

Projekt z PO - Speedcubing Timer - Dokumentacja

Artur Jankowski










1 Motywacja

Celem projektu było stworzenie aplikacji okienkowej do Speedcubingu (tj. układania kostki Rubika na czas).

2 Użytkowanie

2.1 Instalacja

Rysunek 1: Przykładowy wygląd folderu z programem

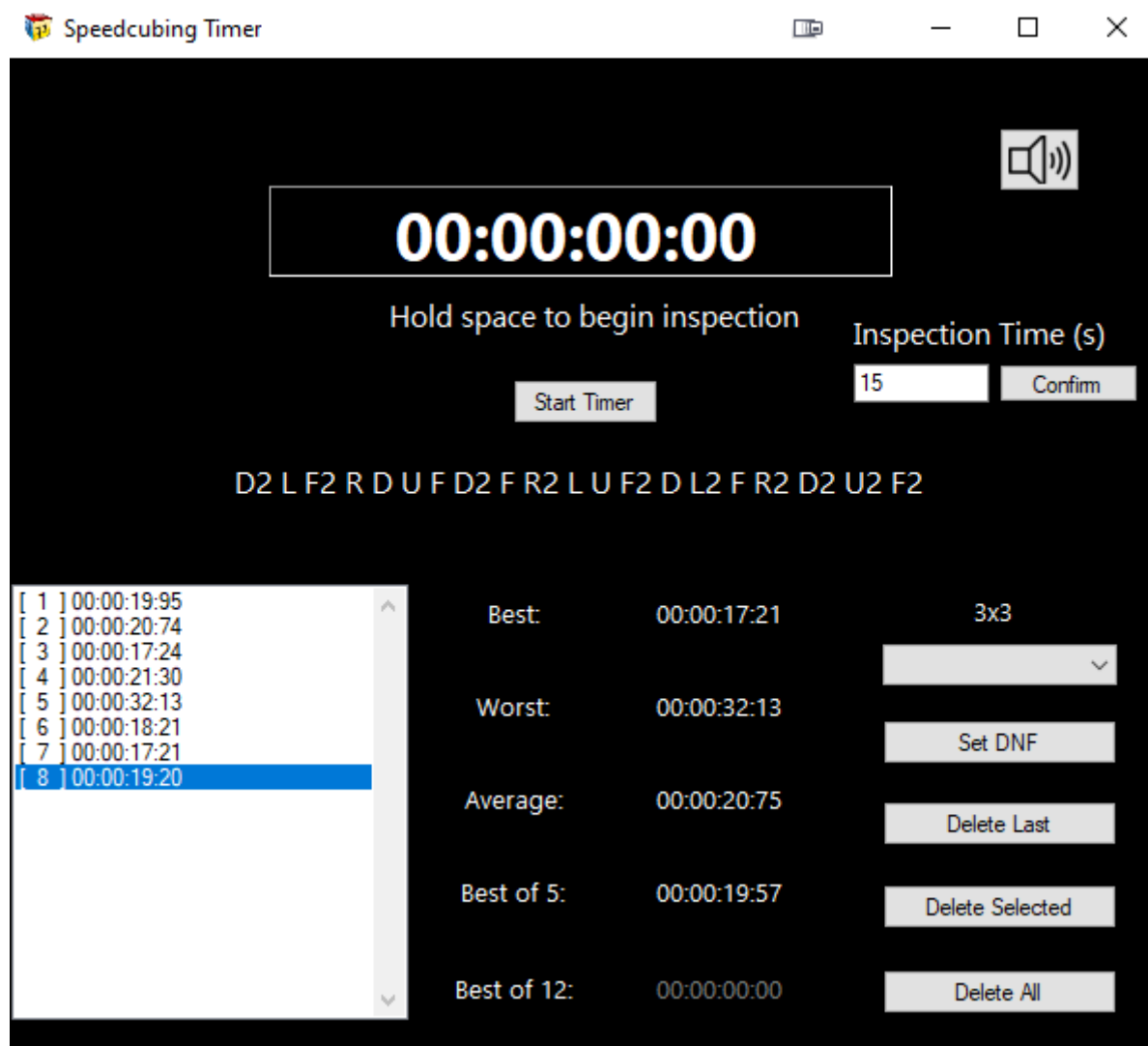
Nazwa	Typ	Kozmiar
 InspectionEndSound	Plik WAV	46 KB
 InspectionSound	Plik WAV	93 KB
 SolveEndSound	Plik WAV	279 KB
 Solves	Plik wartości oddz...	1 KB
 SpcTimer	Aplikacja	157 KB
 SpcTimer.exe.config	XML Configuratio...	1 KB
 SpcTimer.pdb	Program Debug D...	54 KB
 TimerLibrary.dll	Rozszerzenie aplik...	20 KB
 TimerLibrary.pdb	Program Debug D...	88 KB

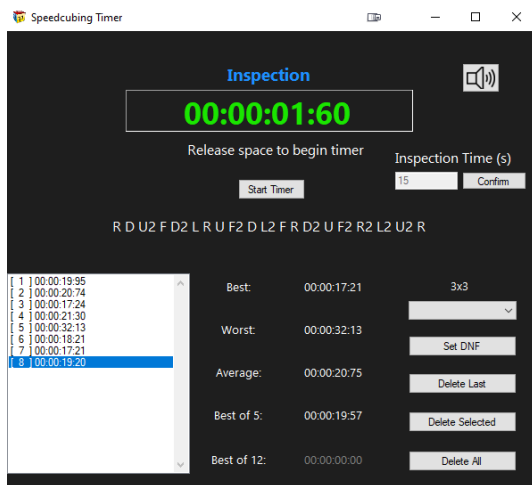
Wystarczy wypakować pliki do wybranego folderu, i uruchomić SpcTimer.exe. Wyniki będą zapisywane w pliku Solves.csv(który utworzy się automatycznie). Jeśli będzie brakować któregoś z plików dźwiękowych, to po prostu w programie nie będą działać dźwięki.

2.2 Używanie programu

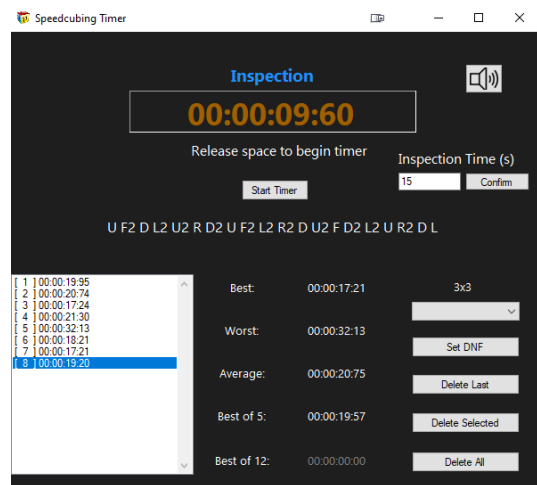
Aplikacja w trybie oczekiwania pokazuje obecnie losowe pomieszenie, daje możliwość wyłączenia/włączenia dźwięku, zmiany obecnie układanej kostki i interakcji z zapisanymi statystykami. Program na bieżąco liczy statystyki (kiedy to możliwe).

Rysunek 2: Główne okno programu w trybie oczekiwania





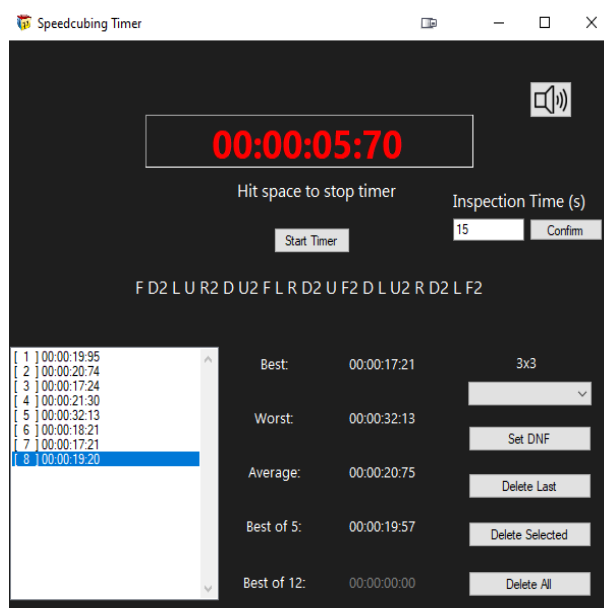
(a) początek czasu inspekcji



(b) koniec czasu inspekcji

Po wciśnięciu spacji, aż do jej puszczenia rozpoczyna się inspekcja, program przypomina o upływającym czasie przy pomocy zmieniającego się na czerwony kolor zegara i sygnałów dźwiękowych.

Rysunek 3: Główne okno programu w trakcie układania



Puszczenie spacji rozpoczyna układanie, od tego momentu, aż do ponownego wciśnięcia spacji liczy się czas. Po zakończeniu ułożenia zapisuje się on do bazy i program przechodzi w tryb oczekiwania.

3 Funkcjonalność

3.1 Funkcjonalność gotowa

- Graficzne okno przedstawiające stoper, scramble i najlepsze statystyki
- UI reagujące zmianą koloru kontrolek na różne wydarzenia
- Stoper, obsługiwany wejściem z klawiatury (przez przytrzymanie/puszczenie spacji) lub przez przycisk,
- Możliwość inspekcji (tj. czasu - modyfikowalnego przez użytkownika, na obejrzenie kostki, który nie liczy się do czasu ułożenia i gdy ten czas minie, bez rozpoczęcia stopera przez użytkownika, to ułożenie jest dyskwalifikowane)
- Dźwięk informujący o końcu czasu inspekcji/rozpoczęciu układania/zakończeniu ułożenia. Możliwość wyłączenia dźwięku
- Generator scramble'i, czyli permutacji/pomieszania kostki
- Śledzenie wyników , wraz z liczeniem średniej z ostatnich ułożeń i najlepszych dotychczasowych wyników (tj. Best of 5, Bo12, Bo100)
- Automatyczne zapisywanie wyników do tekstowej bazy danych wraz z możliwością usunięcia pojedynczych zaznaczonych lub ostatniego lub wszystkich rekordów.
- Oznaczenie jako ułożenie zdyskwalifikowane przez użytkownika
- Możliwość trenowania różnych rodzajów kostek (2x2, 3x3, 4x4), z osobnymi statystykami dla poszczególnych z nich.

3.2 Funkcjonalność nie gotowa - plany na dalszy rozwój

- Wizualizacja pomieszania kostki
- Możliwość zmiany kolorów przez użytkownika
- Lepiej działająca responsywność - w szczególności rozszerzanie się UI wraz z oknem
- Rysowanie wykresów wyników w czasie
- Połączenie z bazą SQL

4 Wykorzystane narzędzia

C# z wykorzystaniem WinForms (.Net Framework 4.7.2, Visual Studio 2019), XUnit do testów, GIT do kontroli wersji(<https://github.com/arturJan4/SpeedCubing-Timer>)

5 Wzorce projektowe

- MVC - do rozdzielenia UI (Form jako implementacja interfejsu View), kontrolera (klasa Controller) i modeli (klasy Move, Solve itp.)
- Singleton - chcemy mieć tylko jedną instancję Stopera - Timer, bo ze specyfiki programu musi on być dokładny, a przez to jest zasobożerny
- Strategy - do generowania różnych scramble'i na podstawie danego typu kostki, aby można było łatwo dodawać inne algorytmy mieszające (np. zgodne ze zmieniającymi się regułami oficjalnych zawodów WCA, np. 2x2 nie ma wszystkich tych samych ruchów w scramble'u co 3x3, a 4x4 ma ich więcej). Czyli implementujemy interfejs od generowania scramble'i, a potem zmieniamy strategię scramble'owania według potrzeb.
- Bridge - do łatwego połączenia między bazą SQL, a naszym programem. Nie zaimplementowałem go, dlatego że wymagałoby to dużo więcej pracy w SQL i bazie niż w samym kodzie programu.