión, se debe intentar cambiar de carril, si no existen vehículos en el carril de destino. Todos los vehículos deben ser generados con una velocidad aleatoria, pero menor a la velocidad del camión. Las figuras 2 y 3 ilustran cómo se vería el proceso de cambio de carril.

Cuando ya ha acabado el proceso de cambio de carril, se indica que el simulador está en modo de conducción normal de nuevo. Puede existir el caso en el que el cambio de carril no sea posible, porque el carril de destino está ocupado y ocurriría una colisión. En este caso se debe indicar que el cambio no es posible. Refiérase a la figura 4 para entender este caso.

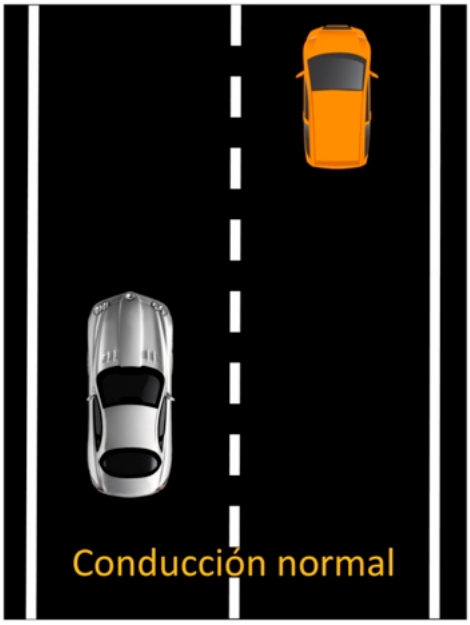

Figure 1: Simulador en conducción normal.

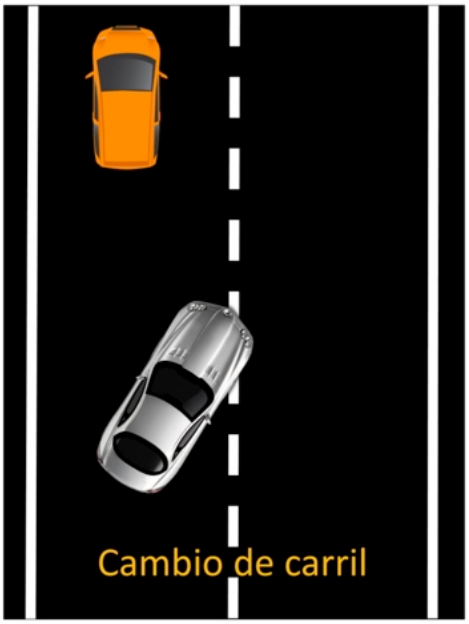

Figure 2: Proceso de cambio de carril.

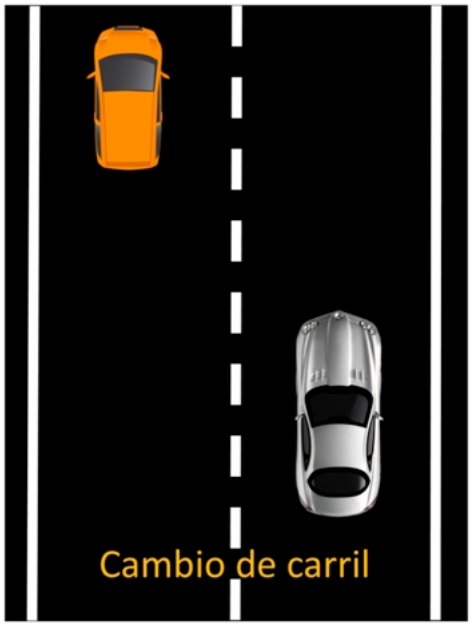

Figure 3: Proceso de cambio de carril.

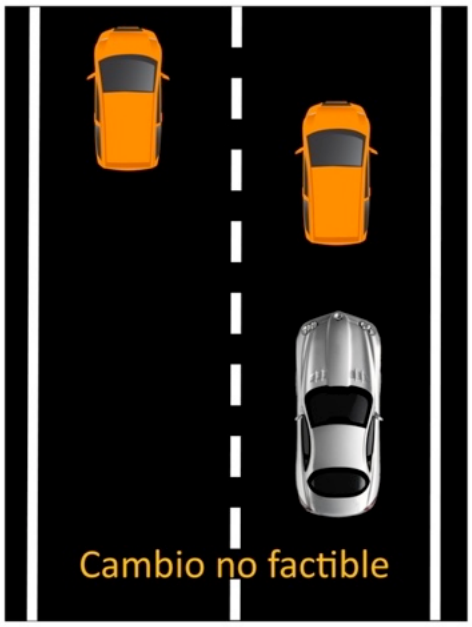

Figure 4: Cambio de carril no posible.

Las imágenes y la animación deben ser definidos por el programador y serán un aspecto importante dentro de la calificación.

El simulador debe contener una configuración, que va a mostrar al menos los siguientes parámetros que regirán el juego, deben configurarse por medio de alguna opción:

- Tiempo de presentación del splash
- Uso del CB-EW
- Parámetro de proximidad
- Indicador de paso de carril o de frenado de emergencia

Los parámetros deben almacenarse en un archivo de texto y recuperarse cada vez que se ejecuta el programa.

Consideraciones a realizar.

- Establecer una relación tiempo – espacio - acción.
- Puede utilizar un modelo de objetos si lo desea. En este caso debe documentarlo adecuadamente en la documentación externa.
- Las funciones deben ser realizadas usando recursividad.
- Considere utilizar la plataforma de desarrollo colaborativo GitHub para el manejo de su código fuente.

4. Funciones a investigar.

Debe **investigarse** el uso de algunas funciones referentes a validaciones de datos y despliegue de información. Las funciones que podrían utilizarse, entre otras son:

- Utilización de multimedia: integración de animaciones, sonidos y otros.
- Generación de números aleatorios
- Manejo de archivos de texto

5. Documentación.

La documentación interna en el programa fuente, debe contener antes de definir cada función, al menos una explicación de lo que realiza la función, las entradas, salidas y restricciones consideradas.

La documentación externa debe incluir:

- Tabla de contenidos o índice
- Introducción
- Descripción del problema.
- Análisis de resultados. (incluyendo corridas de ejemplo). Deben explicarse y mostrarse (puede ser con pantallas de ejemplo) las funcionalidades que se pudieron implementar y las que no fueron posibles.
- Dificultades encontradas: problemas en el desarrollo y que se hizo para corregirlos
- Bitácora de actividades: se deben ir anotando todas las actividades, tipo de actividad, su descripción y duración.

- Estadística de tiempos: un cuadro que muestre un resumen de la Bitácora de Actividades en cuanto las horas **REALES** invertidas. Ejemplo:

FUNCION	**TOTAL**
Requerimientos/diseño	xx horas
Investigación de funciones	xx horas
Programación	xx horas
Documentación interna	xx horas
Pruebas	xx horas
Elaboración documento	xx horas
TOTAL	**xx horas**

- Conclusión personal

6. Evaluación.

Documentación 20%
- Interna 5%
- Externa 15%

Resultados (ejecución, eficiencia, presentación)

Funciones:
- Manejo interfaz 20 %
- Splash animado 5 %
- Manejo general del simulador 50 %
- Parámetros de configuración 5 %

7. Aspectos Administrativos.

- El proyecto es individual.

- Se debe entregar hasta el día 11/06/2020 hasta las 11:59 pm, en forma electrónica, en un archivo comprimido con los nombres de los estudiantes, que contenga TODO lo necesario para poder ejecutarla. Ese día se le asignará una cita de revisión del programa. **No se aceptarán tareas después de la fecha y hora indicadas.** Debe enviarse un archivo readme.txt con la versión de Python a utilizar para la revisión y alguna otra indicación que se considere importante. Se debe enviar al correo tareasintrotaller.ce@gmail.com

- La defensa o revisión del proyecto es indispensable. En esta revisión se preguntará sobre aspectos relacionados con funcionalidad, así como sobre el código. Cada estudiante debe mostrar TOTAL dominio de estos dos temas, de lo contrario, el proyecto puede ser considerado como una copia.

- En caso de probarse algún tipo de fraude en la elaboración de la tarea se aplicarán todas las medidas indicadas al inicio del curso, incluyendo una carta al expediente del estudiante.

- Se debe incluir en el archivo comprimido la documentación solicitada. Debe entregarse en formato electrónico (archivo .pdf).

- No se aceptarán tareas cuyo archivo sobrepase 2 mb de espacio en disco.
- Cualquier falta a los aspectos aquí enunciados implicará pérdida de puntos.

7. Bibliografía.

Documentación técnica Python

8. Consultas.

Puede dirigir cualquier consulta a jschmidtcr@gmail.com o al grupo del curso en Telegram.