

# Symulacja ruchu drogowego na przykładzie ronda Grunwaldzkiego w Krakowie

Projekt zespołu 05 na przedmiot  
*Symulacja Systemów Dyskretnych*

Łukasz Łabuz  
Dawid Małecki  
Mateusz Mazur

13 grudnia 2023

## Postępy prac

## Prace wykonane na rzecz projektu w okresie od ostatniego spotkania

- 1 Finalizacja prac nad modelem formalnym
- 2 Stworzenie dokumentu z opisem modelu formalnego
- 3 Rozpoczęcie implementacji modelu formalnego
- 4 Implementacja GUI symulatora

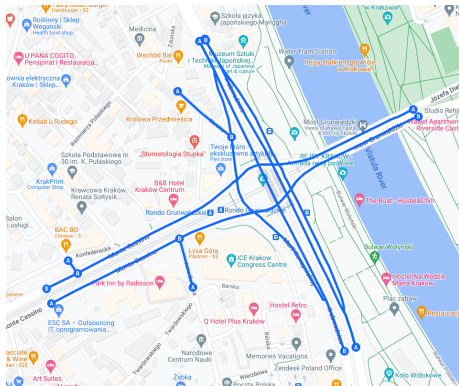
## Zestawienie osób i wykonanych przez nie zadań

Zadanie	Łukasz Łabuz	Dawid Małecki	Mateusz Mazur
Finalizacja prac nad modelem formalnym	✓	✓	✓
Stworzenie dokumentu z opisem modelu formalnego			✓
Rozpoczęcie implementacji modelu formalnego	✓	✓	✓
Implementacja GUI symulatora			✓

## Model formalny

## Przypomnienie celu projektu

Celem projektu jest stworzenie modelu symulacyjnego ruchu drogowego na rondzie Grunwaldzkim w Krakowie.



Rysunek 1: Obszar symulacji. Źródło: Google My Maps

# Materiały źródłowe

Kwerenda literaturowa naszego projektu została podzielona na dwie sekcje - główną oraz pomocniczą.

Materiały z sekcji głównej stanowią podstawę naszego modelu formalnego, natomiast materiały z sekcji pomocniczej są dodatkowymi źródłami informacji, które mogą okazać się przydatne w trakcie jego implementacji.

## Główne materiały źródłowe

- Gora P. *Adaptacyjne planowanie ruchu drogowego* [1]
- Rasouli A. *Pedestrian Simulation: A Review* [2]

## Skale oraz technika symulacji

W materiale [2] przedstawione zostały definicje różnych skal oraz technik symulacji.

W naszym projekcie wykorzystujemy następujące:

### Techniki symulacji

**Model komórkowy** - model polegający na dyskretyzacji obszarów, na których poruszają się symulowane jednostki. Według założenia, każda z nich może zajmować jedną komórkę na siatce w danym momencie. W każdym kroku symulacji, jednostki mogą zmienić swoją pozycję na sąsiednią komórkę.



## Skale symulacji

**Agent-Based** - skala, w której każda jednostka jest rozróżnialna, ma własne, zdefiniowane statystyki oraz zbiór możliwych do podjęcia decyzji. Na jej zachowanie ma wpływ otoczenie, infrastruktura czy też inne jednostki.

**Entity-Based** - skala, w której jednostki są z założenia nierozróżnialne. Nie wyróżniają się niczym. Zachowują się według ściśle ustalonych reguł. Nie mają wpływu na otoczenie.

# Elementy modelu formalnego

## Automat komórkowy

W naszym modelu formalnym wykorzystujemy definicję automatu komórkowego przedstawioną w [1] - rysunek 2.

**Definicja 1.3.1.** *Automat komórkowy to krotka:*

$$CA = \langle T, C, N, S, S_0, F \rangle, \quad (1.1)$$

gdzie:

- $T$  - Przedział czasu, w którym odbywa się ewolucja automatu ( $T = \{0, 1, 2, \dots, T_{MAX}\}$ , gdzie  $T_{MAX} \in \mathbb{N} \cup \{\infty\}$ )
- $C$  - Zbiór komórek
- $N : C \rightarrow \mathcal{P}(C)$  - Funkcja, która każdej komórce ze zbioru  $C$  przyporządkowuje jej otoczenie
- $S$  - Zbiór możliwych stanów komórek
- $S_0 : C \rightarrow S$  - Początkowa konfiguracja komórek (stan komórek w chwili  $t = 0$ )
- $F : T \times C \rightarrow S$  - Reguła przejścia, taka że  $\forall c \in C \forall t \in T \ c_{t+1} = F(t, c)$ , gdzie  $c_t$  - stan komórki  $c$  w chwili  $t \in T$ .

Rysunek 2: Definicja Automatu komórkowego przedstawiona w [1]

## Jednostki

Algorytmy zachowania jednostek, które wchodzi w skład naszego modelu, przedstawiają następujące rysunki:

- pojazdy - rysunek 3
- piesi - rysunek 4
- sygnalizacja świetlna - rysunek 5.

---

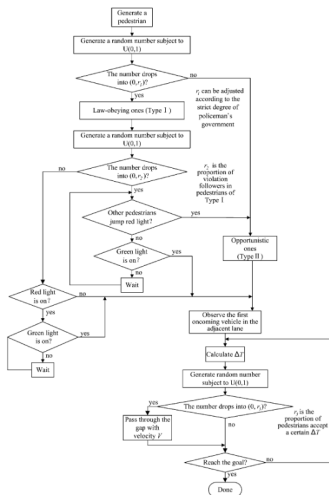
**Algorithm 2** Algorytm ruchu pojazdu *car* w kroku *t*

---

**Require:**  $G = (V, E)$ ,  $t \in T$ ,  $car \in CARS(t)$ ,  $turnPenalty$ ,  $crossroadPenalty$ ,  $prob$ *increaseVelocity(car, t);***if** *stopOnSignal(car, t)* **then***reduceVelocityOnSignal(car, t)***else****if** *turnOnCrossroad(car, t)* **then***reduceVelocity(car, t, turnParameter);***else****if** *crossroad(car, t)* **then***reduceVelocity(car, t, crossroadParameter);***end if****end if****end if****if** *shouldChangeLane(car, t)* **then***changeLane(car, t);***end if***safeReduceVelocity(car, t)*with probability *prob*: *reduceVelocity(car, t);**makeMove(car, t);*

---

Rysunek 3: Algorytm ruchu pojazdów przedstawiony w [1]



Rysunek 4: Algorytm ruchu pieszych przedstawiony w [2]

---

**Algorithm 1** Algorytm przejścia dla sygnalizacji świetlnej  $s$  w kroku  $t$

---

**Require:**  $s \in \text{SIGNALS}(G), t \in T$

**Ensure:**  $\text{state}(s, t + 1) \in \{\text{GREEN}, \text{RED}\}$

```

if  $t_{\text{change}}(s) > 0$  then
     $t_{\text{change}}(s) := t_{\text{change}}(s) - 1$ 
     $\text{state}(s, t + 1) := \text{state}(s, t)$ 
    return  $\text{state}(s, t + 1)$ 
else
    if  $\text{state}(s, t) = \text{RED}$  then
         $t_{\text{change}}(s) := t_{\text{green}}(s)$ 
         $\text{state}(s, t + 1) := \text{GREEN}$ 
    else
         $t_{\text{change}}(s) := t_{\text{red}}(s)$ 
         $\text{state}(s, t + 1) := \text{RED}$ 
    end if
    return  $\text{state}(s, t + 1)$ 
end if
    
```

---

Rysunek 5: Algorytm działania sygnalizacji świetlnej przedstawiony w [1]

## Podsumowanie

Praca [1], oprócz wyżej wymienionych definicji i algorytmów (Rysunki 2, 3, 5), zawiera również szerokie opisy poszczególnych elementów modelu oraz ich zachowań.

Praca [2] zawiera krótki, ale konkretny opis algorytmu ruchu pieszych oraz ich zachowania.

Na bazie tych materiałów, stworzyliśmy model formalny, który posłuży nam jako podstawa implementacji symulatora.



## Model formalny

Nasz model jest połączeniem elementów z obu prac.

Dzięki obszernym opisom z pracy [1] łatwo było nam zrozumieć, jak poszczególne elementy modelu powinny ze sobą współpracować oraz jak, do modelu przedstawionego przez jego autora, dodać symulację pieszych z pracy [2].

Stworzenie aplikacji symulacyjnej na podstawie tak przygotowanego modelu nie powinno zatem stanowić problemu. Potwierdzają to dotychczasowe postępy prac.

## Implementacja symulatora

## Minione zadanie - Próba implementacji prostego modelu NaSch

Jakiś czas temu postanowiliśmy spróbować zaimplementować prosty model NaSch, aby lepiej zapoznać się z biblioteką *CellPyLib* oraz problematyką projektu.

Wyniki zadania pokazały, że implementacja modelu NaSch nie jest trudna, ale biblioteka *CellPyLib* nie nadaje się idealnie do implementacji tego typu modelu.

Po dalszej analizie zdecydowaliśmy się zrezygnować z biblioteki *CellPyLib* i zaimplementować model samodzielnie.

## Aktualny stan prac

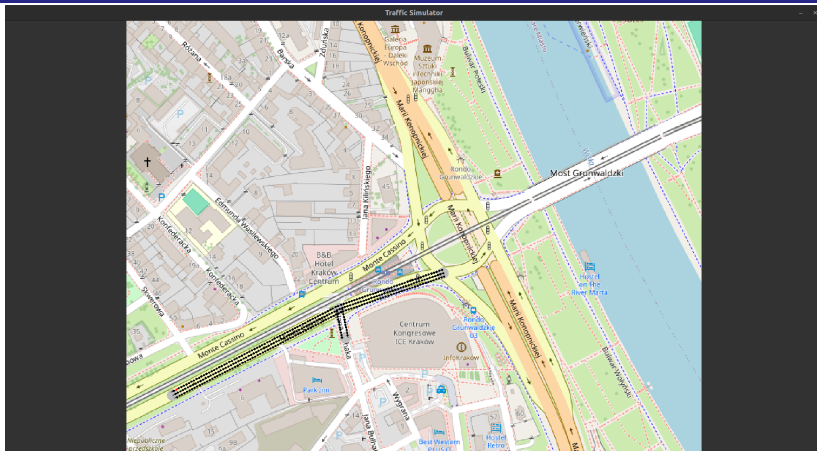
W trakcie ostatnich dwóch tygodni udało nam się zaimplementować podstawowe elementy składowe modelu formalnego.

Naszą uwagę skupiliśmy głównie na implementacji GUI symulatora, aby móc w łatwy sposób testować działanie naszego modelu.

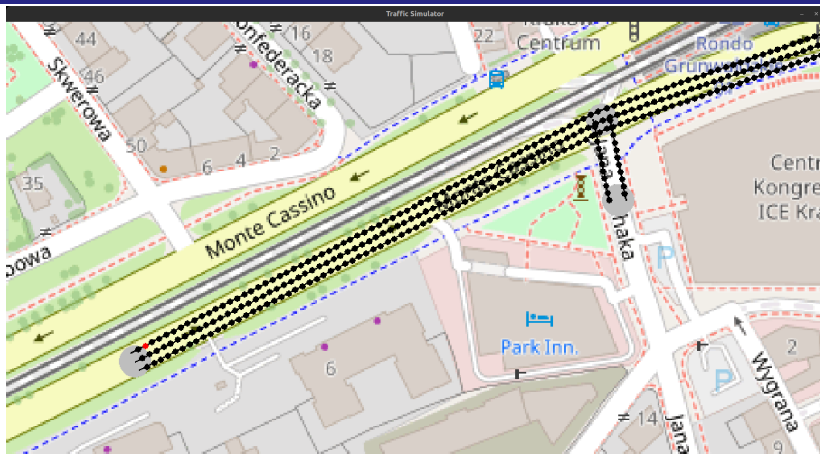
### GUI symulatora

Nowe GUI symulatora zostało wykorzystując bibliotekę *Pygame*.

Umożliwia ono wyświetlanie stanu symulatora w czasie rzeczywistym. Możliwe jest przybliżanie (klawisze z,c) i przesuwanie widoku (strzałki). Obecny wygląd GUI symulatora przedstawiają rysunki 6 oraz 7.



Rysunek 6: GUI symulatora - widok ogólny. Dla zwiększenia czytelności graf dróg jest nanoszony na obraz z map (źródło: OpenStreetMap). Wierzchołki grafu (skrzyżowania) przedstawione są jako szare punkty. Krawędzie składają się z co najmniej jednej, kropkowanej linii. Linie obrazują pasy ruchu, a kropki komórki automatu.



Rysunek 7: GUI symulatora - widok przybliżony. Dla zwiększenie czytelności graf dróg jest nanoszony na obraz z map (źródło: OpenStreetMap). Wierzchołki grafu (skrzyżowania) przedstawione są jako szare punkty. Krawędzie składają się z co najmniej jednej, kropkowanej linii. Linie obrazują pasy ruchu, a kropki komórki automatu.

## Pytania

Dziękujemy za uwagę



## Bibliografia

# Bibliografia

- [1] Gora, P. 2010. *Adaptacyjne planowanie ruchu drogowego*. Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki.
- [2] Rasouli, A. 2021. Pedestrian simulation: A review. *arXiv preprint arXiv:2102.03289*. (2021).