Pontificia Universidad Católica Madre y Maestra

(PUCMM)

Integrantes:

Vianny Manuela Cruz Cruz

José Rafael Jáquez

Asignatura:

Programación Paralela y Concurrente

Tema:

Proyecto Final

Profesor:

Freddy Peña

Santiago de los caballeros

28 de agosto de 2024

Documentación - Proyecto Final

Link al repositorio en la rama:
https://github.com/ViannyCruz/ProyectoFinal_ICC303.git

Video ejecutando el programa:
https://youtu.be/uuMogKgHrso

Escenario 1

-Descripción del diseño del sistema.

El sistema es un simulador de tráfico en una intersección con cuatro direcciones (Norte, Sur, Este y Oeste). Se utiliza JavaFX para la interfaz gráfica y se maneja la lógica del tráfico mediante hilos y colas de prioridad de bloqueo. Cada vehículo puede continuar recto, girar a la izquierda o a la derecha, los mismos deben hacerlo en orden de llegada, si existe un vehículo de emergencia este debe tener prioridad.

-Estructura del Proyecto

HelloApplication.java:

- La clase principal que extiende Application de JavaFX.
- Inicializa la interfaz gráfica y carga el archivo FXML.
- Contiene una función con un hilo de trabajo que maneja la lógica del tráfico, incluyendo la gestión de vehículos y la detección de emergencias.

HelloController.java:

- Controlador de la interfaz gráfica.
- Maneja la creación de vehículos y la lógica de movimiento hacia todas las direcciones.
- Contiene métodos para agregar vehículos comunes y de emergencia a las colas de prioridad y para manejar las acciones de los botones de la interfaz.

Vehicle.java:

- Clase que representa un vehículo en el simulador incluyendo su respectiva imagen, dirección a tomar y ubicación inicial.
- Implementa Comparable para permitir la ordenación en las colas de prioridad.

hello-view.fxml:

- Archivo FXML que define la interfaz gráfica.
- Contiene botones para crear vehículos en diferentes direcciones y para manejar emergencias.
- Incluye un StackPane para mostrar las imágenes de los vehículos.

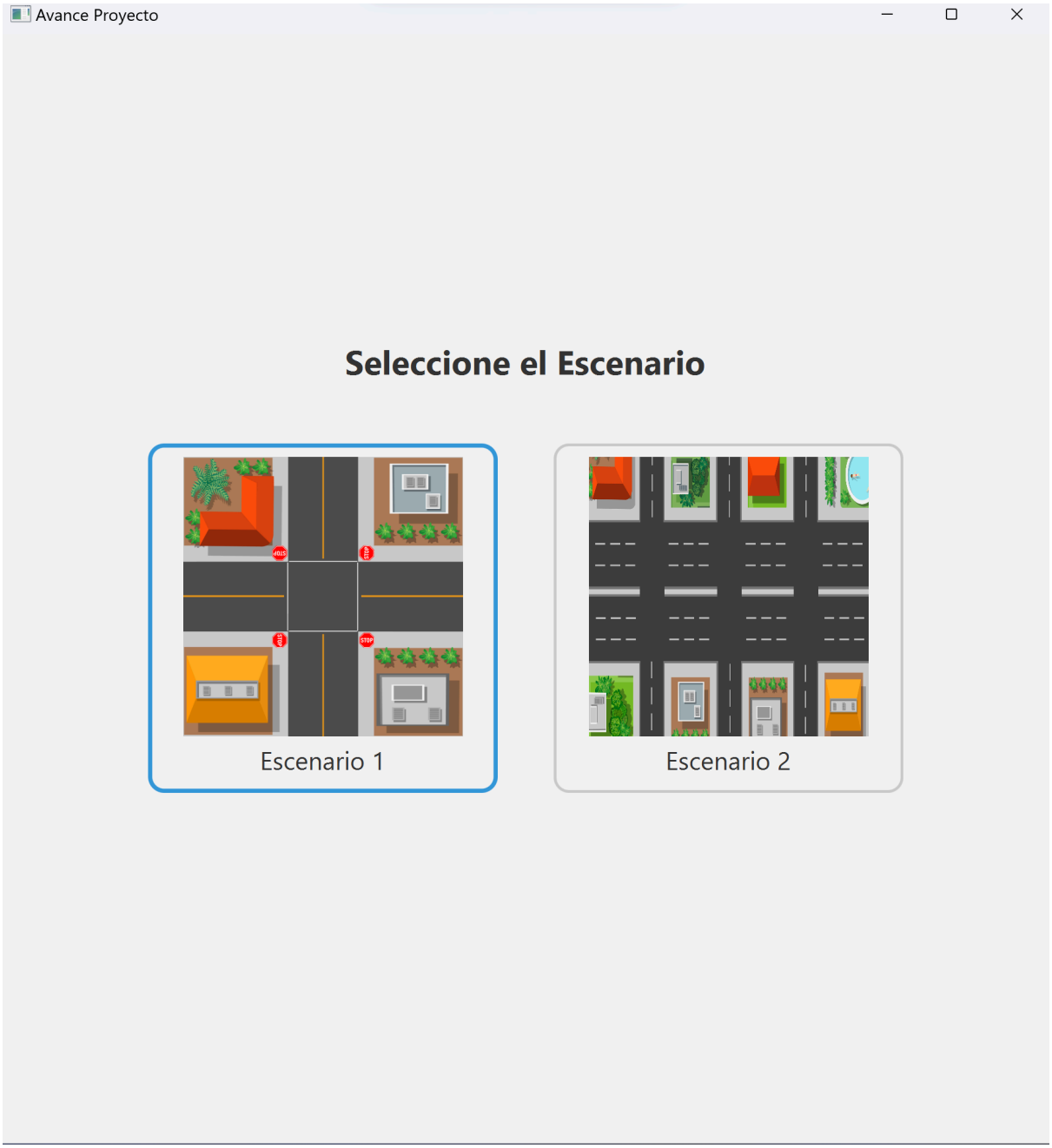
-Explicación de los algoritmos de control.

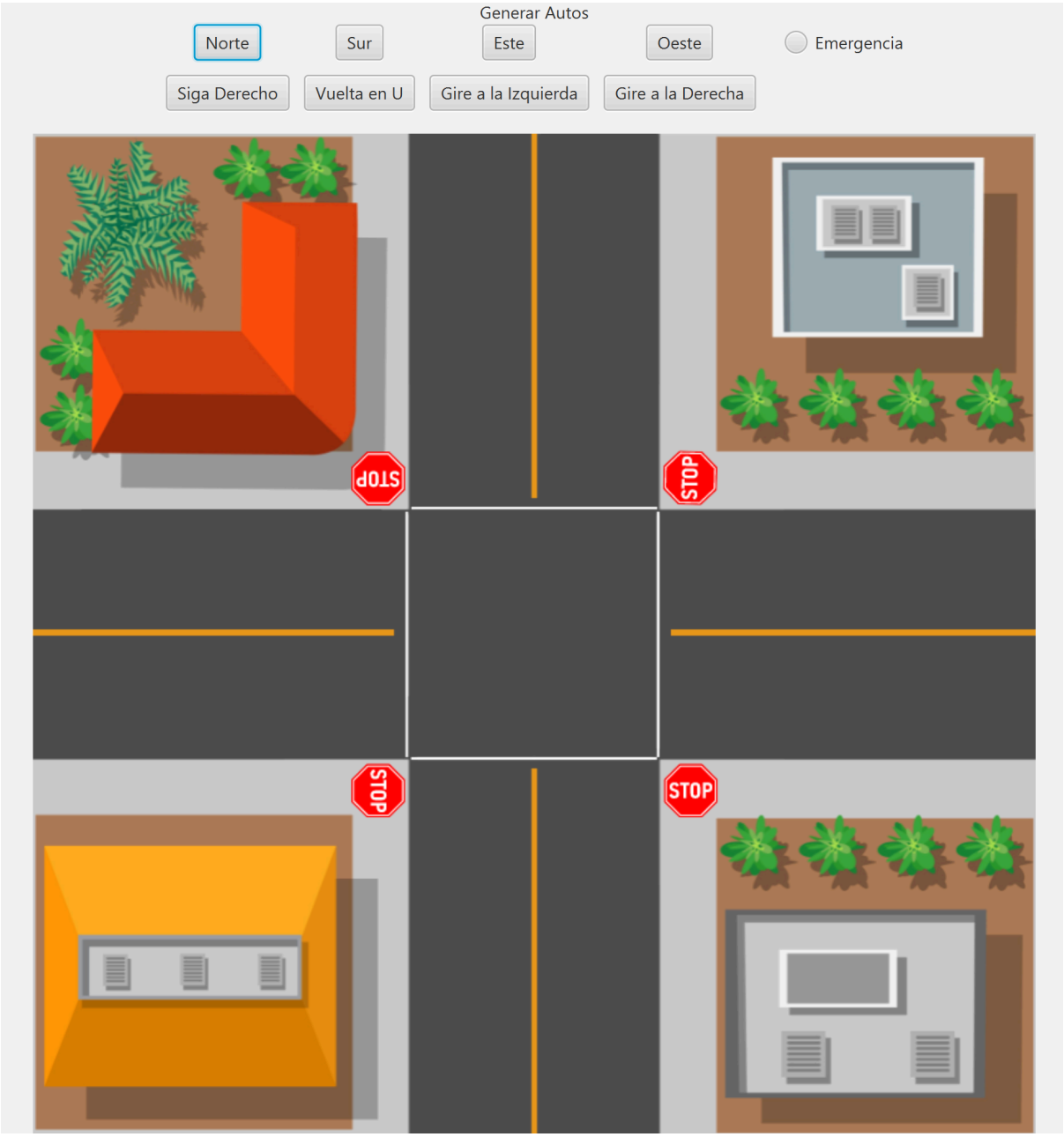
El sistema se basa en un algoritmo de control que inicializa y carga la interfaz gráfica mediante la clase HelloApplication. HelloController gestiona la creación y gestión de vehículos a través de acciones de botones en la interfaz, donde cada vehículo se añade a una cola de prioridad según su dirección y tipo. Cada acción ejecutada desde una función distinta dependiendo de los botones y las acciones seleccionadas. Un hilo de trabajo en HelloApplication maneja la lógica del tráfico, verificando periódicamente las colas de vehículos y priorizando el movimiento de aquellos en estado de emergencia. En caso de emergencia, el sistema procesa y mueve los vehículos a través de la intersección hasta que se encuentra y procesa el vehículo de emergencia. Si no hay emergencias, el sistema procesa los vehículos en orden de prioridad en cada dirección, utilizando TranslateTransition de JavaFX para animar el movimiento de los vehículos y actualizar las posiciones de los vehículos restantes en la cola. Este enfoque asegura una gestión eficiente y segura del tráfico en un entorno multi-hilo, proporcionando una experiencia visual fluida y dinámica.

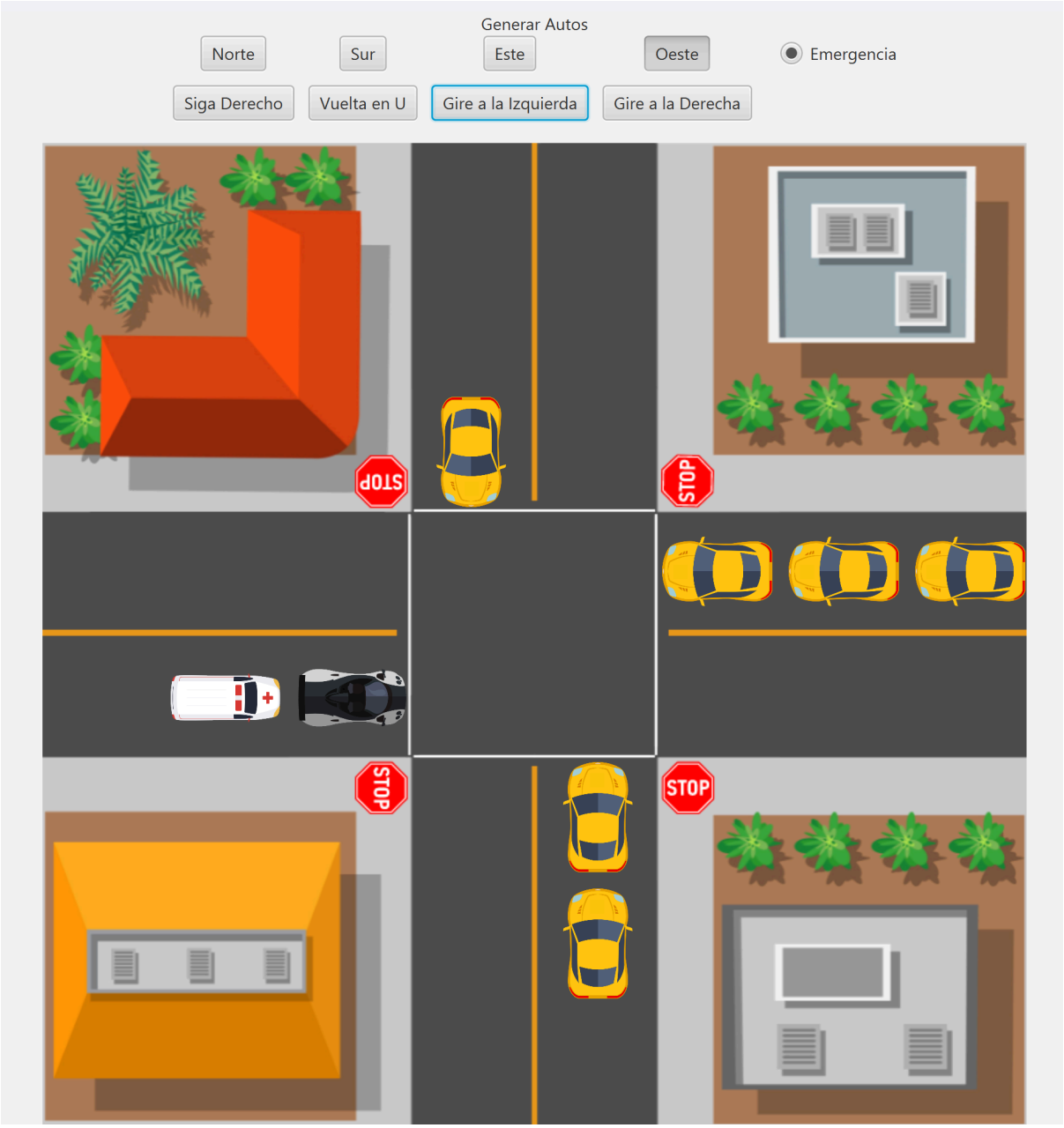
-Instrucciones para ejecutar la aplicación.

Para ejecutar la aplicación basta seleccionarlo en el menú de escenarios al correr el archivo HelloApplication.java del paquete com.example, luego, aparecerá la interfaz que permitirá despegar vehículos en distintas direcciones para así tener la simulación.

-Capturas de pantalla de la interfaz de usuario.




Generar Autos
Norte
Sur
Este
Oeste
Emergencia
Siga Derecho
Vuelta en U
Gire a la Izquierda
Gire a la Derecha
STOP
STOP
STOP
STOP


Generar Autos
Norte
Sur
Este
Oeste
Emergencia
Siga Derecho
Vuelta en U
Gire a la Izquierda
Gire a la Derecha
STOP
STOP
STOP
STOP

-Resultados de pruebas y evaluación del sistema
Al momento de la creación de nuevos vehículos se muestran correctamente en la posición adecuada, tanto los vehículos comunes como los de emergencia, estos últimos teniendo preferencia para adelantar aún si no fue el primero en llegar, por otro lado al crear una gran cantidad de vehículos, principalmente varios vehículos de emergencia, se puede apreciar una disminución en la efectividad del mismo debido a errores como la prioridad en vehículos de emergencia y ciertas colisiones visuales.

Escenario 2

Descripción del Diseño del Sistema
El sistema es un controlador de tráfico para 2 calles y 3 intersecciones con 3 carriles y 3 direcciones. El sistema está diseñado para gestionar el flujo de vehículos, priorizando los vehículos de emergencia cuando sea necesario. Se utiliza una combinación de colas prioritarias, semáforos, y un sistema de animaciones para simular el movimiento de los vehículos a través del escenario.

Estructura del Proyecto
El proyecto está estructurado en dos clases principales:

1. TrafficController_02: Esta clase es responsable de gestionar la actualización de los vehículos.

2. HelloController: Esta clase es el controlador de la interfaz gráfica y se encarga de la creación de vehículos, la gestión de los semáforos, y la actualización de las posiciones de los vehículos en la interfaz.

Explicación de los Algoritmos de Control
TrafficController_02
- Constructor y Thread de Procesamiento: Al instanciar `TrafficController_02`, se inicia un hilo de trabajo (`workerThread`) que procesa la cola de vehículos en un bucle continuo.

- Método `processQueue`: Este método es el núcleo del controlador. Revisa si hay vehículos de emergencia en `Emergencyqueue` y los procesa primero. Si no hay vehículos de emergencia, procesa los vehículos en `queue`. Utiliza `addVehicleAnimation` para animar el movimiento de los vehículos.

- Método `addVehicleAnimation`: Este método añade una tarea a la cola de tareas (`tasks`) para animar el movimiento de un vehículo. Utiliza `CountDownLatch` para sincronizar la animación con la actualización de la interfaz.

HelloController

- Inicialización y Configuración: En el método `initialize`, se configuran los semáforos y se inicia el ciclo de cambio de luces de los semáforos.

- Métodos de Creación de Vehículos: Métodos como `handleCreateVehicleCenterWest`, `handleCreateVehicleCenterEast`, etc., crean nuevos vehículos y los añaden a las colas correspondientes.

- Métodos de Actualización de Posiciones: Métodos como `updatePositionsCenterWest`, `updatePositionsRightWest`, etc., actualizan las posiciones de los vehículos en la interfaz. Estos métodos utilizan algoritmos para determinar la siguiente posición del vehículo y animan su movimiento.

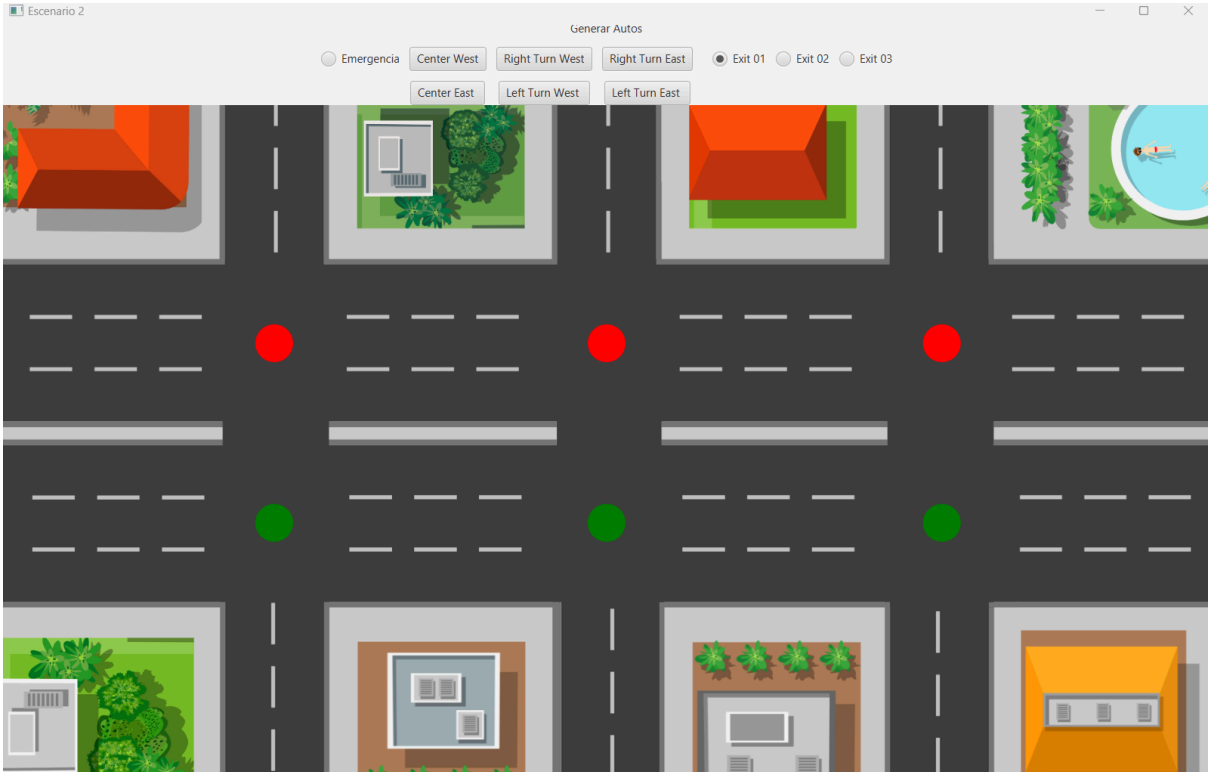
- Métodos de Movimiento de Vehículo: Métodos como `moveVehicle_Fix`, `moveVehicle_Fix_Right`, etc., animan el movimiento de los vehículos utilizando `TranslateTransition` y `RotateTransition` de JavaFX.

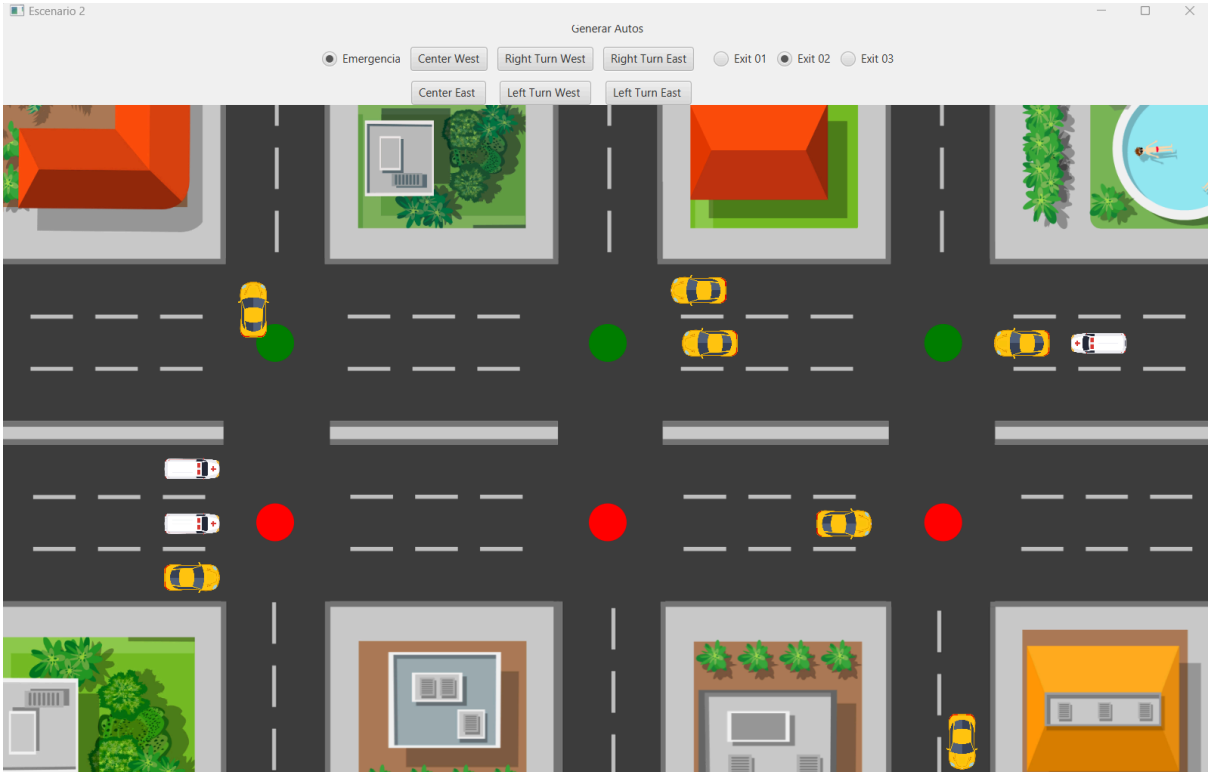
- Métodos de Posicionamiento: Métodos como `getNextPosition_Fix`, `getNextPosition_Fix_Right`, etc., determinan la siguiente posición del vehículo basándose en la lógica de la intersección y el estado de los semáforos.

-Instrucciones para ejecutar la aplicación.

Para ejecutar el escenario 2, primero debemos de ejecutar la clase “HelloApplication” que se encuentra en la carpeta “com.example”, luego, veremos 2 botones, y ahi seleccionamos el correspondiente al escenario 2.

-Capturas de pantalla de la interfaz de usuario.





-Resultados de pruebas y evaluación del sistema

Luego de generar varias situaciones en el simulador, podemos afirmar que este funciona adecuadamente para el escenario que representa. Debido a la lógica utilizada, al tener muchos automóviles tanto normales como de emergencia se puede percibir que el simulador se ralentiza, esto debido a todas las decisiones que se deben tomar para evitar colisiones, pero a pesar de esto los autos terminan de llegar a sus posiciones previstas de manera segura.