Politechnika Śląska w Gliwicach
Wydział Automatyki, Elektroniki i Informatyki
Programowanie Komputerów
Projekt semestralny

Autor	Pych Szymon
Prowadzący	Dr inż. Roman Starosolski
Rok akademicki	2017/2018
Kierunek	Informatyka
Semestr	4
Termin laboratorium	Wtorek 13:45-15:15
Grupa	6
Sekcja	2
Data oddania sprawozdania	27.06.2018

1. Temat

Projekt polegał na stworzeniu bazy danych bez wykorzystania istniejących już systemów bazodanowych, katalogującej stworzenia z gry Pokémon Red Version zwane po prostu Pokémonami.

2. Analiza tematu

Gra, na podstawie której projekt był tworzony, zawiera wewnętrzną bazę danych zwaną "Pokédex", która służy do katalogowania Pokémonów. Ze względu na fakt iż gra była przeznaczona na konsolę Nintendo GameBoy, do kontroli było osiem przycisków, z czego dwa z nich były praktycznie nieużywane.

Projekt został zaplanowany z myślą by przywoływać na myśl oryginał, posiadając czarno-biały interfejs wyglądający podobnie do Pokédexu, sterowany za pomocą nie więcej niż siedmiu klawiszy. Obsługa projektu miała być zbliżona do growego odpowiednika, zapożyczając z niego charakterystyczną przejrzystość i prostotę, włączając do tego dodatkowe możliwości oraz informacje.

Projekt został wykonany przy użyciu języka C++ w IDE Code::Blocks 16.01 oraz Microsoft Visual Studio Community 2015. Użyte dodatkowe biblioteki to:

- **fstream** do zarządzania plikami dołączonymi do bazy
- **sstream** do łatwego przenoszenia danych z pliku do pamięci
- **vector** do organizacji danych w przystępny sposób
- **chrono** oraz **thread** do zatrzymania programu na krótką chwilę
- **regex** do funkcji wyszukującej w danych
- **SFML** do zarządzania oknem oraz widokiem programu

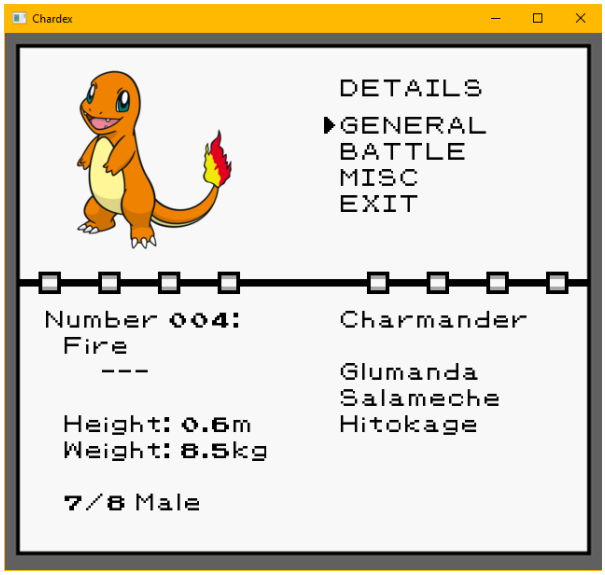
Zastosowane techniki z laboratorium to wyjątki, wzorce, kontenery STL, wyrażenia regularne.

3. Specyfikacja zewnętrzna

Ilustracja 1: Ekran tytułowy Chardexu

Po uruchomieniu się programu, ukazuje się ekran tytułowy, bazowany na ekranie tytułowym z gry Pokémon Red Version. Po wciśnięciu któregokolwiek z klawiszy funkcyjnych ([Z] bądź [X]) użytkownikowi pokazuje się główne menu wraz z listą wprowadzonych Pokémonów. Klawiszem funkcyjnym [Z] wybiera się element na liście, po czym element w menu:

- **VIEW** – zmienia sposób prezentacji listy Pokémonów
- **DATA** – pokazuje szczegółowe dane na temat wybranego Pokémona
- **SEARCH –** pozwala na wyszukiwanie Pokémonów po nazwie bądź za pomocą wyrażenia regularnego, zwraca pełną bazę gdy nic nie znajdzie
- **SORT –** zmienia sposób sortowania listy z numerycznego na alfabetyczny
- **HELP –** włącza krótki tekst oraz instrukcję obsługi
- **ABOUT –** włącza informacje o programie
- **QUIT –** zamyka program

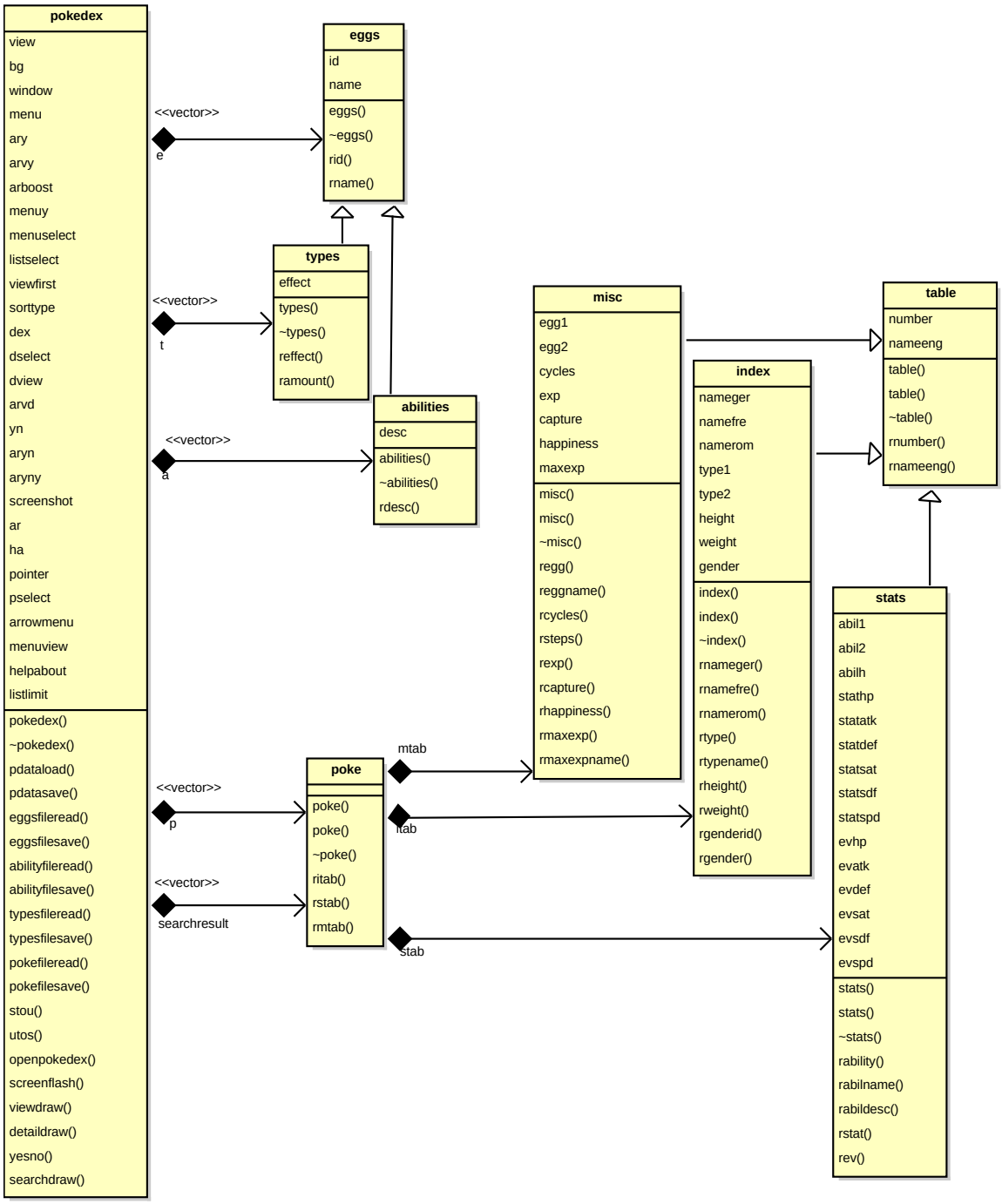


Ilustracja 2: Widok DATA dla Charmandera

Sterowanie:
Klawisze strzałek do wyboru opcji
[Z] do zatwierdzania wyboru
[X] do anulowania wyboru
[Esc] do zamknięcia programu

Program do obsługi wymaga dwóch folderów z plikami, **/img** oraz **/txt**, w którym są rzeczy niezbędne do jego poprawnego działania. W folderze **/txt** są cztery pliki tekstowe **(eggs.txt, ability.txt, types.txt, poke.txt**) zawierające bazy danych wykorzystywane przez program, może się też w przypadku błędu pojawić piąty plik o nazwie **error.txt**, opisujący rodzaj błędu który nastąpił. W folderze **/img** są obrazy wykorzystywane przez program do swojego interfejsu oraz folder **/dex** zawierający obrazy Pokémonów w bazie. W przypadku plików z błędnymi informacjami, program może wyświetlać błędne bądź nawet całkowicie niepasujące informacje, co nie zostanie zarejestrowane w pliku **error.txt**. W takim przypadku trzeba pobrać pliki na nowo.

4. Specyfikacja wewnętrzna



Ilustracja 3: Diagram UML klas Projektu wygenerowany programem Bouml

Klasy oraz ważniejsze pola i metody:

- *eggs* – klasa bazowa do tabel pobocznych oraz tabela dla pliku eggs.txt
- *types* – klasa tabeli dla pliku types.txt
- *abilities* – klasa tabeli dla pliku ability.txt
- *table* – klasa bazowa dla tabel pobocznych z pliku poke.txt
- *index* – klasa tabeli do segregacji podstawowych informacji dla głównej tabeli poke
- *stats* – klasa tabeli do segregacji statystyk dla głównej tabeli poke
- *misc* – klasa tabeli do segregacji pozostałych informacji dla głównej tabeli poke
- *poke* – klasa tabeli dla pliku poke.txt
- *pokedex* – klasa zarządzająca oknem i wszystkimi zebranymi danymi
    - vector<eggs> **e** – tabela z pliku eggs.txt
    - vector<abilities> **a** – tabela z pliku ability.txt
    - vector<types> **t** – tabela z pliku types.txt
    - vector<poke> **p** – tabela z pliku poke.txt
    - sf::RenderWindow **window** – okno programu
    - char **view** – zmienna wskazująca na rodzaj widoku
    - vector<poke>* **pointer** – wskaźnik na tabelę **p** bądź na wyniki wyszukiwania
    - bool **pdataload()** – metoda wczytująca wszystkie potrzebne dane
    - void **pdatasave()** – metoda zapisująca dane z powrotem do pliku
    - void **openpokedex()** – metoda zawierająca główne instrukcje oraz game loop programu
    - void **viewdraw()** – metoda rysująca główne menu programu
    - void **detaildraw()** – metoda rysująca menu danych szczegółowych
    - void **searchdraw()** – metoda wchodząca w mniejszą pętlę z wpisywaniem tekstu

Program najpierw wczytuje wszystkie dane z pliku, po czym wchodzi w główną pętlę, w której rodzaj widoku jest zależny od zmiennej view. Główna pętla zawiera w sobie dwie części – rysowanie oraz reagowanie na zdarzenia. Wewnątrz obydwu jest instrukcja switch decydująca o tym które z widoków jest aktualnie na ekranie i jak ma program reagować na użytkownika.

Program jest opatrzony funkcjami zgłaszającymi wyjątki w przypadku błędu wczytania któregoś z istotnych plików tekstowych bądź obrazów. W takim przypadku program tworzy plik error.txt i dodaje do niego stosowną informację.

Został zastosowany w programie wzorzec string **tostring**(cont T& t) w celu zamiany różnych typów danych na typ string, co jest wymagane przy tak dużej ilości zmiennych typu int bądź float w zbiorze danych.

W celu łatwej organizacji danych, w dużej ilości są utylizowane kontenery STL w postaci wektorów, do których są dodawane obiekty typu *eggs, abilities, type* oraz *poke*.

By ułatwić odnajdywanie grup bądź poszczególnych elementów, funkcja wyszukująca wspiera wyrażenia regularne wpisane przez użytkownika w oknie opcji SEARCH.

5. Testowanie

Program został przetestowany przez kilka różnych osób, które nie znalazły większych problemów podczas działania. Jedyną znalezioną wadą jest duże zużycie procesora wynikające z braku licznika w głównej pętli, przez co okno jest rysowane z bardzo dużą częstotliwością. Dodatkowe ulepszenia wynikałyby tylko z dołączaniu nowych funkcji bądź informacji do programu.

Program został przetestowany pod względem wycieków pamięci. Nie zostały znalezione żadne wycieki, co świadczy o skuteczności zastosowanych obiektów.

6. Wnioski

Wpisywanie tekstu do programu przez użytkownika okazało się być jeszcze prostsze niż myślałem, co w sumie jest prawdziwe dla wszystkiego, lecz dla wpisywania miałem nieco bardziej skomplikowane wyobrażenie.

Każdy z moich projektów do tej pory miał swój ezoteryczny błąd, a Chardex żadnym wyjątkiem od reguły nie jest – w pewnym momencie by rozwiązać problem graficzny, zmusiłem program do zaczynania od wektora z identyfikatorem **-1**. Ku mojemu i znajomych zdziwieniu, działało przy pierwszym uruchomieniu programu, a potem z losową szansą. Fakt, że to nie było poprawne okazał się przy dodaniu całkowicie niezwiązanego z wektorem warunku do nstrukcji **if**, w którym to momencie 'rozwiązanie' przestało działać.

Biblioteka SFML jest wyjątkowo przyjemna w użyciu i większość problemów które powstawały, wynikały jedynie ze źle dobranych wartości przy pozycjonowaniu elementów. Projekt był bardzo dobrą nauką używania grafiki w programowaniu i na pewno będę go traktował jako bramę do bardziej zaawansowanych rzeczy.