**(Średnio)zaawansowane programowanie w C++ (ZPR2012L)**

**Dokumentacja funkcjonalna**

1. **Temat:**

Implementacja prostego symulatora ruchu miejskiego.

1. **Opis docelowego modelu:**

Podczas symulacji poruszają się obiekty różnego typu: samochody małe, duże oraz piesi. W różnych miejscach ulic rozmieszczone są "inteligentne" kamery – każda posiada swoje parametry: dokładność, kąt widzenia, kierunek obserwacji. Kamery są próbkowane co 1 sek. Jeżeli coś zauważy – generuje obserwacje i podaje współrzędne zaobserwowanych obiektów zmierzone zgodnie ze swoimi parametrami.

1. **Docelowa funkcjonalność:**
2. Pobieranie danych z plików konfiguracyjnych XML:
   * Mapa ulic w postaci węzłów grafu (w celu ich wizualizacji)
   * Trasy samochodów wraz z ich typami w postaci kolejnych punktów przejazdu
   * Trasy pieszych w postaci kolejnych punktów ścieżki
   * Współrzędne kamer i ich parametry
3. Samochody różnią się od siebie parametrami (prędkość maksymalna, masa), rozpędzają się i zwalniają przed zakrętami (zależnie od promienia skrętu) – prosta fizyka
4. Piesi chodzą po liniach prostych, skręcają 'w miejscu'. Mogą poruszać się po całej mapie
5. Samochody skręcają po łukach – symulator automatycznie wyznacza płynne trasy przejazdu
6. Kamery mają swoje parametry: dokładność, kąt widzenia, kierunek obserwacji
7. Dokładność obserwacji uzależniona jest od parametrów kamery
8. Dane z kamer są logowane do pliku gdy podczas próbkowania w ich zasięgu widzenia znajdują się jakieś obiekty
9. Prosta wizualizacja graficzna symulacji w postaci obiektów geometrycznych
10. Możliwość zapętlenia symulacji – obiekty w kółko poruszają się po swoich trasach
11. Menu pozwalające ustawić parametry początkowe symulacji – wskazanie plików konfiguracyjnych lub bezpośrednie wprowadzenie danych
12. Ze względu na ograniczenia czasowe:
    * Nie zakładamy kolizji obiektów – samochody i piesi przenikają się
    * Brak zaawansowanego GUI
13. **Funkcjonalność dodatkowa:**
14. Generowanie *n* pieszych i *m* pojazdów poruszających się z wylosowanych miejsc i losowo skręcających na skrzyżowaniach
15. GUI pozwalające ustalić trasy obiektów oraz położenie kamer za pomocą myszy bezpośrednio na mapie
16. Weryfikacja poprawności wprowadzonych danych (pokrycie na mapie)
17. Sprawdzanie poprawności 'wyklikanych' ręcznie danych
18. **Wstępna organizacja projektu:**

Aplikacja składać się będzie z kilku głównych wątków (zostanie podzielona na moduły). Wątek modelowanej symulacji, GUI, timer czuwający nad częstotliwością obliczeń modelu oraz próbkowaniem kamer, logger odpowiedzialny za generowanie do pliku danych z kamer.

Na początku działania aplikacji przewidujemy wyświetlenie prostego menu pozwalające ustawić parametry działania oraz dodać obiekty do symulatora – wstępnie w postaci tekstowej. W przypadku wolnych zasobów czasowych w bardziej przyjaznej postaci graficznej.