PROYECTO FINAL

Simulador de Tráfico con Autómatas Celulares

Nicolás Alberto Arciniegas Rincón - 2240087
Jheither Santiago Ayala Anaya - 2240042
Luciana Moran Rubiano - 2232960
María Camila Cervera Rey - 2240041
Arlex Stiven Rodríguez Barrera - 2230066
Juan Esteban Garcia Soler - 2231892

Introducción al Proyecto

La **simulación con autómatas celulares** permite representar de manera gráfica y dinámica el comportamiento de sistemas complejos, como el tráfico. Este proyecto se enfoca en modelar una carretera con semáforos, ayudando a entender la relación entre vehículos y señales de tránsito.

Aprender sobre estas simulaciones es **fundamental** para analizar y mejorar el tráfico urbano, promoviendo una mejor planificación y gestión vial.

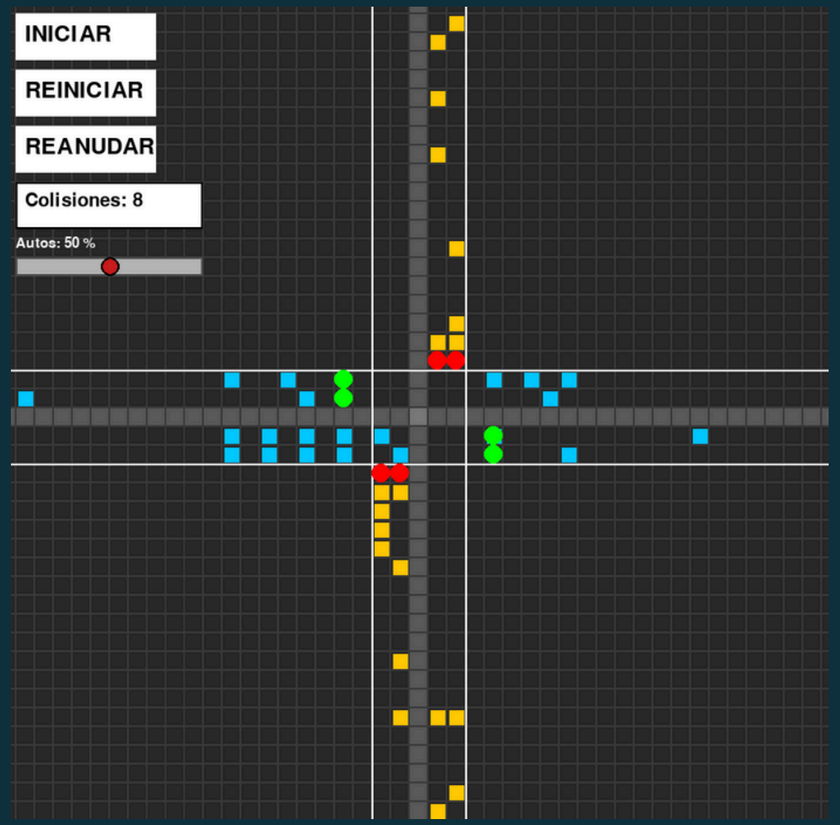
Justificación del proyecto

¿Para qué sirve esto?

Gracias a la simulación de vías, esto nos permite **experimentar y optimizar** los tiempos de los semáforos, probando distintas sincronizaciones con el fin de reducir los tiempos de espera y mejorar la **eficiencia** en la movilidad vehicular.

A partir del análisis de colisiones, es posible **determinar** de forma visual y estadística los puntos con mayor recurrencia en una vía específica, evaluando alternativas para **disminuir** su incidencia.

Funcionamiento



- El estado de un vehículo cambia según las celdas vecinas y los semáforos.
- Los semáforos alternan entre rojo y verde.
- Botones de inicio, contador de colisiones y controlador de flujo.
- Al alcanzar cierto número de colisiones se detiene el movimiento.

Reglas del Autómata Celular

1

- Si el semáforo está en verde el vehículo se mueve, si está en rojo se detiene.
- Si la celda siguiente está vacía el vehículo se mueve.

2

- Si la celda siguiente está ocupada por un vehículo del mismo tipo se detiene.
- Si la celda a la que desea moverse es igual a la de otro vehículo del mismo tipo, se elige uno al azar para ocuparla.

3

- Si la celda siguiente está ocupada por un vehículo de otro tipo se presenta una colisión.
- Si la celda siguiente está fuera de la cuadrícula, el vehículo desaparece.

Explicación sobre el código

Librerías

PYGAME

- Ventana de simulación.
- Herramientas de diseño.
- Actualización de la pantalla.

```
import pygame
```

RANDOM

- Generación de vehículos.
- Problema sobre estos mismos.

```
import random
```

SYS

- Funcionalidad del sistema operativo.
- sys.exit()

```
import sys
```

Clases

- TrafficLight: Administra los semáforos y su ciclo de luces.
- Vehicle: representa cada carro individual del mapa.
- Slider: control gráfico que le permite al usuario cambiar un valor arrastrando una barra.

Grid: integra y coordina todo el sistema

- Controla la lógica del tráfico.
- Maneja el movimiento y eliminación de vehículos.
- Cantidad de colisiones.
- Activación del modo de emergencia.
- Se encarga del renderizado completo.

¿Cómo funciona el programa?


```python
class Slider:
    def __init__(self, x, y, width, height, min_val=0, max_val=100, start_val=30):
        self.rect = pygame.Rect(x, y, width, height)
        self.min_val = min_val
        self.max_val = max_val
        self.value = start_val

        self.knob_radius = max(6, height // 2)
        # knob_x stores integer pixel coordinate
        self.knob_x = int(x + (start_val - min_val) / (max_val - min_val) * width)
        self.dragging = False
```

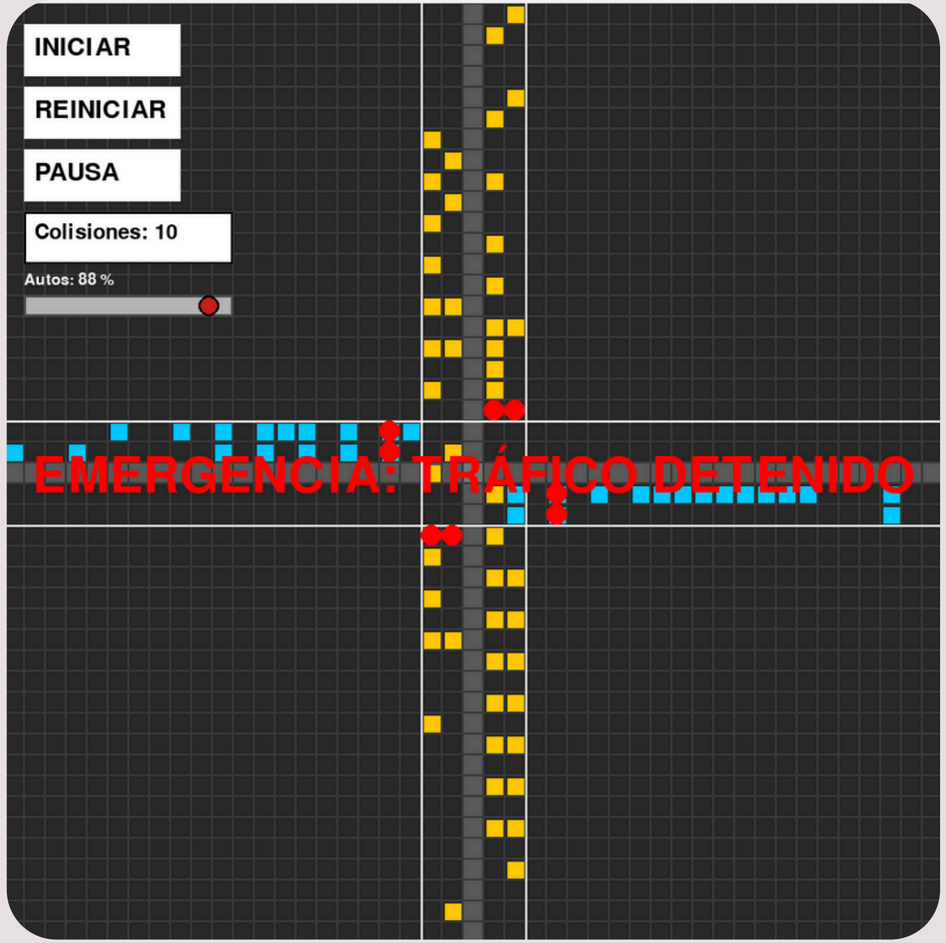
```python
#Controlar el spawn de vehiculos
s = max(0.0, min(1.0, self.spawn_scale))

if random.random() < 0.15 * s:
    # Horizontal - Carril superior
   self.add_vehicle(0, self.rows // 2 - 1, 1, 0, COLOR_AUTO_H) # Fila cy - 1
#CARRIL 1 IZQ-DER
if random.random() < 0.25 * s:
    self.add_vehicle(0, self.rows // 2-2, 1, 0, COLOR_AUTO_H)
#CARRIL 1 DER-IZQ
if random.random() < 0.25 * s:
    self.add_vehicle(self.cols -1, self.rows // 2 + 1, -1, 0, COLOR_AUTO_H)
#CARRIL 2 DER-IZQ
if random.random() < 0.15 * s:
    self.add_vehicle(self.cols -1, self.rows // 2 +2, -1, 0, COLOR_AUTO_H)

#CARRIL 1 ABAJO-ARRIBA
if random.random() < 0.25 * s:
   self.add_vehicle(self.cols // 2 -2, self.rows - 1, 0, -1, COLOR_AUTO_V) # Columna cx + 1
#CARRIL 2 ABAJO-ARRIBA
if random.random() < 0.2 * s:
    self.add_vehicle(self.cols // 2 -1, self.rows - 1, 0, -1, COLOR_AUTO_V)
# CARRIL 1 ARRIBA-ABAJO
if random.random() < 0.2 * s:
    self.add_vehicle(self.cols // 2 + 2, 0, 0, 1, COLOR_AUTO_V)
# CARRIL 2 ARRIBA-ABAJO
if random.random() < 0.27 * s:
    self.add_vehicle(self.cols // 2 +1, 0, 0, 1, COLOR_AUTO_V)
```

¿Cómo funciona el programa?

Contador de Colisiones n*(10)



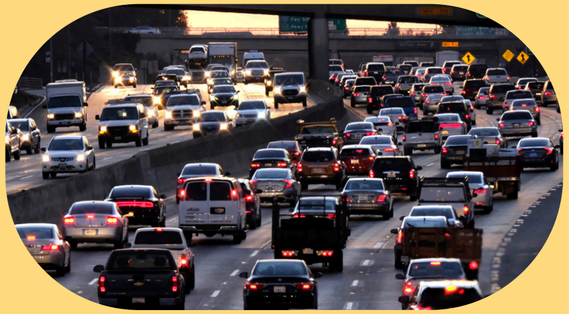
Link Video

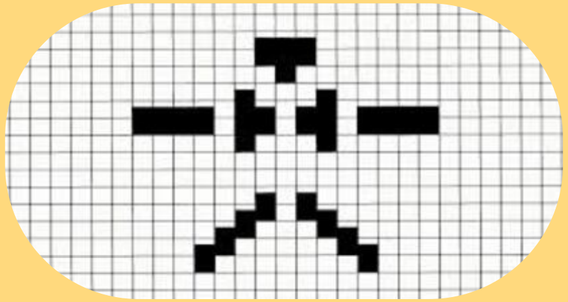
Link repositorio

Conclusiones

En resumen...

En resumen, la simulación nos permite entender mejor el comportamiento del tráfico y la dinámica de los semáforos mediante **autómatas celulares.** Este proyecto ha permitido mostrar cómo se pueden modelar sistemas complejos y lo interesante que puede ser trabajar con autómatas celulares.





Muchas Gracias