Temat projektu: Przeprowadzanie operacji cyfrowych na obrazach, za pomocą z wizualizowanego systemu blokowego. Nazwa kodowa: PIWO - Projekt Informatyczny Wilq & Others.

Piotr Wilk
Piotr Zegar
Mateusz Tylek
Mateusz Kocąb
Wojciech Zbiegieł
Sławomir Librant
Marek Prząda

20 listopada 2008

Spis treści

1 Etap wstępny

1.1 Opis dziedziny przedmiotowej

Dziedziną projektu jest grafika, w głównej mierze operację cyfrowe.

1.2 Cel projektu – po co?

Celem projektu jest zrealizowanie programu umożliwiającego cyfrowe przetwarzanie obrazu, jako z wizualizowanego ciągu bloków, na każdym z nich będzie dodana możliwość poglądu obrazu w każdym jego stadium przekształcania. Użytkownik będzie mógł dodawać własne typy danych jaki i funkcji operujących na nich.

1.3 Zakres projektu – co i jak?

1. stworzenie dokumentacji
2. zrealizowanie graficznego interfejsu
3. implementacja operacji przetwarzania obrazu takich jak:
   (wybór pozostawiony dla pozostałych osób z grupy)
4. dodanie opcji tworzenia własnych dodatków (plug-in)

1.4 Opracowanie wymagań wstępnych

1.4.1 Oczekiwana funkcjonalność systemu

Możliwość:

1. tworzenia bloków reprezentujących wybrane operacje cyfrowe.
2. tworzenie własnych dodatków.
3. wczytania różnych formatów obrazów.
4. zapis powstałych obrazów.
5. zapis aktualnego stanu programu (położenia i połączeń bloków).

1.4.2 Opis rzeczywistych obiektów i zależności między nimi

1.4.3 Ograniczenia (system, środowisko, specyficzne wymagania)

Program zostanie napisany w IDE Borland C++, darmowa biblioteka FreeImage umożliwi wczytywanie wielu formatów plików.

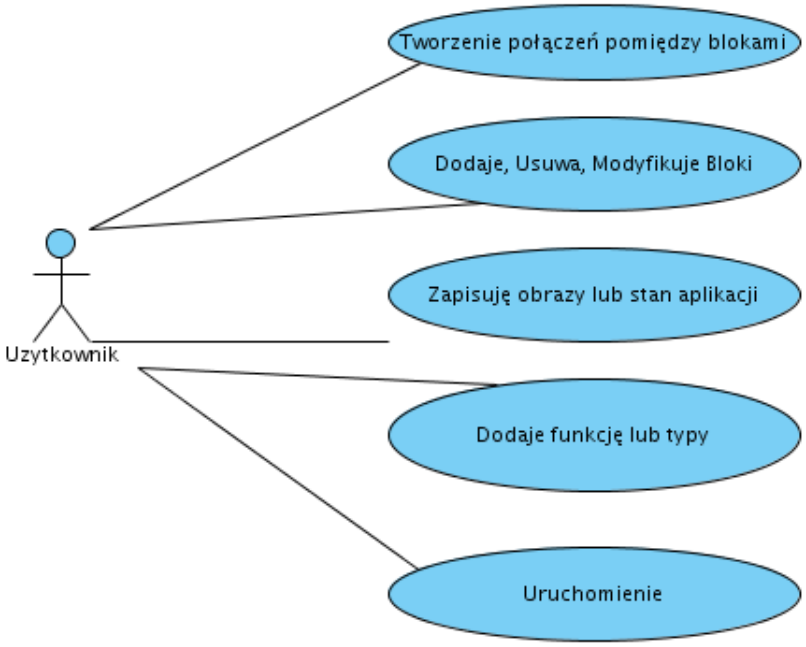
1.5 Harmonogram prac

1. Stworzenie dokumentacji projektu - Piotr Wilk
2. Zaprojektowanie oraz Implementacja silnika aplikacji - Piotr Zegar, Piotr Wilk
3. Implementacja GUI - Piotr Zegar
4. Pisanie wtyczek - pozostałe osoby.

2 Etap projektowania

2.1 Wymagania funkcjonalne

Projekt zawiera tylko jednego aktora - Użytkownika, czyli osoba pracująca z programem, ma on dostęp do wszystkich operacji.



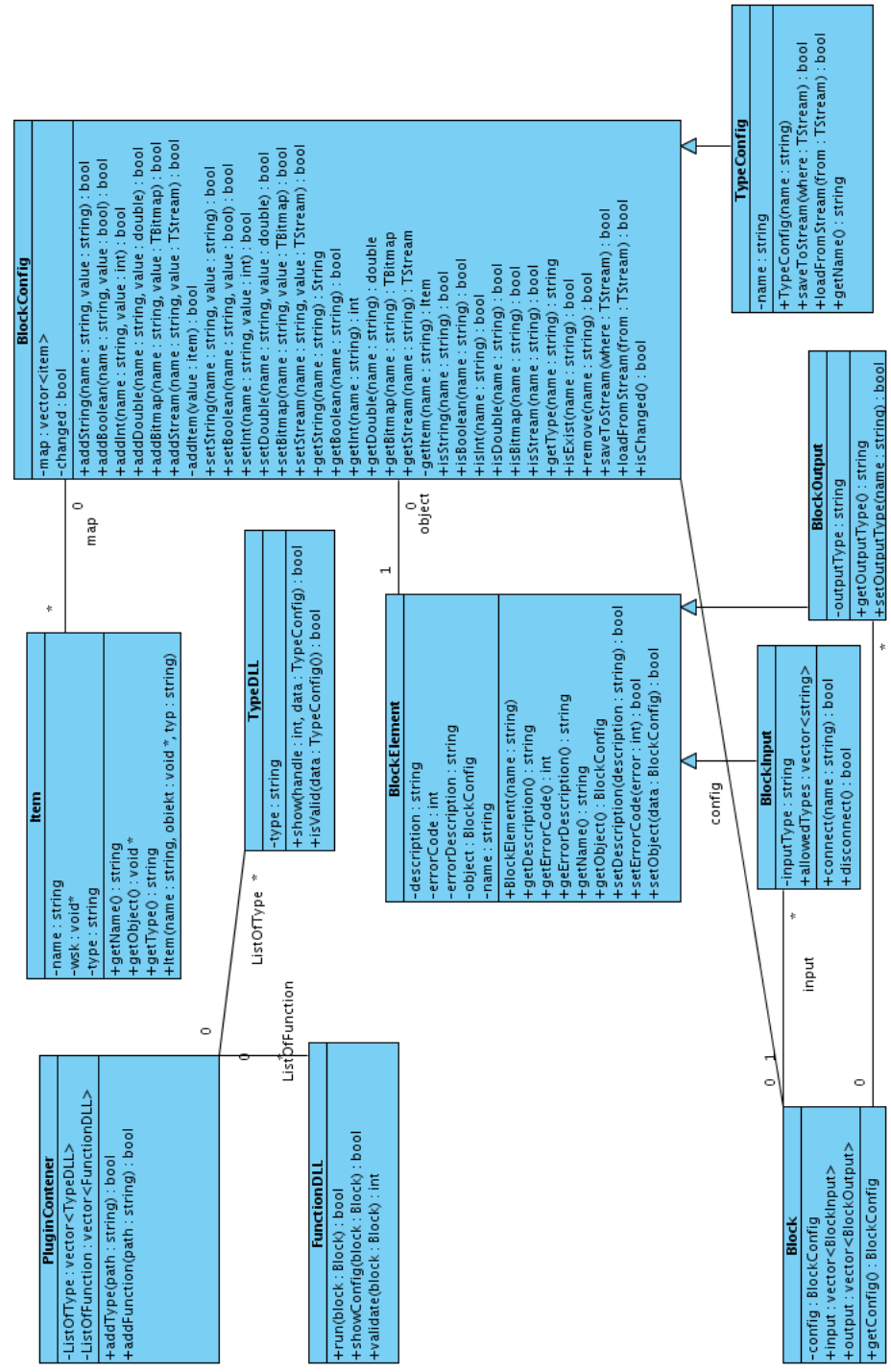
Rysunek 1: Sposoby użycia

2.2 Struktura

2.2.1 Silnik aplikacji

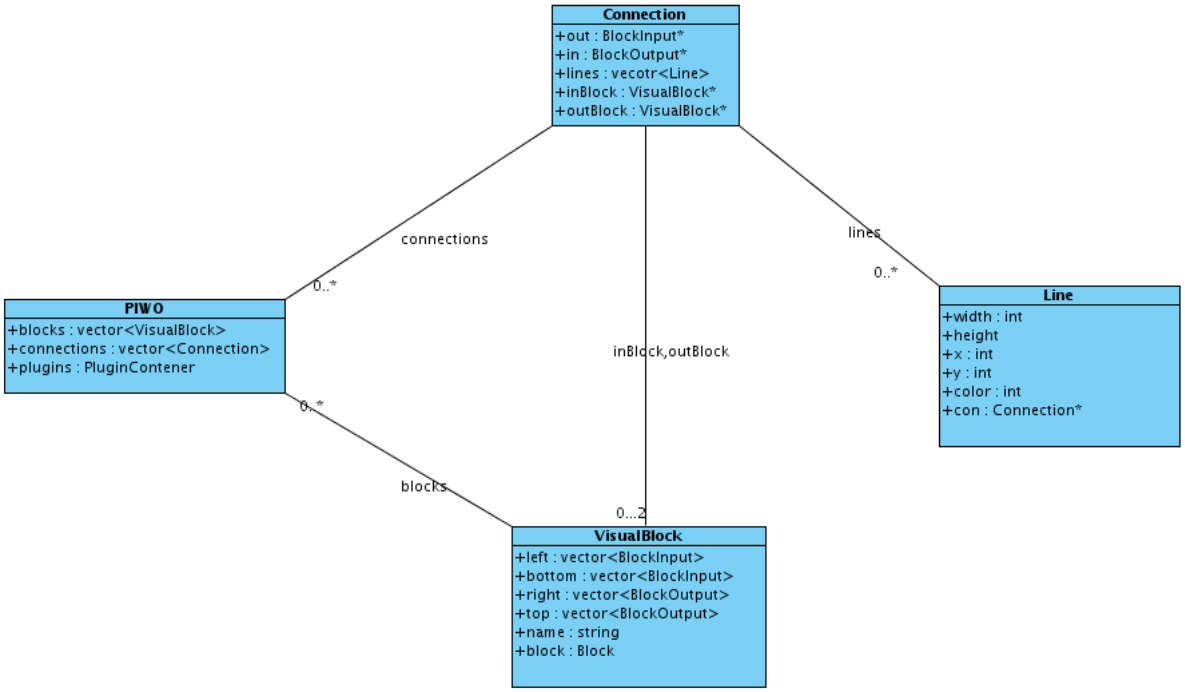
Diagram Przestawiający tylko i wyłącznie silnik aplikacji, nie ma tutaj ujętego GUI.
Klasy:

1. Item - jeden z elementów który będzie używany do opisu konfiguracji bloków, przechowuję informację o najmniejszym "atomie" takie jak typ (np: bitmap,string,int,double), name (np: width, height, byte), oraz wartość.
2. Block - klasa opisująca blok, zawiera jego konfigurację (BlockConfig), oraz jego wejścia i wyjścia.
3. BlockConfig - Konfiguracja jednego bloku, w obiekcie map zapisana jest lista obiektów.
4. BlockElement - klasa reprezentująca wejscia/wyjscia miedzy blokami, zawiera - nazwę, opis, stan - czy jest poprawnie podłączony (errorCode)
5. BlockInput - klasa opisująca tylko wejścia.
6. BlockOutput - klasa opisująca tylko wyjścia.
7. FunctionDll - klasa obsługująca nowe funkcję z dll.
8. PluginContener - przechowuję listę typów i funkcji z dll.
9. TypeConfig - typ który będzie przesyłany do bloku.
10. TypeDll - klasa obsługująca nowe typy z dll.



Rysunek 2: Diagram Klas

2.2.2 GUI



Rysunek 3: Diagram Klass dla Interfejsu Graficznego

2.3 Słownik systemu

3 Etap implementacji

3.1 Formatowanie Kodu

Kod programu będzie tworzony z myślą o automatycznym tworzeniu dokumentacji przy użyciu doxygen. Poniżej przykład formatowania:

```
/**
 *  Przykładowa klasa. i jakis opis do niej.
 *  @author Piotr Wilk
 *  @date 2008.11.19
 *  @version 1.0
 */

class Test
{
  public:
```

```
    /**
     *  Konstruktor.
     *  Opis konstruktora co robi itp.
     */
    Test();

    /**
     * Destruktor
     * Opis Destruktora
     */
   ~Test();

    /**
     * Opis funkcji.
     * @param a opis argumentu a
     * @param s opis argumentu s
     * @see Test()
     * @see ~Test()
     * @see publicVar()
     * @return opis
     */
     int testMe(int a,const char *s);

    /**
     * a public variable.
     * Details.
     */
     int publicVar;

};
```

4 Etap "wdrożenia"