Archiwum Fotogrametrii, Kartografii i Teledetekcji, Vol. 18, 2008
ISBN 978-83-61576-08-2

WYKORZYSTANIE PRZEGLĄDARKI VGLFL DO POZYSKIWANIA I OPRACOWANIA DANYCH ZE ŹRÓDEŁ HETEROGENICZNYCH

VGLFL VIEWER FOR ACQUIRING AND DEVELOPING DATA FROM HETEROGENEOUS SOURCES

Mariusz Twardowski

Akademia Górniczo-Hutnicza

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

STRESZCZENIE: W artykule opisany zostanie system wizualizacji fragmentów cyfrowych zdjęć stereoskopowych o roboczej nazwie VGLFL, który został stworzony do pozyskiwania danych z baz rozproszonych za pośrednictwem systemu agentowego. Ponadto przedstawiona zostanie możliwość wykorzystania tegoż systemu do pozyskiwania i opracowania danych pochodzących z innych źródeł, takich jak publiczne materiały dostępne za pośrednictwem protokołu HTTP. Przeglądarka wyposażona została w interfejs użytkownika oparty na bibliotece FLTK oraz silnik graficzny oparty o bibliotekę OpenGL, która pozwala na wydajną wizualizację, oraz integrację obsługi okularów stereoskopowych. W celu maksymalnego uproszczenia relacji pomiędzy użytkownikiem a systemem agentowym, wprowadzono do programu dedykowany moduł sieciowy, oparty o bibliotekę *libcurl* i realizujący funkcję zapytań do systemu agentowego oraz automatycznego pobierania żądanych fragmentów bezpośrednio do interfejsu graficznego. Rozszerzenie funkcjonalności modułu sieciowego pozwoliłoby na zwiększenie zasobu dostępnych źródeł danych, gdzie oprócz dedykowanej rozproszonej bazy danych, dostępnej przy pomocy systemu agentowego, możliwe byłoby wykorzystanie publicznych źródeł danych obrazowych udostępnianych przy użyciu protokołu HTTP. Wprowadzając odpowiednie zmiany w programie możliwe byłoby wykorzystanie np.: zasobów zdjęć lotniczych Terraserver USA lub zdjęć satelitarnych NASA World Wind, bez konieczności modyfikacji funkcjonalności związanych z systemem agentowym. Rozszerzenie takie polegałoby na implementacji odpowiedniego interfejsu wymiany danych pomiędzy serwerem HTTP udostępniającym dane, a klientem przeglądarki. 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. OGÓLNA CHARAKTERYSTYKA PRZEGLĄDARKI

Badania prowadzone w ostatnich latach w Katedrze Geoinformacji, Fotogrametrii i Teledetekcji Środowiska AGH obejmowały zastosowanie systemów agentowych do pozyskiwania danych obrazowych z rozproszonych baz danych (Jachimski et al., 2004). Efektem badań była metoda składająca się z następujących etapów:

1) wprowadzenie przez użytkownika do interfejsu parametrów określających pożądaną lokalizację,

2) wysłanie zapytania użytkownika z interfejsu do najbliższego serwera na którym działa opracowany system agentowy,
3) wysłanie przez agenta który kolekcjonuje dane i w razie potrzeby wydaje polecenia ich przetwarzania na poszczególnych serwerach należących do systemu.
4) po powrocie agenta na serwer macierzysty podjęcie odpowiednich działań na podstawie danych zgromadzonych przez agenta (pobranie wycinków obrazów, wysłanie informacji o zakończonym procesie do interfejsu użytkownika, wysłanie maila informującego użytkownika o zakończonym procesie itp.),
5) przesłanie zgromadzonych danych do interfejsu użytkownika,
6) wizualizacja danych na ekranie komputera użytkownika.

W wyniku uzyskano zintegrowany sieciowy system przetwarzania danych, w którym jednym z elementów była stereoskopowa przeglądarka obrazów cyfrowych pochodzących ze zdjęć lotniczych. Głównym założeniem podczas tworzenia przeglądarki była jej wieloplatformowość, czyli program powinien, bez żadnych istotnych zmianach w kodzie, pozwolić skompilować swoje źródło na dowolnym systemie operacyjnym. Założenie to pociąga za sobą szereg pochodnych elementów, z których pierwszym i podstawowym był wybór języka programowania. W przypadku przeglądarki zdecydowano się na zastosowanie kompilatora GCC, ponieważ jest on dostępny dla większości systemów operacyjnych włączając w to Linux i Microsoft Windows. Ponadto licencja jaką jest objęte to oprogramowanie, czyli GPL, pozwala na jego wykorzystanie, w dowolnym celu, co jest cechą unikalną zakresie aplikacji tego typu. Dodatkowo przy pracach nad programem starano się zachować takie same kryteria dla wszystkich pozostałych elementów, a w szczególności bibliotek wykorzystanych przy powstawaniu aplikacji.

Wykorzystując możliwości języka C++ program został podzielony na kilka funkcjonalnych modułów zawartych w klasach i metodach poszczególnych klas. Tak stworzone obiekty pozwalają na łatwiejsze kontrolowanie kodu, niż ma to miejsce w przypadku programowania strukturalnego. W efekcie program można podzielić na kilka części odpowiedzialnych za określone funkcje:

1) moduł do zarządzania elementami projektu takimi jak: ilość załadowanych obrazów, stworzone grupy elementów wektorowych lub rodzaj organizacji obrazu,
2) moduł zawierający interfejs użytkownika, na który składa się główne menu wyboru funkcji programu oraz zestaw okien dialogowych pozwalających na wprowadzanie parametrów działania programu,
3) moduł obsługi silnika graficznego, realizującego funkcje wizualizacji graficznych danych rastrowych i wektorowych, które są przedstawiane na ekranie monitora za pośrednictwem karty graficznej,
4) moduł do obsługi danych graficznych, wirtualnie łączącego dane rastrowe i wektorowe, tworząc w ten sposób zunifikowaną warstwę abstrakcji dla danych obrazowych pozwalając na wizualizację przy użyciu silnika graficznego,
5) moduł obliczający parametry transformacji pomiędzy różnymi układami współrzędnych, pozwala na określenie parametrów modelu dla danych graficznych,
6) moduł obsługujący protokoły sieciowe, pozwala na komunikację pomiędzy serwerami a stacją roboczą użytkownika.

Moduły zostały zorganizowane w taki sposób aby przekazywać sobie istotne informacje, bez konieczności ingerencji w ich wewnętrzną strukturę.

Przy tworzeniu programu posłużono się dostępnymi na licencji LGPL bibliotekami, co przyspieszyło w znacznym stopniu proces powstawania aplikacji. Interfejs użytkownika powstał za pośrednictwem biblioteki FLTK, która umożliwia tworzenie podstawowych elementów niezbędnych do stworzenia pełnowartościowej aplikacji okienkowej, przy jednoczesnym małym rozmiarze oraz stosunkowo łatwej i szybkiej możliwości tworzenia interfejsu za pomocą dedykowanej aplikacji Fluid. FLTK jest niewielką, elastyczną biblioteką wieloplatformową, 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.

Silnik graficzny programu powstał przy użyciu popularnej biblioteki OpenGL, która jest szeroko stosowana w aplikacjach wymagających graficznej akceleracji sprzętowej. W przeciwieństwie do bibliotek Microsoft DirectX zachowuje ona elastyczność w kompilacji na różnych platformach sprzętowych i systemowych, jak również nie pociąga za sobą implikacji licencyjnych. OpenGL 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 et al., 2006). Obraz wyświetlany przy pomocy silnika graficznego na ekranie monitora jest generowany z połączenia dwóch źródeł. W przypadku danych rastrowych są one ładowane do pamięci komputera w postaci tzw. *taili* czyli stosunkowo małych kwadratów będących fragmentami rastra. Tak przygotowana tekstura nakładana jest na obszar zdefiniowany poprzez cztery narożniki tzw. *quady*, które reprezentują jeden z rodzajów tworów prymitywnych dostępnych w specyfikacji OpenGL. Następnie na przygotowany w ten sposób zwizualizowany raster nakładane są warstwy danych wektorowych zdefiniowane za pomocą punktów, linii i ciągów liniowych.

2. OBSŁUGA PROTOKOŁÓW SIECIOWYCH

Większość istniejących aplikacji sieciowych zakłada model, w którym z jednej strony połączenia znajduje się klient, czyli komputer żądający połączenia, a z drugiej serwer, czyli komputer oczekujący na nadejście połączenia. Model taki tradycyjnie nazywany jest client-server. Dodatkowo serwery używające tej metody można podzielić na dwie grupy: iteracyjne i konkurentne. W przypadku serwera iteracyjnego czeka on na nadejście połączenia od klienta, przetwarza zapytanie klienta, wysyła odpowiedź i wraca do oczekiwania. Problem powstaje gdy przetwarzanie zajmuje dużo czasu i w tym czasie serwer nie może obsługiwać innego klienta równolegle. Dlatego częściej używany jest serwer konkurentny, który również czeka na połączenie klienta, ale po nadejściu połączenia uruchamia nowy proces serwera np. w nowym wątku, i ten nowy proces

obsługuje klienta bez blokowania oryginalnego serwera. Po zakończeniu przetwarzania i wysłaniu odpowiedzi dodatkowy program zostaje zakończony (Bishop et al 2005).

Wśród rozlicznych usług, które mogą być realizowane w sieciach opartych na protokole TCP/IP, jedną z najbardziej elementarnych i podstawowych jest Telnet (Postel et al., 1983). Technicznie jest to najprostsza z usług sieciowych: służy do nawiązania interaktywnego połączenia terminalowego ze wskazanym komputerem, a dokładniej z programem - serwerem udostępniającym usługę opartą na protokole o nazwie Telnet, pracującym na tym komputerze. Po nawiązaniu takiego połączenia znaki wpisywane na klawiaturze naszego komputera przesyłane są poprzez sieć do maszyny, z którą jesteśmy połączeni, a przesyłane w odwrotną stronę odpowiedzi wyświetlane są na naszym ekranie. Można w ten sposób wydawać komendy komputerowi na którym jesteśmy zalogowani poprzez linię poleceń, bez konieczności dostępu do klawiatury czy monitora tego zdalnego serwera. Protokół ten ma niewielkie zastosowanie w opisywanym projekcie ze względu na brak możliwości bezpośredniego transferu plików.

FTP czyli *File Transfer Protocol*, jest to protokół umożliwiający przesyłanie plików poprzez sieć Internet (Postel et al., 1995). Aby możliwe było przesyłanie plików za pomocą tego protokołu, jeden komputer musi pełnić funkcję serwera FTP. Co pozwala użytkownikowi, na drugim komputerze uruchomić program klienta FTP, za pomocą którego loguje się do serwera. Teraz możliwe jest już przesyłanie plików w dwie strony. Użytkownik nie zawsze jednak ma dostęp do wszystkich plików na serwerze; zależy to od uprawnień, jakie mu przysługują. Popularną usługą jest także anonimowe FTP, dzięki któremu użytkownik może się zalogować do komputera, na którym nie ma żadnych praw dostępu. Wówczas jako nazwę użytkownika podaje "anonymous", hasłem zaś jest jego adres e-mail. Użytkownik ma wtedy możliwość korzystania z wybranych zasobów serwera bez posiadania specjalnego konta użytkownika. Taki anonimowy dostęp pozwala użytkownikowi na korzystanie z materiałów publicznych, które chcemy rozpowszechnić w jak najszerszym kręgu ludzi.

Jednym z najbardziej spopularyzowanych sposobów realizowania połączenia klient-serwer jest udostępnianie danych przy pomocy protokołu http z wykorzystaniem graficznych przeglądarek internetowych. Sposób ten charakteryzuje się dostateczną elastycznością i szybkością, które są niezbędne do obsługi rozproszonego systemu informacji hipermedialnych. Jest to ogólny i zorientowany obiektowo protokół, który może być wykorzystywany do wielu zadań, takich jak zarządzanie serwerami nazw lub systemami obiektów rozproszonych, a to dzięki rozszerzeniu jego zestawu instrukcji. Kiedy użytkownik uruchomi swoją przeglądarkę i wprowadzi nazwę serwera lub jego adres IP, następuje poszukiwanie właściwego miejsca w sieci (korporacyjnej lub w Internecie, zależnie od uzyskanego połączenia) (Fielding i inni 1999). Po jego odnalezieniu nawiązywana jest komunikacja z wybranym serwerem.

Przeglądarka została stworzona jako interfejs pomiędzy użytkownikiem i systemem agentowym, pozwalając tym samym na uproszczenie interakcji pomiędzy nimi. Aby możliwa była taka funkcjonalność program zawiera zintegrowaną obsługę protokołów sieciowych za pośrednictwem biblioteki *libcurl,* 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 (cURL, 2007), dzięki czemu zapewniona zostaje w programie elastyczność w przypadku konieczności rozbudowy lub zmiany protokołu wymiany danych.

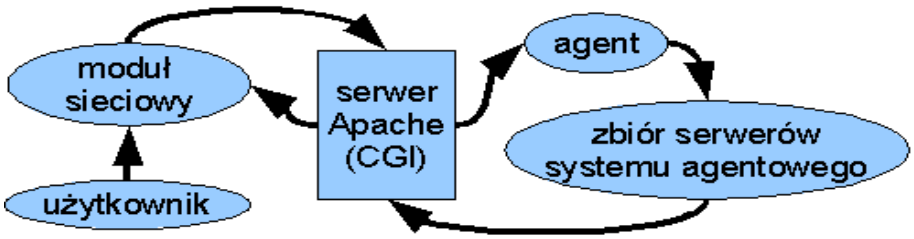
3. WYKORZYSTANIE PROTOKOŁU HTTP

HTTP to protokół sterujący, odpowiedzialny za ustanowienie połączenia pomiędzy klientem a serwerem i przekazujący instrukcje pomiędzy tymi systemami. HTML jest językiem formatującym dokumenty, które w odpowiedni sposób interpretowane są przez przeglądarkę, a następnie wyświetlane na monitorze. Każde zapytanie (żądanie) klienta (czyli przeglądarki) składa się z trzech części:

1) linii zapytania (klient: polecenie method, adres dokumentu i wersja protokołu HTTP; serwer: wersja protokołu HTTP, kod stanu serwera, opisujący m.in. informację czy dana strona istnieje, czy nie i opis,
2) listy nagłówków (klient: opcjonalne powiadomienie serwera m.in. o tym, jakie rodzaje dokumentów potrafi obsłużyć przeglądarka, typ przeglądarki itp.; serwer: dane dotyczące serwera i dokumentu),
3) części zasadniczej (klient: informacje dodatkowe wykorzystywane przez serwer np. podczas wykonywania skryptów CGI; serwer: treść dokumentu, wynik działania skryptu CGI bądź, w wypadku gdy żądany dokument nie istnieje, bądź skrypt nie może być wykonany, czytelną informację o zaistniałym błędzie).

HTTP jest protokołem bezstanowym, co oznacza że serwer nie może wysyłać zapytań do klienta (Kristol et al., 1997). W zasadzie po zapytaniu klienta i pobraniu danych połączenie jest przerywane. Jest to poniekąd pożądane ze względów bezpieczeństwa, ale jednocześnie jest wadą pod względem wygody użytkowania. Serwer mogąc pobrać informacje ze strony klienta mógłby w lepszy sposób automatycznie reagować na zapytania, niestety odbywałoby sie to kosztem prywatności. Ze względu na tą cechę powstały odrębne mechanizmy pozwalające na wysyłanie informacji, których potrzebuje serwer, a o których nie wie przeglądarka użytkownika. Przykładem takim mogą być tzw. *cookies*, które są małymi plikami pobieranymi bez naszej wiedzy z serwera a następnie, przy kolejnym połączeniu z tą samą witryną, są odsyłane zawarte w nich informacje w postaci niezamierzonego zapytania klienta. W rezultacie, mimo że serwer nie może wysłać zapytania do klienta to i tak otrzymuje informacje które potrzebował. Dzięki temu istnieje np. możliwość automatycznego logowania na strony.

W rozwiązaniu przyjętym do tej pory w przeglądarce komunikacja polega na połączeniu po protokole HTTP z serwerem Apache i wysłania zapytania przy użyciu metody POST. Serwer Apache interpretując kod zapytania uruchamia usługę CGI, poprzez którą parametry wykonania przesyłane są do agenta. Usługa CGI obsługiwana jest przez specjalny mały dedykowany program, który interpretuje komendy pozyskane z metody POST. Po tej operacji program agenta rozpoczyna gromadzenie danych, a użytkownik po pewnym czasie może sprawdzić wyniki zapytania. Kontrola zapytań jest rozwiązana w taki sam sposób jak samo zapytanie. Graficznie schemat przepływu informacji można przedstawić tak jak na rysunku (1).



Za wszystkie operacje sieciowe zawarte w przeglądarce odpowiada moduł sieciowy oparty o wspomnianą już bibliotekę libcurl. Moduł sieciowy przeglądarki generuje zapytanie dla serwera, co można przedstawić w następujący sposób:

curl_easy_setopt(chnd, CURLOPT_POSTFIELDS, postdata);

curl_easy_setopt(chnd, CURLOPT_URL, hosturl);

curl_easy_setopt(chnd, CURLOPT_WRITEFUNCTION, curlwriter);

curl_easy_setopt(chnd, CURLOPT_WRITEDATA, &reply);

curl_easy_perform(chnd);

Gdzie ciąg znaków „postdata" jest odpowiednio wygenerowanym zapytaniem metody POST, „hosturl" jest ciągiem zawierającym adres sieciowy, „curlwriter" jest metodą określającą interpretację danych przesłanych z serwera, a „reply" to referencja do zarezerwowanej pamięci w której odpowiedź serwera ma zostać zapisana. Korzystając z tego przykładu widać, że w bardzo prosty sposób, nawet bez ingerencji w kod źródłowy modułu można zmodyfikować zapytanie wysyłane przez moduł sieciowy tak, aby dane były pobierane bezpośrednio z serwera HTTP udostępniającego pliki graficzne. Możliwa jest wtedy niejako emulacja przeglądarki, która zazwyczaj łączy się z serwerem map. Oczywiście wymagana jest późniejsza odpowiednia interpretacja danych w programie, aby nastąpiła poprawna wizualizacja danych.

Korzystając z publicznie dostępnego serwera Terraserver USA można pobrać inicjalną stronę przeglądową mapy. Pobierając stronę pod adresem terraserverusa.com uzyskujemy kod HTML z którego analizy wynika, że pozyskać dane graficzne można używając metody jawnej GET lub dwóch metod POST. Korzystając z pierwszej metody uzyskujemy dwa poziomy mapy przeglądowej dostępne np. pod adresami:

http://terraserverusa.com/cmap.aspx?src=0&ppd=1&r=4&c=3&W=0

http://terraserverusa.com/Cmap.aspx?Src=0&T=0&PPD=8&R=4&C=3&W=1

W pierwszym przypadku wybieramy Stany Zjednoczone, a w drugim jeden z południowo-wschodnich obszarów. W kolejnym kroku otrzymujemy już obraz pochodzący ze zdjęć lotniczych. Ponownie wybierając kolejne poziomy mapy możemy otrzymać adresy następujących fragmentów terenu np.:

http://terraserverusa.com/image.aspx?T=0&S=16&R=19&C=16&W=1

http://terraserverusa.com/image.aspx?T=1&S=15&Z=16&X=75&Y=563&W=1

http://terraserverusa.com/image.aspx?T=1&S=14&Z=16&X=148&Y=1126&W=1

http://terraserverusa.com/image.aspx?T=1&S=13&Z=16&X=294&Y=2252&W=1

http://terraserverusa.com/image.aspx?T=1&S=12&Z=16&X=586&Y=4504&W=1

Analizując tak pozyskane dane można zauważyć, że dane na serwerze uporządkowane w siatkę kwadratów po której można się poruszać operując zmiennymi R i C w przypadku mapy przeglądowej, a następnie zmiennymi X, Y i S w przypadku docelowej ortofotomapy, gdzie S oznacza jej poziom skali. Następnie analizując uzyskany kod HTML z serwera można określić nazwy poszczególnych fragmentów mapy, oraz odnośniki do ich dwukrotnego powiększenia zamieniając nazwę skryptu „image.aspx" na nazwę „tile.ashx" co pozwala pobrać faktyczne dane obrazu o rozmiarze 256x256 pikseli i zapisanego w formacie JPEG. Drugą możliwością pozyskania danych obrazowych jest skorzystanie z dostępnych na stronie dwóch metod typu POST. Pierwsza z metod znajduje się w formie „advfind.aspx" gdzie w polach metody można wpisać nazwę lokalizacji w postaci ulicy, miasta i stanu. Druga metoda znajduje się w formie „image.aspx" w której można znaleźć interesujący obszar przy pomocy szerokości i długości geograficznej. Uzyskana w ten sposób strona przedstawia zdjęcia na poziomie S=14, z którego to można nawigować dalej kożystając w sposób przedstawiony powyżej.

Wykorzystując tę analizę widać, że można wykorzystać moduł sieciowy przeglądarki do pozyskania danych udostępnianych poprzez publiczny serwer HTTP. W tym celu należy w polu „hosturl" podać bezpośredni adres do wybranego taila danych graficznych metodą GET, lub w przypadku korzystania z metody POST wypełnić odpowiednio dostępne pole „postdata", a następnie uruchomić metodę „curl_perform" biblioteki libcurl. W zależności od zapytania dane do interpretacji lub wyświetlenia będą znajdować się w obszarze pamięci komputera zadeklarowanej przez referencję „reply". Wspomnieć należy, że pozyskane w ten sposób dane będą musiały być zapisane w tzw. *cache,* czyli wyodrębnionym zarezerwowanym wcześniej obszarze pamięci dynamicznej lub dyskowej w celu ponownego ich wykorzystania bez konieczności ponownego transferu danych z serwera. Działanie takie ma na celu przyspieszenie działania aplikacji, w przypadku kolejnych odwołań do tego samego obszaru.

Przeprowadzając podobną do przedstawionego przykładu analizę kodu źródłowego HTML pozyskanego z innych udostępniających dane serwerów HTTP, można wyodrębnić właściwe znaczniki i metody którymi należy się posłużyć aby możliwe było uzyskanie odpowiednich danych do wizualizacji. Niestety nie ma automatycznej metody uniwersalnej, która by taką operację umożliwiała, ze względu na indywidualną specyfikę poszczególnych serwerów obrazowych. Jednak możliwym byłoby wyodrębnienie pewnych grup serwerów które używają tych samych lub podobnych metod.

4. LITERATURA

Alkaley K., Segal M., 2006. The OpenGL Graphic System: A Specification.

Bishop S., Fairbairn M., Norrish M., Sewell P., Smith M., Wansbrough K., 2005. TCP,UDP, and Sockets: rigorous and experimentally-validated behavioural specification.http://www.cl.cam.ac.uk/~pes20/Netsem/tr.pdf.

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

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

Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., 1999. Hypertext Transfer Protocol - HTTP/1.1, RFC-2616 IETF. http://ietf.org.

Fosner R., 1996. OpenGL Programming for Windows 95 and Windows NT. Addison-Wesley, http://opengl.org.

Jachimski J., Twardowski M., 2003. Projekt wykorzystania systemu agendowego dla popularyzacji geoprzestrzennych informacji w postaci fotomap oraz stereogramów zdjęć lotniczych. Sztuczna inteligencja: organizacje wirtualne, WAT, Siedlce.

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

Kristol D., Montulli L., 1997. HTTP State Management Mechanism, RFC-2109 IETF. http://ietf.org.

Martz P., 2006. OpenGL Distilled. Addison Wesley Professional.

Postel J., Reynolds K., 1983. Telnet Protocol Specification, RFC-854 IETF. http://ietf.org.

Postel J., Reynolds K., 1985. File Transfer Protocol (FTP), RFC-959 ITEF. http://ietf.org.

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

VGLFL VIEWER FOR ACQUIRING AND DEVELOPING DATA FROM HETEROGENEOUS SOURCES

KEY WORDS: viewer, stereo, network, http, images

Summary

The paper describes a visualisation system of digital stereoimages, tentatively termed VGLFL. The system was develop to acquire data from sparse databases through a mobile agent system. Furthermore, a possibility of using the system to acquire and develop data originating from other sources, e.g., public resources accessible through the HTTP network protocol, will be introduced. The stereoviewer is a multi-platform solution, which means that its source code should compile without modifications on different platforms. The stereoviewer's user interface was based upon FLTK library, and the graphic engine was written using OpenGL, which allows efficient visualisation as well as integration of stereographic shutterglasses support. In terms of screen organization, the viewer supports display of one image, split screen, quadruple screen for identifying one homologous point in four images, quadbuffer stereo for shutter glasses, and anaglyph mode. Additionally, the libcurl library was used to create a network module that further simplifies interactions between the user and the mobile agent system allowing automatic acquisition and loading of images into the user interface. By extending the network protocol, there will be a possibility to increase the resource pool, whereby – in addition to standard sparse databases - other public sources such as Terraserver USA or Nasa World Wind would be available. All the solutions used in the project as well as the operating system and the development environment are fully based on open source software released under GPL/LGPL or a similar license.

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