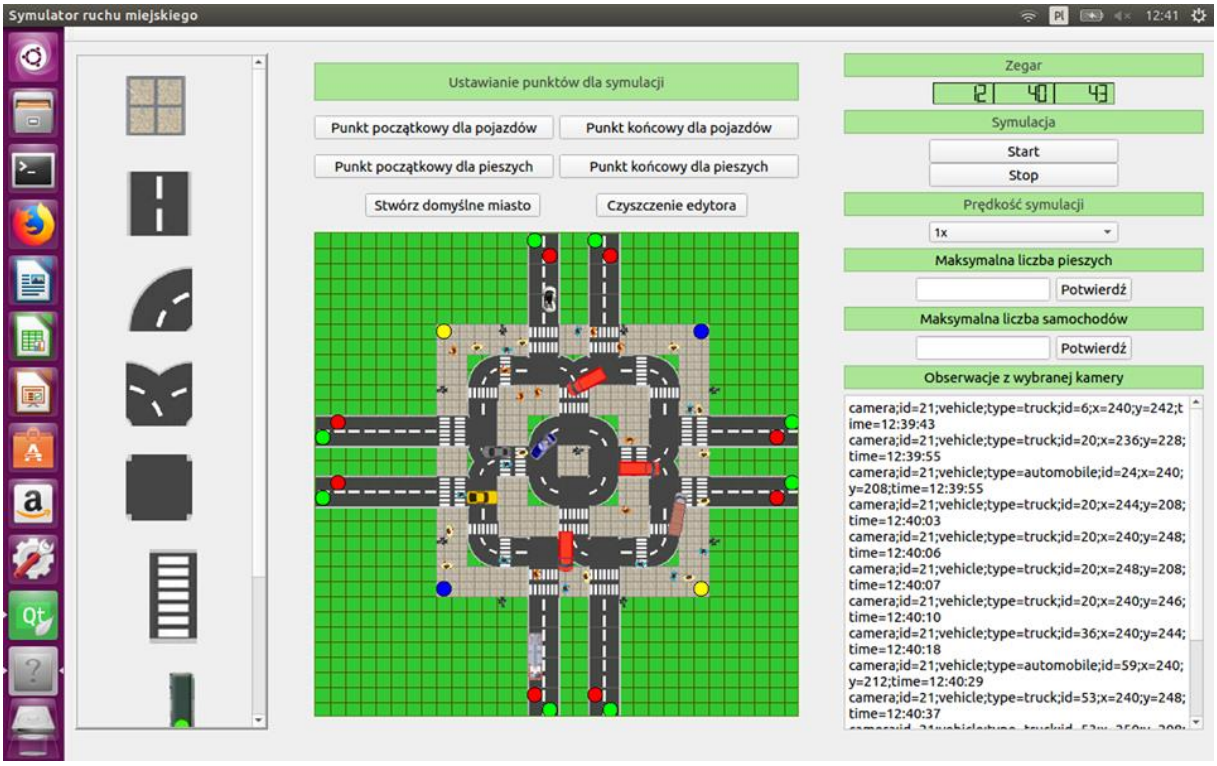
Dokumentacja końcowa

Temat projektu: Symulator ruchu drogowego
Zespół projektowy: Marcin Chrostowski, Paweł Ostaszewski

1. Opis tematu zaczerpnięty ze strony przedmiotu
Zaimplementować prosty symulator ruchu miejskiego. Za jego pomocą powinno być możliwe zamodelowanie kilku ulic po których powinny poruszać się samochody (min. 2 rodzaje: mniejsze, większe) oraz piesi. Aktualną sytuację należy w sposób prosty zwizualizować (np. koło symbolizuje samochód, a prostokąt pieszego). W różnych miejscach ulicy mogą być rozmieszczone „inteligentne" kamery. Każda z nich ma parametry takie jak dokładność, kąt widzenia, kierunek obserwacji. Kamery powinny być próbkowane co 1s. Jeżeli kamera coś „widzi" to generuje obserwacje, gdzie podaje pomierzone współrzędne obiektu.

2. Zastosowane narzędzia
Aplikacja została napisana w języku C++ w środowisku QtCreator i korzysta z biblioteki standardowej C++, biblioteki rozszerzającej C++ - Boost oraz z bibliotek Qt. Aplikacja zawiera plik Makefile, dzięki czemu możliwe jest skompilowanie aplikacji za pomocą polecenia make. Aplikacja tworzona była na systemie operacyjnym Linux Ubuntu 16.04 LTS (Xenial Xerus).

3. Wygląd aplikacji


4. Szata graficzna
Stworzona została głównie za pomocą narzędzia Design oferowanego przez Qt editor. Za jego pomocą możliwe było rozmieszczenie niezbędnych elementów w prosty sposób oraz nazwanie ich i nadanie podstawowych własności takich jak np. tekst w wyświetlany w Labelach. Jeżeli była potrzeba aby odwołać się do jakiegoś obiektu, należało to zrobić po nadanej mu nazwie.

5. Struktura aplikacji

Główną klasą odpowiedzialną za wyświetlanie wszystkich niezbędnych elementów graficznych jest MainWindow w której to bezpośrednio odwołuje się do poszczególnych obiektów graficznych umieszczonych za pomocą Designera. W tej klasie znajdują się metody obsługujące aktualizujące to co jest wyświetlane na zegarze oraz polu odpowiedzialnym za obserwacje z wybranej kamery. Istnieją również metody emitujące odpowiednie sygnały po kliknięciu na przyciski wstawiania punktów początkowych oraz końcowych dla pieszych i pojazdów.

Klasy CityConstructorWidget oraz CityTrafficSimulatorWidget odpowiadają za część powiązaną z dodawaniem nowych elementów do obszaru symulacji. Odpowiadają za dodanie odpowiednich obrazków do lewego panelu z którego wybiera się obiekty, które po przetransportowaniu powinny trafić na obszar symulacji. Dbają o odpowiednie reagowanie na kliknięcia użytkownika na te obrazki.

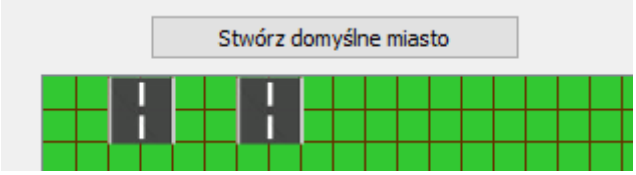
Większość logiki aplikacji zawarta jest w klasie Controller. Odpowiada ona za prawidłowe funkcjonowanie włączonej symulacji oraz za umieszczanie obiektów na obszarze symulacji. Poza tym reaguje na kliknięcia na niektóre obiekty, np. usuwa daną część symulacji po kliknięciu na nią prawym przyciskiem myszy. Znajduje się tutaj sposób w jaki poruszają się pojazdy, w jaki sposób wybierają dalszą trasę. Klasa odpowiada również za generowanie nowych uczestników ruchu w wyznaczonych miejscach z zachowaniem podstawowych zasad poruszania się po ulicy. Klasa komunikuje się również z innymi klasami przekazując i pobierając niezbędne informacji np. przekazuje do MainWindow dane potrzebne do wyświetlenia obserwacji z wybranej kamery. Najczęściej komunikacja ta występuje za pomocą emisji odpowiednich sygnałów i poinformowaniu danego slotu o jego wystąpieniu, po czym dany slot (metoda) zostaje wykonana.

Istnieje również hierarchia klas, która odpowiada wszystkim obiektom, które można umieścić na planszy. Wszystkie obiekty dziedziczą od CityObject. Z tej klasy dziedziczą dwie – StreetElement oraz RoadUser. Pierwsza z nich jest podstawą dla wszystkich klas reprezentujących elementy, które pobiera się z lewego panelu aplikacji, czyli np. ulice. Z klasy RoadUser dziedziczą klasy Car oraz Pedestrian, którzy odpowiadają uczestnikom ruchu.

6. Sposób działania aplikacji i funkcjonalności

Po włączeniu aplikacji powinno pojawić się przed nami ekran przedstawiony na początku dokumentacji. W tym momencie możemy zdecydować co chcemy zrobić. Jeżeli nie chcemy samemu tworzyć miasta istnieje przycisk stworzenia przykładowego miasta, który buduje już wcześniej przygotowane wzorcowe miasto, po którym po wciśnięciu przycisku odpowiadającego za start aplikacji będą poruszały się obiekty.

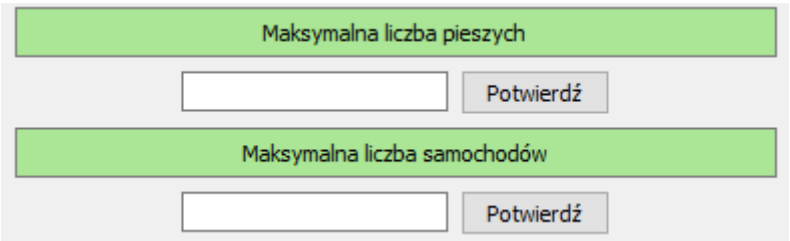
Jeżeli chcemy samemu stworzyć miasto albo dobudować elementy do już wcześniej stworzonego, należy z lewego panelu należy wybrać interesujący nas obiekt, a następnie przenieść go na plansze. Obiekty o rozmiarze 2x2 można umieścić jedynie co dwie kratki, tj. np. licząc od górnej lewej na kratce pierwszej i drugiej tak, ale na drugiej i trzeciej nie. Poprawne rozmieszczenie obrazuje obrazek poniżej. Jest to wycinek lewego, górnego fragmentu mapy. Aplikacja sama nie pozwoli umieścić obiektów w złych miejscach.



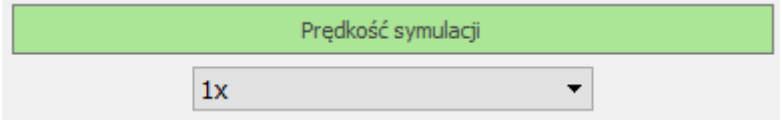
Po stworzeniu makiety miasta należy z górnego części aplikacji wybrać w dowolnej kolejności przyciski odpowiadające za rozmieszczenie punktów początkowych i końcowych i następnie klikając lewym przyciskiem myszy rozmieścić je w wybranych przez nas miejscach na mapie. Zdjęcie poniżej przedstawia przyciski.

Punkty początkowe i końcowe dla pojazdów można rozstawiać jedynie na prostych ulicach w odpowiednich miejscach. Przykład prostej ulicy znajduje się na obrazku przedstawiającym prawidłowe rozmieszczenie elementów 2x2. Kiedy skończymy, należy kliknąć jeszcze raz na ostatnio wybrany punkt aby odznaczyć ustawianie i następnie możemy uruchomić naszą symulację. Służą do tego przyciski start oraz stop.

Zaraz po kliknięciu przycisku start powinna uruchomić się symulacja. Przycisk stop wstrzymuje całą symulację. Obiekty są losowo generowane z podstawowych punktów i losowo wybierają kierunek poruszania się. Samochody ruszają się tylko po odpowiednich dla siebie pasach z uwzględnieniem prawostronnego poruszania się. Jeżeli pojazd nie ma gdzie jechać, to zatrzymuje się. Piesi mogą poruszać się po pasach i po całym dostępnym dla nich chodniku, tj. mogą poruszać się w każdą stronę, jeżeli jest tam chodnik lub pasy. Możliwe jest również ustawienie konkretnej ilości pieszych oraz samochodów, która od momentu zaakceptowania potwierdzenia nie będzie przekraczana. Póki w symulacji znajduje się więcej lub tyle samo uczestników ruchu co wskazana wartość, to nie będą dodawani nowi. Jeżeli aktualnie w symulacji jest więcej niż wskazana liczba, to pozostaną w niej do momentu dojechania do punktu końcowego. Zmianę liczby samochodów można dokonać w miejscu przedstawionym na zdjęciu poniżej.



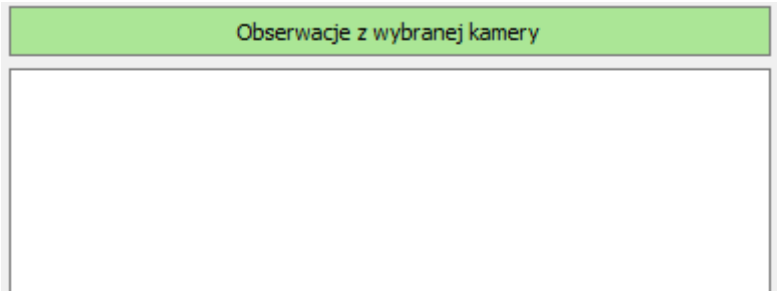
Możliwa jest również zmiana prędkości symulacji.



Po zmianie prędkości poza prędkością symulacji, zwiększa się również prędkość z jaką mija czas, który pokazany jest na zegarze w prawym górnym rogu.

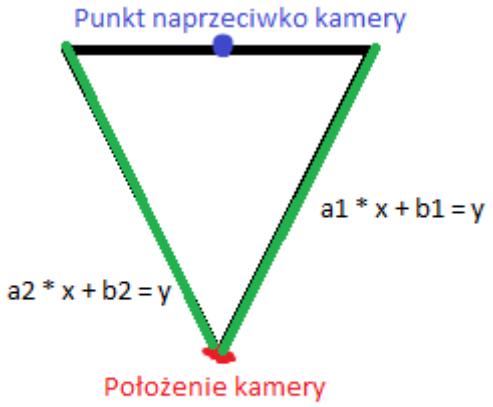
Zegar pierwotnie synchronizuje się z czasem systemowym.

Jeżeli chcemy zobaczyć co takiego obserwuje postawiona przez nas kamera należy kliknąć na nią lewym przyciskiem (upewnić się, że nie mamy wciśniętego przycisku odpowiedzialnego za stawianie punktów początkowych lub końcowych). W panelu po prawej powinny pojawiać się kolejne obserwacje, w których znajduje się id kamery, typ obserwowanego obiektu, jego położenie, id obiektu oraz czas obserwacji. Obserwacje oddzielone są średnikiem.



Poza wyświetlaniem obserwacji w przedstawionym wyżej polu, obserwacje ze wszystkich kamer zapisywane są w pliku Kamery.txt, który znajduje się z folderze, w którym dokonał się build aplikacji. Jako, że zapisywane dane mają wspólny format, można zapisać je pobrać i zapisać w wybranej przez użytkownika bazie danych.

Sama mechanika obserwacji kamer opiera się na porównywaniu tego, czy współrzędne obiektu znajdują się pomiędzy punktem położenia kamery, a punktem znajdującym się na przecięciu wysokości trójkąta i boku trójkąta naprzeciwko punktu położenia kamery. Badane jest też, czy punkt znajduje się między dwoma prostymi ograniczającymi pole widzenia kamery. Kamery mają zdefiniowany w kodzie kąt widzenia, zasięg oraz wyznaczany ze względu na położenie kierunek obserwacji. W trakcie tworzenia kamery wyliczane są współrzędne a i b prostych tych boków trójkąta, które wychodzą z punktu położenia kamery



do sprawdzenia, czy punkt znajduje się między dwoma prostymi. Sprawdza się to podstawiając odpowiednio x lub y punktu do równania w zależności od kierunku obserwacji kamery i porównuje się z niepodstawianą współrzędną. Rysunek obrazujący:

7. Napotkane trudności

W trakcie realizacji projektu natknęliśmy się na pewne trudności, z którymi należało się zmierzyć. Główną był brak czasu potrzebnego na zaimplementowanie wszystkich przepisów ruchu drogowego, przez co niestety byliśmy zmuszeni zrezygnować z tego pomysłu. Problemem było również poruszanie się obiektów, które również powodowało sporo problemów. Wprowadzone zostało uproszczenie polegające na tym, że ulice 2x2 można stawiać tylko w opisany w punkcie 6 sposób. Obiekty poruszają się też losowo. Sporo czasu zajęło również znalezienie i przetestowanie rozwiązania dla obserwacji kamer, co mocno wydłużyło całą implementację.

8. Podsumowanie

Projekt okazał się bardziej czasochłonny i pracochłonny niż pierwotnie zakładaliśmy. Poniżej przedstawiamy tabelę porównującą przybliżone czasy planowane i rzeczywiste realizacji poszczególnych części projektu.

Zadanie	Czas wykonania w godzinach planowany	Czas wykonania rzeczywisty
Stworzenie klasy uczestnika ruchu i klas dziedziczących po niej	3	5
Stworzenie klasy świateł drogowych	1	-
Stworzenie klasy kamer	2	6
Podłączenie licznika czasu	1	4
Synchronizacja licznika czasu ze zdarzeniami zachodzącymi w aplikacji	4	4
Implementacja podstawowych zasad ruchu drogowego	3	3
Stworzenie funkcji odpowiadających za losową zmianę trasy uczestnika ruchu, zgodną z przepisami ruchu drogowego.	2	6
Synchronizacja zmiany kierunku ruchu uczestnika ruchu drogowego, żeby np. dwa samochody nie zderzyły się ze sobą	3	-
Stworzenie mechanizmu konstrukcji ulic	4	6
Stworzenie głównego okna aplikacji	1	1

Stworzenie okna, w którym będą wyświetlać się informacje o obiektach, jakie zaobserwowała wybrana kamera	1	1
Oprogramowanie przycisków	2	4
Przypisanie kształtów do uczestników ruchu	1	2
Stworzenie animacji ruchu	6	7
Całkowity czas	**34**	**49**