WOJSKOWA AKADEMIA TECHNICZNA

**im. Jarosława Dąbrowskiego**

#### WYDZIAŁ CYBERNETYKI



**Programowanie zdarzeniowe**

Dokumentacja projektowa

Spis treści

[1. Wstęp 3](#_Toc536042282)

[2. Wybór środowiska oraz technologii 4](#_Toc536042283)

[3. Idea działania aplikacji, zastosowanie, ograniczenia 4](#_Toc536042284)

[4. Przypadki użycia (Use-case) 6](#_Toc536042285)

[5. Architektura aplikacji 7](#_Toc536042286)

[6. Diagramy sekwencji 8](#_Toc536042287)

[7. Testowanie 10](#_Toc536042288)

[8. Zakończenie 13](#_Toc536042289)

# Wstęp

Dokumentacja dotyczy projektu „RecoilAssist”, który jest realizacją zadania projektowe o temacie „System korygujący ruchy myszki w celu skorygowania skupienia strzału w grach FPS”. Projekt ma celu wytworzenie oprogramowania **w celach edukacyjnych** które ma na celu wspomaganie gracza w grach typu FPS w zakresie korygowania odrzutu broni podczas strzelania ogniem ciągłym.

Dokumentacja projektowa jest równocześnie sprawozdaniem z zadania projektowego jak i dokumentacją techniczną wytworzonego oprogramowania.

W rozdziale drugim zostały przedstawione i uzasadnione wybory na poziomie planowania, takie jak wybór technologii, środowiska, rozwiązań itp.

W rozdziale trzecim opisana została idea działania aplikacji wraz z zarysem modelu konstrukcyjnego.

Trzy następne rozdziały (czwarty, piąty i szósty) dotyczą modelowania projektu i opisują projekt z kolejno następujących perspektyw:

* funkcjonalnej (przypadki użycia);
* architektury;
* procesowej (diagramy sekwencji).

Rozdział siódmy poświęcony jest testowaniu aplikacji. Projekt nie zawiera testów jednostkowych, ale zostanie zaprezentowany scenariusze w jaki sposób testowana była aplikacja, oraz ich rezultaty. Dodatkowo w rozdziale została umieszczona szczegółowa prezentacja działania aplikacji.

W rozdziale ósmym są wysunięte wnioski z projektu, oraz zaprezentowane możliwe rozwinięcia projektu.

# Wybór środowiska oraz technologii

Aplikacja została zaimplementowana w oparciu o [platformę .NET](https://dotnet.microsoft.com). Wybór ten jest głównie uzasadniony pewnym doświadczeniem, który przekłada się na jakość oprogramowania, oraz czas implementacji. Dodatkowo dla platformy .NET oraz SO Windows dostępne jest wysokiej jakości darmowe środowisko Microsoft Visual Studio w wersji Community, które zostało wykorzystane do zbudowania aplikacji. Tutaj pod tym względem jedynym realnym konkurentem byłby środowisko IntelliJ IDEA, który posiada darmową wersję niekomercyjną dla celów edukacyjnych. Możliwości obu technologii w zakresie wymagań projektów wyglądają podobnie, więc wybrana została platforma .NET, tak jak zostało to wspomniane wyżej, z powodu znajomości tej technologii.

Projekt został zbudowany w języku C# w zgodności z .NET Framework w wersji 4.7.1 (najnowszej w momencie budowania projektu). Projekt został tak skonfigurowany aby mógł być odpalany w trybie konsolowym jak i zarówno okienkowym.

Graficzny interfejs został zaimplementowany w oparciu o bibliotekę WindowsForms. Technologia ta aktualnie jest uznana za przestarzałą, ale prostota oraz szybkość implementacji interfejsu jednak zdecydowała o wybraniu tej technologii, szczególnie dlatego że w projekcie zaistniała potrzeba zaimplementowania tylko jednej formatki.

Narzędzie ORM ([object-relational mapping](https://en.wikipedia.org/wiki/Object-relational_mapping) tool) wykorzystane w tym projekcie to [Entity Framework 6](https://docs.microsoft.com/pl-pl/ef/ef6/). Jest to najpopularniejsze narzędzie ORM wykorzystywane w połączeniu z .NET Framework. Tutaj Microsoft oficjalnie zachęca migrować rozwiązania zbudowane w oparciu o EF6 na Entity Framework Core, głownie z powodu nowszego „lepszego” kodu, aczkolwiek podane zalety nie są istotne w projekcie i tutaj zdecydowanie EF6 wydaje się najlepszym wyborem

# Idea działania aplikacji, zastosowanie, ograniczenia

Celem aplikacji jest redukcja odrzutu broni w grach podczas strzelania ogniem

ciągłym. Tutaj na pewno warto byłoby zbudować aplikację która jest wygodna w użyciu, uniwersalna – możliwa do uruchomienia dla każdej gry, niewykrywalna przez gry, oraz dokładna. Niestety ale wytworzenie takiej aplikacji wiązałoby się z implementacją zaawansowanej i rozbudowanej aplikacji lub całego systemu z rozbicie na aplikacji kliencie i maszyny uczące się. Dlatego aby projekt był możliwy do wykonania przy dostępnych możliwościach należy okroić wymagania, tak aby spełniała swój główny cel (redukcja odrzutu) i w miarę możliwości spełniała część z wymienionych kryteriów jakościowych.

Dodatkowo poza wymaganiami jakie zostały nałożone na aplikację, należy spełnić cel ćwiczenia jakim jest wykorzystanie mechanizmów programowania zdarzeniowego w praktyce. W tym celu aplikacja powinna wykorzystywać przeróżne mechanizmy programowania zdarzeniowego takie jak: wątki, zdarzenia, strumienie, itp.

Głównymi sposobami na redukcję odrzutu jest modyfikacja działania aplikacji, lub kontrolowanie zachowania myszki na poziomie sprzętowym. W przypadku pierwszego rozwiązania możliwymi podejściami są:

* Aplikacja działająca w tle (np. Cheat Engine);
* „Łatka” do gry, tzn. zmodyfikowana aplikacja, która poprzez mechanizmy inżynierii wstecznej modyfikuje działanie aplikacji;
* Modyfikacja do gry (o ile gra udostępnia taką możliwość), która poprzez udostępniony interfejs koryguje odrzut.

To rozwiązanie raczej odbiega od tematu przedmiotu, oraz wystawia użytkownika na zagrożenie wykrycia przez mechanizmy zwalczające oszustów.

Pozostaje więc korygowanie ruchu myszki poprzez poziom sprzętowy. Tutaj pojawiają się problemy takie jak:

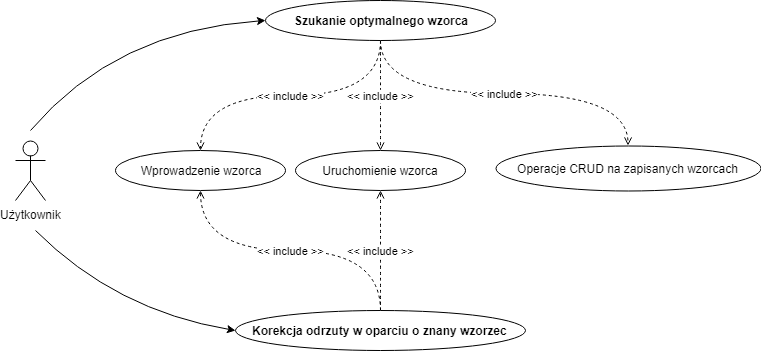
* Kiedy korygować ruch myszki, skoro nie wiemy kiedy gracz strzela;
* Jak korygować ruch, skoro każda broń w każdej grze ma inne parametry;
* Jak obliczyć drogę jaką ma pokonać myszka;

Niestety przy obecnych możliwościach program docelowy nie jest w stanie rozwiązać tych problemów, ale jest możliwość obejścia tych problemów. Użytkownik z wykorzystaniem utworzonych wzorców zachowania myszki, będzie uruchamiał program który przy każdym przytrzymaniu lewego przycisku myszy będzie korygował jej ruch. Oprócz delikatnej niewygody jakim jest ruch myszki przy dłuższym kliknięciu niezależnie od tego czy gracz strzela czy nie, użytkownik musi pozyskać wzór na ruch myszki. Jest to raczej trudne zadanie i pewnie sprowadza się do heurystycznych algorytmów.

Aplikacja powinna być wygodna w dwóch przypadkach użycia: testowania wzorców w celu oceny jakościowej takiego wzorca, oraz użycie już znanego wzorca do praktycznego wykorzystania.

# Przypadki użycia (Use-case)

W aplikacji można wyróżnić dwa przypadki użycia:

* Szukanie optymalnego wzorca
* Korekcja odrzuty w oparciu o znany wzorzec

**Szukanie optymalnego wzorca** – użytkownik chce otrzymać wzór który wspomoże go podczas gry. W tym celu na podstawie informacji zdobytych lub własnego przeczucia wprowadza do aplikacji wzorce które następnie ocenia na aplikacji docelowej (grze) biorąc pod uwagę dokładność takiego wzorca.

**Korekcja odrzuty w oparciu o znany wzorzec –** użytkownik posiada już wzorzec, którego chce wykorzystać. Wzorzec jest wprowadzany, lub jest ładowany do aplikacji poprzez skrypt lub z bazy danych. Następnie automatycznie odrzut jest korygowany aż do wyłączenia aplikacji.

Oczywiście oba przypadki będą korzystały z tych samych mniejszych przypadków użycia a więc należy je wyróżnić:

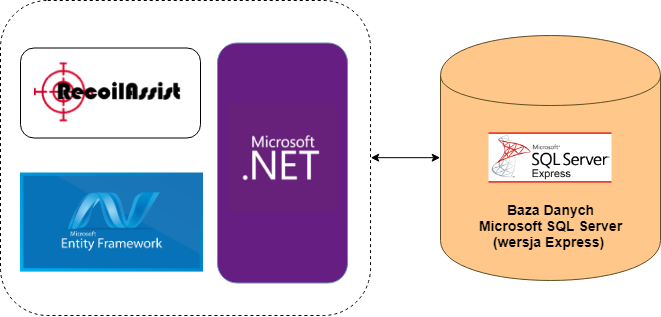
**Wprowadzenie wzorca –** użytkownik wprowadza wzorzec lub wykorzystywany jest aktualny wzorzec z bazy danych.

**Uruchomienie wzorca** – uruchomienie funkcjonalności reagującej na wciśnięcie przycisku myszy i skorygowanie strzału w oparciu o wprowadzony wzorzec.

**Operacje CRUD na zapisanych wzorcach** – w przypadku korzystania z BD, należy mieć możliwość wprowadzania nowych, odczytywania, zapisywania i usuwania wzorców z BD.

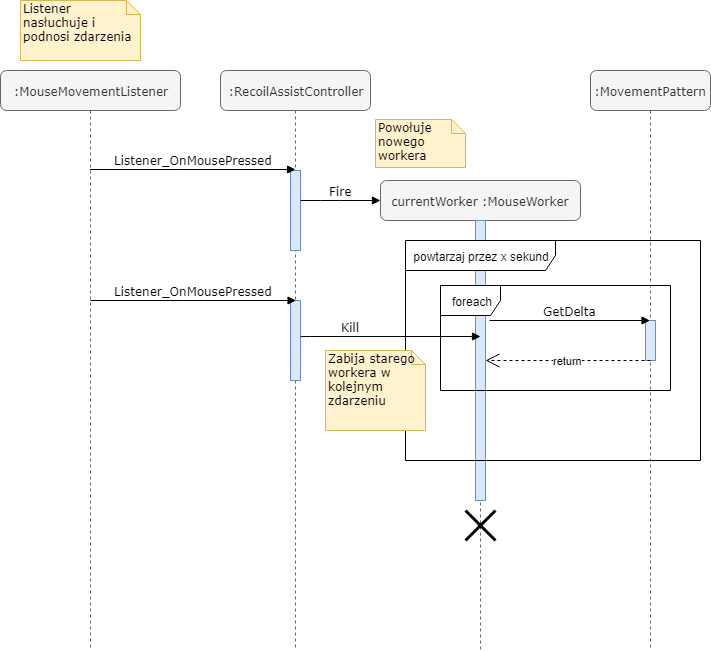
# Architektura aplikacji

Aplikacja bezpośrednio korzysta z bibliotek .NET Framework, oraz Entity Framework 6. Dodatkowo aplikacja komunikuje się z bazą danych SQL Server Express. Architektura aplikacji przedstawiona jest na rysunku.

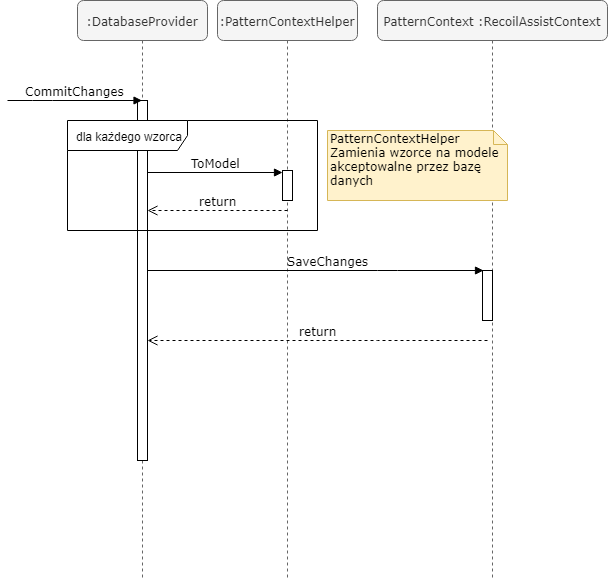


# Diagramy sekwencji

Tutaj najważniejszym procesem będzie uruchamianie nowych wątków korygujących ruch myszki. Diagram przedstawiający ten proces przedstawia poniższy rysunek:

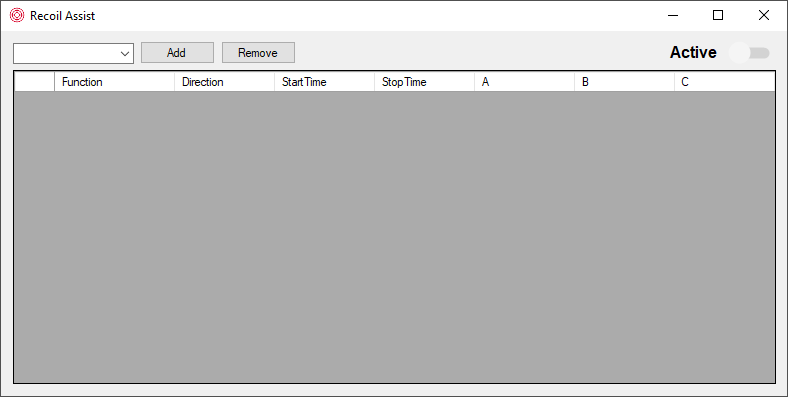


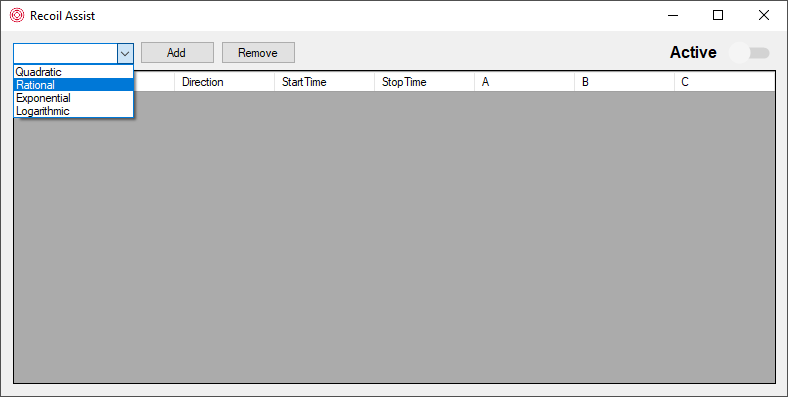
Podczas zapisu wzorców do bazy wykorzystywany jest model wzorca przechowujący dane potrzebne do odtworzenia takiego wzorca. Zamiana jest dokonywana „On demand” czyli na żądanie. To znaczy, że wzorce są zamieniane na model i na odwrót dopiero gdy zaistnieje taka potrzeba. Można to zauważyć podczas wywołania metody *CommitChanges*() klasy *DatabaseProvider*, jej przebieg przedstawia następujący diagram:

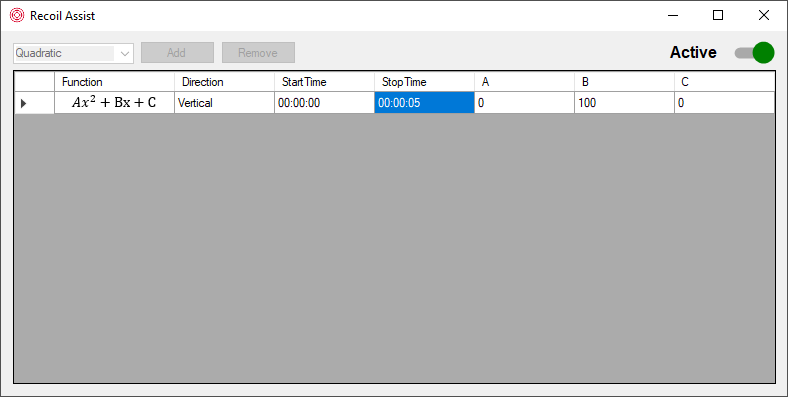


# Testowanie

Poprawność działania aplikacji była oceniania przez poniższy schemat przebiegu wydarzeń. Jeżeli przebieg wydarzeń przeszedł pomyślnie, to można wywnioskować że ten przepływ jest poprawny.

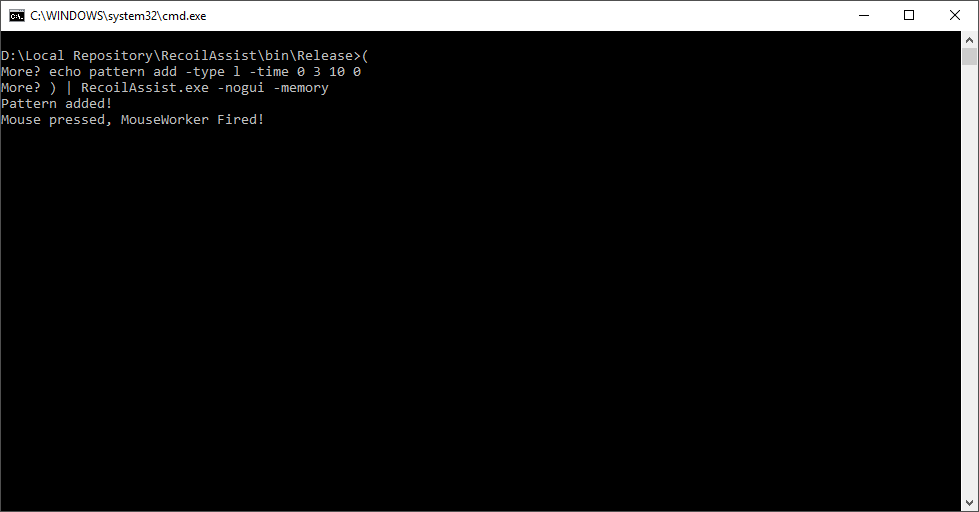
**Uruchamiamy aplikację w trybie GUI** – tryb GUI jest trybem domyślnym i powinien wystartować po uruchomieniu pliku exe (lub debuggera). Okienko wygląda następująco: 

**Wprowadzamy wzorzec** – wzorzec wprowadza się poprzez wybranie typu wzorca, kliknięciu przycisku „Add”, oraz uzupełnieniu danych w tabelce: 

**Przełączamy aplikację w tryb aktywny** – można tego dokonać poprzez kliknięcię w kontrolkę po prawej stronie: 

**Sprawdzamy korektę** – po wciśnięciu lewego przycisku myszki, kursor myszki powinien się zachować zgodnie z wzorcem: 

Jeżeli powyższy przebieg wydarzeń ukończy się pomyślnie, można go zmodyfikować o różne rodzaje funkcji, oraz ich zestawienie. Dodatkowo możemy testować elementy GUI, czyli np. klikać w niespodziewany sposób na różne elementy GUI, dodawać, usuwać wzorce itp.

Aplikację można również uruchomić w trybie konsolowym, w takim wypadku należy uruchomić aplikację z komendą **-nogui**. Można to połączyć z skryptem i od razu wprowadzić wzorce:

# Zakończenie

Aplikacja nie jest zbyt praktyczna ani wygodna, aczkolwiek przy atmosferze zagorzałej konkurencji mogłaby znaleźć zastosowanie, co można wywnioskować bo zachowaniu graczy do tej pory. Przykładem może być ostatnie wydarzenia związane z nowym sposobem oszukiwania w Battlefield 5. Gracze modyfikowali ustawienia graficzne tak aby poziom dokładności modeli (ilość wielościanów na model) był na najniższym poziomie niezależnie od odległości od widza, dzięki czemu gracze mogli łatwiej wypatrzeć wroga. Nie jest to rozwiązanie wygodne, gdyż gra wygląda bardzo nieatrakcyjne, ale potrzeba uzyskania nawet niewielkiej przewagi nad przeciwnikiem jest silniejsza.

Projekt oczywiście nie będzie rozwijany, ale krytycznie patrząc na aplikację można by ją rozwinąć o:

* Możliwość tworzenia skryptów z poziomu GUI
* Możliwość zapisywania zestawów skryptów, oraz możliwość szybkiego przełączania się pomiędzy nimi
* Skróty klawiszowe
* Sparować wzorce (lub zestawy) z użytkownikami, dzięki czemu każdy użytkownik by miał swoje wzorce
* Wzbogacić tryb konsolowy o dodatkowe komendy i argumenty
* Zaimplementować algorytm heurystyczny, który by szukał wzorców optymalnych (lub przybliżonych)