Archiwum Fotogrametrii, Kartografii i Teledetekcji, Vol. 17b, 2007
ISBN 978-83-920594-9-2

PRZEGLĄDARKA STEREOGRAMÓW UTWORZONYCH ZE ZDJĘĆ POZYSKIWANYCH Z BAZ ROZPROSZONYCH

STEREO VIEWER OF IMAGES ACQUIRED FROM SPARSE DATABASE

Mariusz Twardowski

Katedra Geoinformacji, Fotogrametrii i Teledetekcji Środowiska,
Akademia Górniczo-Hutnicza w Krakowie

SŁOWA KLUCZOWE: przeglądarka, stereo, sieci, bazy, rozproszone, agent

STRESZCZENIE: W artykule przedstawiony został system wizualizacji fragmentów cyfrowych zdjęć stereoskopowych, pozyskiwanych z rozproszonych baz danych za pośrednictwem systemu agentowego. Wcześniejsze badania nad poprawnością działania systemu agentowego ujawniły potrzebę stworzenia zintegrowanego pakietu spełniającego zarówno funkcję interfejsu graficznego, a także interfejsu pozwalającego użytkownikowi na komunikowanie się z systemem agentowym, bez konieczności używania i znajomości innych technik pracy zdalnej i transferu danych. Głównymi czynnikami przemawiającymi za powstaniem takiego zintegrowanego interfejsu były nie tylko oczywiste elementy takie jak uproszczenie obsługi systemu, ale również czynniki związane ze stabilnością, elastycznością i otwartością systemu. Prezentowana przeglądarka jest rozwiązaniem wieloplatformowym, i testowana była na systemach operacyjnych Linux i Windows. Interfejs użytkownika oparty jest na bibliotece FLTK, a jego silnik graficzny na bibliotece graficznej OpenGL. Aktualnie w programie dostępne są tryby graficzne pojedynczego zdjęcia, dzielonego ekranu dla stereoskopu zwierciadlanego, poczwórnie dzielonego ekranu oraz tryb sprzętowy stereo dla okularów migawkowych lub polaryzacyjnych. Wszystkie zastosowane w projekcie rozwiązania, jak również system operacyjny i środowisko programistyczne, w którym te rozwiązania powstały, oparte są w pełni na oprogramowaniu typu open source, rozpowszechnianym na zasadach licencji GPL/LGPL lub z nią zgodnej.

1. WPROWADZENIE

Od kilku lat w byłym Zakładzie Fotogrametrii i Informatyki Teledetekcyjnej prowadzone były badania nad wykorzystaniem systemu agentów mobilnych do pozyskiwania fragmentów obrazów stereoskopowych (Twardowski 2004). Badania nad poprawnością działania systemu ujawniły potrzebę stworzenia zintegrowanego pakietu spełniającego zarówno funkcję interfejsu graficznego, a także interfejsu pozwalającego użytkownikowi na komunikowanie się z systemem agentowym, bez konieczności używania i znajomości innych technik pracy zdalnej i transferu danych. Głównymi czynnikami przemawiającymi za powstaniem takiego zintegrowanego interfejsu były nie tylko oczywiste elementy takie jak uproszczenie obsługi systemu, ale również czynniki związane ze stabilnością, elastycznością i otwartością systemu. Analiza tego problemu pozwoliła na

określenie wytycznych, którymi kierowano się przy późniejszym procesie powstawania interfejsu.

Pierwszym i nadrzędnym założeniem rozwiązania było jego wieloplatformowość, czyli program powinien, bez niemal żadnych poprawek, pozwolić skompilować swoje źródło na dowolnym systemie operacyjnym. Założenie to pociąga za sobą szereg pochodnych elementów, z których pierwszym i podstawowym jest wybór języka programowania. W środowisku informatycznym trwa niekończąca się debata nad wyższością jednego języka nad innymi, powodując lawinę argumentów, które nie przynoszą jednoznacznej odpowiedzi. Niezmiennym elementem natomiast jest fakt, że wybór języka jest w głównej mierze subiektywny i zależy od osoby tworzącej konkretną aplikację.

Dziś stosunkowo wysoka wydajność sprzętu komputerowego tworzy precedens, w którym historyczny czynnik wyboru języka w zależności od wymaganej szybkości aplikacji jest pomijalny. Teraz bardziej liczy się wygoda i przejrzystość kodu źródłowego niż fakt, że ten sam algorytm napisany przy użyciu różnych języków spowoduje po kompilacji różnice w wydajności końcowej aplikacji. Dlatego też do realizacji projektu zdecydowano się na język C++, który jest uniwersalny, wystarczająco szybki i subiektywnie wygodny (Eckel 2000). Uniwersalność tego języka odbija się bezpośrednio na wielorodności jego zastosowań, poczynając na systemach operacyjnych poprzez niezliczone biblioteki, a kończąc na prostych aplikacjach (Bulka, Meyhew 1999).

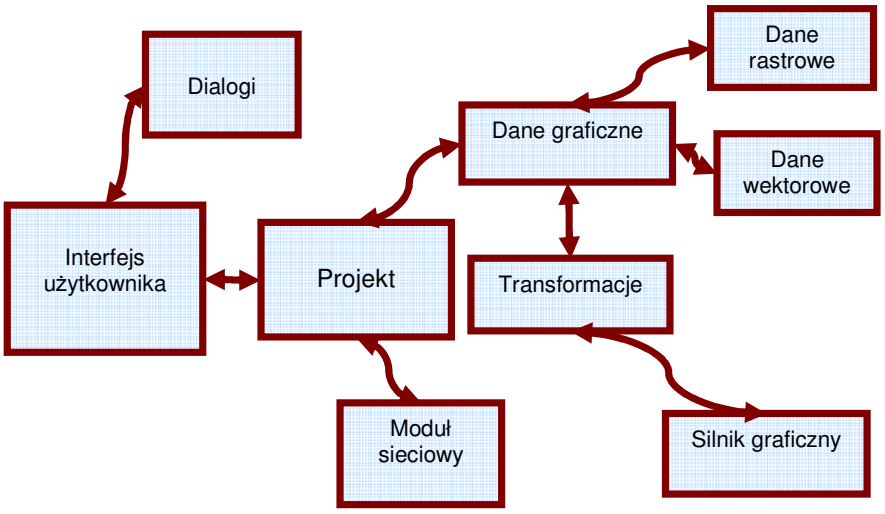
Wybór języka programowania to tylko jedna strona zagadnienia tworzenia programu, która znajduje się bardziej w sferze teoretycznych dywagacji i gramatycznej konsystencji. Drugą stroną jest wybór kompilatora tego języka, który tłumaczy idee wyrażone za pomocą wybranej składni na język zrozumiały dla procesora. Ponieważ kompilator wprowadza praktyczny wymiar dla idei jaką jest składnia języka, dlatego należy pamiętać, że kod źródłowy będzie przypominał częściowo zmodyfikowany język, co można porównać do slangu lub gwary w przypadku języka naturalnego. Innymi słowy, kod źródłowy napisany dla konkretnego kompilatora C++ może zawierać wyrażenia będące rozszerzeniem tego języka, które niekoniecznie muszą być zrozumiałe dla innego kompilatora (Stroustrup 1997).

Dlatego też, opierając się na powyższych argumentach, zdecydowano się na zastosowanie GCC jako kompilatora, ponieważ jego implementacje są obecne dla większości systemów operacyjnych. Ponadto licencja GPL tego kompilatora pozwala generalnie na jego wykorzystanie, w dowolnym celu, co jest cechą unikalną zakresie aplikacji tego typu. W trakcie tworzenia programu starano się zachować takie same kryteria dla wszystkich wykorzystanych elementów, w szczególności bibliotek które posłużyły do powstania tej aplikacji.

2. UOGÓLNIONA HIERARHIA APLIKACJI

Dzięki obiektowej charakterystyce języka C++ (Straustrup 1997) możliwe było zaprojektowanie systemu podzielonego na funkcjonalne moduły. Pozwala to, lepiej niż w przypadku programowania strukturalnego, kontrolować kolejne etapy powstawania programu i szybciej lokalizować błędy w miarę dodawania kodu źródłowego. Głównymi elementami składającymi się na program są:

- kod zarządzający projektem, który użytkownik tworzy w programie,
- kod interfejsu użytkownika, na który składa się główne menu wyboru funkcji programu oraz zestaw okien dialogowych pozwalających na wprowadzanie parametrów działania programu,
- kod silnika graficznego, który realizuje funkcje wizualizacji danych graficznych, zarówno rastrowych jak i wektorowych, na ekranie monitora za pośrednictwem karty graficznej,
- kod obsługujący dane graficzne, który wirtualnie łączy dane rastrowe i wektorowe tworząc zunifikowaną warstwę abstrakcji dla danych obrazowych pozwalając na uniwersalna wizualizację przy użyciu silnika graficznego,
- kod realizujący transformacje pomiędzy układami współrzędnych, pozwala na określenie parametrów modelu dla obrazu graficznego,
- kod do obsługi protokołów sieciowych, pozwalający na komunikację pomiędzy serwerami wchodzącymi w skład systemu agentowego a stacją roboczą użytkownika.



Rys. 1. Schemat hierarchii aplikacji dedykowanego interfejsu graficznego.

Przedstawiony na rysunku 1 uproszczony schemat hierarchii programu określa sposób przepływu danych w programie interfejsu. Dane obrazowe pobrane za pomocą modułu sieciowego wprowadzane są do projektu i interpretowane poprzez moduł obsługujący dane graficzne. Następnie na podstawie współczynników transformacji są one wizualizowane za pomocą silnika graficznego.

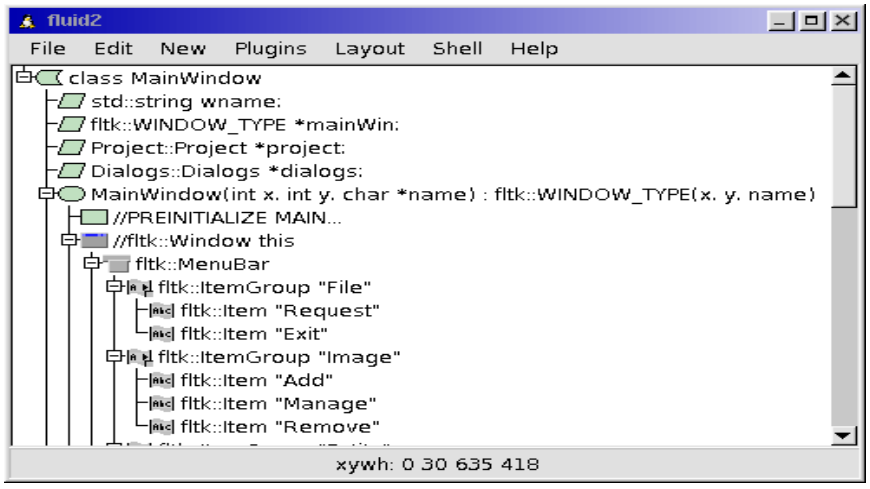
3. INTERFEJS UŻYTKOWNIKA

Każdy program mający na celu interakcję z użytkownikiem musi posiadać interfejs, który będzie pozwalał na wprowadzanie parametrów programu i modyfikację jego zachowań. Pod tym względem program przeszedł w trakcie prac nad projektem wiele modyfikacji.

Pierwsza wersja programu nie posiadała interfejsu graficznego w postaci menu i dialogów, a interakcja z programem odbywała się za pomocą linii poleceń podczas uruchamiania programu oraz skrótów klawiszowych. Pomimo że taka metoda w pełni realizuje założone w projekcie cele, pozwalając na wizualizację wycinków obrazów, to po pewnym czasie okazało się, że tendencje w praktycznej obsłudze aktualnych aplikacji wymagają graficznej reprezentacji wykonywanych operacji, poprzez wskazania kursora, czyli tzw. interfejsu okienkowego w postaci menu i dialogów.

Ponieważ decyzja odnośnie wyboru takiego interfejsu wpływała na całokształt projektu pod względem tworzenia programu jak i samej jego późniejszej obsługi, dlatego przeprowadzono szereg prób wykorzystując najbardziej popularne biblioteki realizujące funkcję interfejsu użytkownika. Wymagało to żmudnych badań nad praktycznym zastosowaniem m.in. bibliotek: freeglut (Olszta *et al.,* 2005), wxWidgets (Roebling et al., 2007), Qt (Trolltech 2007), glfw (Berglund 2006) , FLTK (Spitzak 2007), plib (Baker 2007), gtk2 (Rietvield, Janik 2006). Każda z bibliotek została przeanalizowana pod kątem wygody, spójności kodu, objętości kodu w stosunku do rezultatów i przystosowania współpracy z biblioteką graficzną OpenGL. Niektóre zostały odrzucone ze względu na zupełny brak trybu menu (np.: freeglut, glfw), niektóre ze względu na rozmiar lub licencję (np.: wxWidgets, gtk2, Qt).

Ostatecznie wybór padł na bibliotekę FLTK2, która zawiera podstawowe elementy niezbędne do stworzenia pełnowartościowej aplikacji okienkowej, przy jednoczesnym małym rozmiarze oraz stosunkowo łatwej i szybkiej możliwości tworzenia interfejsu za pomocą dedykowanej aplikacji Fluid.



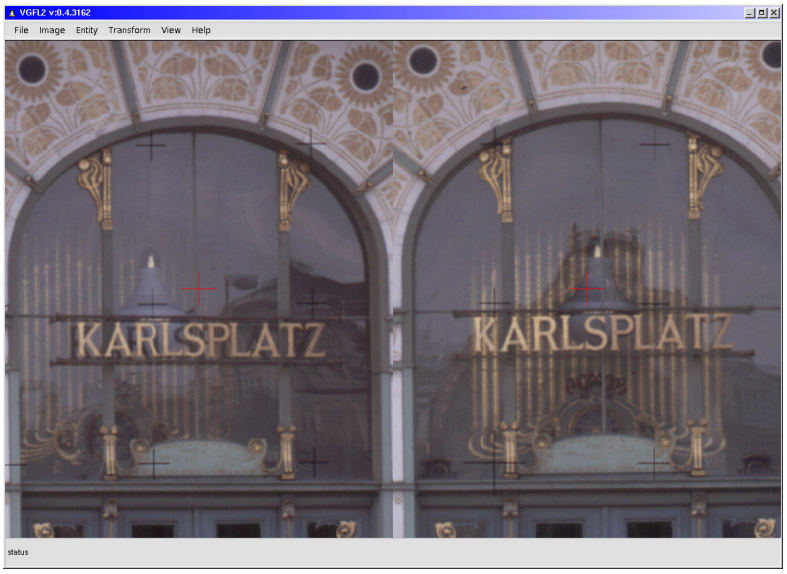
Rys. 2. Aplikacja Fluid do tworzenia interfejsu programu.

FLTK jest niewielką, elastyczną biblioteką wieloplatformową i z powodzeniem została przetestowana na kilku systemach operacyjnych i pozwala ona na proste i szybkie tworzenie menu graficznego. Ponadto posiada wbudowaną obsługę dla akcelerowanego trybu graficznego OpenGL za pośrednictwem klasy GlWindow, która pozwala silnikowi graficznemu na renderowanie obiektów bezpośrednio do okna aplikacji (Spitzak 2007). Zastosowanie jej w projekcie pozwoliło na stworzenie uniwersalnej platformy do dalszej rozbudowy programu.

4. ARCHITEKTURA SILNIKA GRAFICZNEGO

Silnik graficzny służący do przedstawiania obiektów na ekranie oparty jest o wieloplatformową bibliotekę OpenGL, która pozwala nie tylko na wydajne renderowanie obiektów ale również na pracę w trybie stereo z zastosowaniem okularów migawkowych lub polaryzacyjnych (Martz 2006). Biblioteka ponadto jest dobrze udokumentowana w zastosowaniu na różnych systemach operacyjnych (Cojot 1996; Kilgard 1996; Fosner 1996) i wspomagana przez niemal wszystkich producentów kart graficznych m.in.: nVidia i ATI. Przejrzyście i jednoznacznie zdefiniowana specyfikacja biblioteki pozwala na dokładne odwzorowanie obrazu na ekranie monitora, przy pomocy stosunkowo łatwego w obsłudze API kompatybilnego z językiem C/C++ (Alkaley, Segal 2006).

Ze względu na charakterystykę wizualizacji program interfejsu zawiera obsługę dzielenia okna graficznego w zależności od wymaganego trybu co przedstawiono na rysunku 3. Aktualnie dostępne są tryby: pojedynczego zdjęcia, podwójnie dzielonego ekranu dla stereoskopu zwierciadlanego, poczwórnie dzielonego ekranu, na potrzeby identyfikacji jednego punktu na czterech pokrywających się zdjęciach oraz tryb stereo dla wykorzystania okularów migawkowych lub polaryzacyjnych.



Rys. 3. Tryb dzielonego ekranu dla stereoskopu zwierciadlanego.

W przypadku migawkowego trybu stereo wymagany jest odpowiedni sprzęt w postaci karty graficznej obsługującej tzw. quadbuffer, który pozwala na renderowanie osobno lewego i prawego tylnego bufora obrazu, gdy w międzyczasie przedni lewy i prawy bufor są widoczne i wyświetlane naprzemiennie na ekranie monitora.

Dodatkowo do poprawnej obserwacji w tym trybie wymagane są okulary migawkowe LCD (Liquid Crystal Display), które po podłączeniu do karty graficznej (kablem lub bezprzewodowo) realizują funkcję przesłaniania lewego lub prawego oka obserwatora i są zsynchronizowane do treści wyświetlanej na ekranie. Działanie okularów migawkowych opiera się na zasadzie polaryzacji ciekłych kryształów w nich zawartych na skutek przyłożenia potencjału. Przepływ prądu przez materiał polaryzujący zawarty w normalnie przeźroczystym okularze powoduje, że następuje polaryzacja roztworu w którym kryształy zmieniają położenie i blokują przepływ promieni widzialnych.

W momencie zaniku napięcia kryształy powracają do pierwotnego położenia i okular ponownie przepuszcza światło. Dlatego też okulary migawkowe mogą być używane wyłącznie z monitorami CRT, lub z rzutnikami. W przypadku paneli LCD występuje konflikt polaryzacji i należy stosować okulary statycznie spolaryzowane.

Pod względem wykonywania pomiaru cechą unikalną programu, w porównaniu z innymi rozwiązaniami, spotykanymi w stacjach cyfrowych, jest działanie w trybie nieruchomego kursora, podczas gdy obraz pod kursorem przesuwa się w zależności od ruchów urządzenia wskazującego np. myszy. Zaletą tego rozwiązania jest większy komfort obserwacji dla użytkownika, ponieważ znaczek pomiarowy zawsze znajduje się w centrum pola widzenia. Wadą natomiast to, że do płynnego przesuwania obrazu w czasie rzeczywistym wymagana jest większa wydajność procesora komputera powiązana z dodatkową sprzętową akceleracją graficzną.

5. CHARAKTERYSTYKA OBSŁUGIWANYCH DANYCH GRAFICZNYCH

Prezentowany program obsługuje obecnie obrazy w postaci plików rastrowych zapisanych w formacie zgodnym ze specyfikacją TIFF 6.0 o dowolnej ilości bitów na piksel (8, 16, 32), dowolnym formacie przechowywania danych (tile, strip, line) oraz kompresjach LZW, Deflate oraz JPEG. Przy czym algorytm LZW ograniczony jest wyłącznie do ekstrakcji, ze względu na ograniczenia patentowe. Obsługa formatu TIFF odbywa się za pośrednictwem biblioteki libtiff wspomaganej przez bibliotekę geotiff w celu przyszłościowego rozwoju aplikacji o obsługę obrazów z zawartymi informacjami w tym formacie. Dodatkowo eksperymentalnie wprowadzona została obsługa obrazów w formacie BMP, JPEG, PNG i GIF za pośrednictwem biblioteki FLTK. Program przewiduje jednoczesną obsługę dowolnej ilości obrazów i zarządzanie nimi w dowolnej konfiguracji wizualizacyjnej.

Ze względu na znaczne rozmiary zdjęć fotogrametrycznych program w odpowiedni sposób zarządza zasobami wycinając do wizualizacji wymagane w danym momencie dane. W przypadku wycinków obrazów pobieranych z systemu agentowego jest to cecha pomijalna, jednak ten sam algorytm znajduje zastosowanie w obu przypadkach. Biorąc pod uwagę tempo rozwoju technologii cyfrowej można przypuszczać, że w nadchodzących latach rozmiar pamięci operacyjnej przeciętnych komputerów będzie na tyle duża, że aktualne obrazy będą się w niej mieścić bez większego trudu. Mogłoby to wskazywać, że funkcjonalność alokowania tylko widocznego obszaru obrazu stanie się nieistotna. Należy

jednak pamiętać, że oprócz rozwoju techniki rośnie też zapotrzebowanie użytkowników i prawdopodobnie w przyszłości znacznie zwiększą się także rozmiary obrazów cyfrowych.

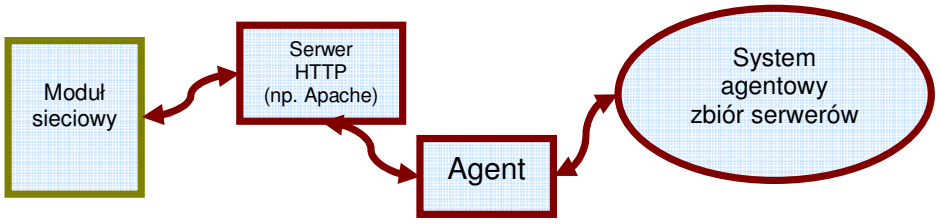
Oprócz obsługi danych rastrowych program przewiduje wizualizację danych wektorowych, na potrzeby np. mierzenia punktów lub rysowania linii. Moduł dodawania elementów wektorowych pozwala na wybranie jednej ze zdefiniowanych grup lub wprowadzić własną grupę. Poza tym można zdefiniować które współrzędne chcemy użyć przy pomiarze, co ma potencjalne zastosowanie np. w przypadku zdjęć naziemnych, na których tradycyjnie pomiar odbywa się w układzie obróconym w stosunku do wykorzystywanego przy pomiarze na zdjęciach lotniczych lub satelitarnych.

Po przeprowadzeniu pomiaru można wyświetlić współrzędne punktów w zależności od układu odniesienia. Ponadto w tym samym oknie dialogowym możliwa jest edycja parametrów wybranej grupy: koloru, rozmiaru symbolu, aktywność i widoczność grupy, oraz możliwość zdefiniowania autonumeracji. Obecnie w programie dostępna jest możliwość pomiaru punktów na obrazach, która to może być przeprowadzona niezależnie dla każdego zdjęcia i w dowolnych grupach.

6. INTEGRACJA INTERFEJSU GRAFICZNEGO Z SYSTEMEM AGENTOWYM

W celu maksymalnego uproszczenia relacji pomiędzy użytkownikiem a systemem agentowym, wprowadzono do programu dedykowany moduł sieciowy realizujący funkcję zapytań do systemu agentowego oraz automatycznego pobierania żądanych fragmentów bezpośrednio do interfejsu graficznego. Moduł wymaga podania najbliższego serwera, na którym działa system agentowy, współrzędnych wymaganej lokacji oraz rozmiaru żądanego fragmentu. Następnie użytkownik musi wysłać zapytanie do serwera i po pewnym okresie sprawdzić rezultaty kwerendy.

Na podstawie wprowadzonych parametrów moduł łączy sie z serwerem za pomocą protokołu HTTP wysyłając zapytanie typu POST. Serwer HTTP wykorzystując informacje za pośrednictwem skryptu CGI wywołuje na serwerze program agenta, który gromadzi dane ze wszystkich serwerów należących do systemu. Jeżeli agent znajdzie odpowiadające dane, wówczas zostają one udostępnione na serwerze który zapoczątkował proces i możliwa jest ich wizualizacja w interfejsie graficznym.



Rys. 4. Schemat przepływu informacji pomiędzy interfejsem a systemem agentowym.

Do stworzenia modułu wykorzystano bibliotekę libcurl [cURL 2007], która realizuje funkcję transmisji danych pomiędzy interfejsem a serwerem HTTP, dzięki czemu moduł

obsługi sieciowej mógł zostać w znacznym stopniu uproszczony. Użyta biblioteka zapewnia ponadto obsługę wielu innych protokołów sieciowych takich jak: FTP, FTPS, HTTP, HTTPS, SCP, SFTP, TFTP, TELNET, dzięki czemu zapewniona zostaje w programie elastyczność w przypadku konieczności rozbudowy lub zmiany protokołu wymiany danych.

7. LITERATURA

Alkaley K., Segal M., 2006. The OpenGL Graphic System: A Specification, http://opengl.org.

Berglund C., 2006. GLFW - An OpenGL Framework, http://glfw.sourceforge.net.

Bulka D., Meyhew D., 1999. Efficient C++ Performance Programming Techniques, Addison Wesley.

Cojot V. , 1996. OpenGL Programming on Linux, Linux Journal.

cURL, 2007. http://curl.haxx.se.

Eckel B., 2000. Thinking in C++ 2nd ed., Prentice Hall, http://prenhall.com.

Fosner R. , 1996. OpenGL Programming for Windows 95 and Windows NT, Addison-Wesley.

FSF, 2007. Philosophy of the GNU Project, Free Software Foundation, http://gnu.org.

Gray R. 1997. Agent TCL: A flexible and secure mobile-agent system, Dartmouth College.

Kilgard M., 1996. OpenGL Programming for the X Window System, Addison-Wesley.

Libtiff, 2007. TIFF Library and Utilities, http://remotesensing.org/libtiff.

Olszta P., Umbach A., Baker S., 2005. The Free OpenGL Utility Toolkit, http://freeglut.sf.net.

Paszotta Z., 2003. Exterior orientation and other photogrammetric solutions through the Internet, Automatic Georeferencing of Aerial Images by Means of Topographic Database Information, Aalborg University, pp. 25-32.

Rietveld K., Janik T., 2006. Highlights of GTK+ 2.10, http://www.gtk.org/~timj/papers/gtk-guadec2006.pdf.

Roebling R., Zeitlin V., Csomor S., Smart J., Slavik V., Dunn R., 2007. wxWidgets Cross-Platform GUI Library, http://www.wxwidgets.org.

Spitzak B., 2007. Fast Light Toolkit, http://fltk.org.

Stroustrup B., 1997. The C++ Programming Language, Third Edition, Addison Wesley Longman Inc..

Trolltech, 2007. http://trolltech.com.

Twardowski M., 2004: System agentowy w geoinformatyce obrazowej, Archiwum Fotogrametrii, Kartografii i Teledetekcji, vol. 14, Białobrzegi - Warszawa 2004

STEREO VIEWER OF IMAGES ACQUIRED FROM SPARSE DATABASE

KEY WORDS: viewer, stereo, network, database, sparse, agent

SUMMARY: The paper presents visualisation system for pieces of digital stereo-photographs acquired from sparse database, through mobile agent system. Past studies showed the necessity to create an integrated package providing graphic interface and communication facility with agent system. The main pros of creating such package were not only such obvious issues as simplified use, but also those related to the stability, flexibility and openness of system. Presented viewer is a multi-platform solution, and was tested in Linux and Windows systems. User interface is based on the FLTK library, and the graphic engine relies on OpenGL toolkit. At present, the viewer provides graphical modes of a single image, double split screen for mirror stereoscope, quad split screen and hardware quadbuffered stereo for LCD shutter glasses. Every solution used in this project, including the operating system and development environment, is based fully on open source software with GPL/LGPL license or a similar compatible one.

dr inż. Mariusz Twardowski
e-mail: misiek@kpg.pl
telefon: 502 252 950