F X S y s t e m s P i o t r D a s z e w s k i

P R A C A N A U K O W A

Piotr DASZEWSKI

System generowania cyfrowego modelu terenu w oparciu o mapy fizyczne

SPIS TREŚCI

1. Wstęp

1.1. Cel i zakres realizacji projektu.

Tematem pracy jest stworzenie systemu umożliwiającego generowanie cyfrowych modeli terenu na podstawie map fizycznych. By w pełni rozwinąć to zagadnienie głównym celem pracy, będzie opracowanie oprogramowania wykonującego to działanie.

Realizacja polegać będzie na systematycznym wypracowaniu pewnego podejścia do zagadnienia, próbach rozwiązania związanych z tym problemów, oraz końcowej implementacji w wybranym środowisku programistycznym.

Podstawą będzie zrozumienie problemu odzwierciedlania modelu terenu. Początki będą musiały opierać się o zagadnienia czysto kartograficzne. Dzięki temu możliwe będzie nawiązanie logicznego połączenia między reprezentacją terenu na mapach fizycznych, a komputerowymi modelami powierzchni. Następnie niezbędne będzie opracowanie formatu przechowywania cyfrowych danych. Będzie to związane z rodzajem rozpoznawanych danych, jak również ich późniejszym wykorzystaniem, ale o tym mowa w dalszej części pracy. Końcowym etapem rozwoju myśli związanych z pracą, będzie opracowanie rozwiązań umożliwiających przechowywanie, analizowanie i wykorzystanie zgromadzonych informacji. Sam koniec, polegać będzie na eliminowaniu błędów powstałych podczas wcześniejszych kroków i dostosowanie projektu do realnych możliwości implementacyjnych i wykonawczych.

1.2. Uzasadnienie podjęcia tematu.

Zagadnienia związane z reprezentacją danych przestrzennych, w jakiejkolwiek postaci są wyjątkowo obiecujące. W dzisiejszym świecie otacza nas niezliczony ogrom takich informacji. Praktycznie większość interesujących nas zagadnień ma swoje odzwierciedlenie w przestrzeni. Umiejscawiać można budynki, instytucje, a nawet ruchome samochody, czy zjawiska meteorologiczne. W związku z tym, moje zaciekawienie daną tematyką, będzie mogło się pogłębiać wraz z realizacją projektu ściśle z tym związanego.

Mam nadzieję stworzyć, może nie tyle dobre oprogramowanie, gdyż sądzę, że opracowanie wyjątkowo wydajnego i uniwersalnego, wymaga dziś całych zespołów i wielkich nakładów czasowych, ale stworzyć pewną myśl, pomysł na rozwiązywanie niektórych problemów związanych z cyfrowym przetwarzaniem informacji geograficznych. Mam nadzieję opracować wstępne założenia co do rodzaju niezbędnych danych i ich pozyskiwania na podstawie już istniejących zbiorów map fizycznych. Zdaje sobie całkowicie sprawę, że rozwiązanie problemu w prostej pracy jest raczej nie możliwe, ale wyzwanie jest interesujące i nakłania mnie do podjęcia starań, które mogą przynieść ciekawe rezultaty.

1.3. Wprowadzenie do problematyki projektu

Tworzenie map było zawsze interesującym wyzwaniem dla człowieka. Obecne badania historyczne dowodzą, że już wieki temu ludzie próbowali, mniej lub bardziej skutecznie, obrazować swoje otoczenie. Dzisiaj potęgą są komputery. Ich istnienie zaznacza się w każdej chwili naszego życia. Są również wykorzystywane do prezentowania opisu przestrzennego, otaczającego nasz świata. Właśnie tym zajmować się będę, tworząc projekt niniejszej pracy.

Celem, jak już pisałem, jest przetworzenie fizycznej mapy terenu na jej cyfrowy odpowiednik. By podołać temu wyzwaniu należy zrozumieć jego sens. Właściwie możemy zacząć od prostego stwierdzenia, które uprości nam wiele późniejszych rozważań, a jednocześnie da drogę lepszego zrozumienia niektórych problemów. Mapy które będziemy przetwarzać są obrazami cyfrowymi (ale nie mapami cyfrowymi) map fizycznych. Mowa tu o materiałach skanowanych, fotografowanych i w inny sposób digitalizowanych. Zaznaczam ten fakt ponieważ będzie to wyjątkowo istotna sprawa przy rozpoczęciu rozważań dotyczących pobierania danych i informacji zawartych w tych materiałach. Tak więc moim zadaniem jest zebranie jak największej ilości informacji ze źródeł, którymi będą zdigitalizowane mapy fizyczne. Następnie informacje, które otrzymam, będę starał się usystematyzować i w wydajny sposób przechować. Jeśli uda mi się to osiągnąć będę mieć otwartą drogę wielu możliwości ciekawego wykorzystania informacji przestrzennych.

Zaczynając pracę nad projektem należy zapoznać się z podstawami kartografii. Nie będę opisywał czym jest i dokładnie czym się ona zajmuje, ale w kolejnych częściach pracy zwrócę uwagę na pewne aspekty tej dziedziny. Pomogą one zacząć tworzyć sposób pobierania danych z map fizycznych – czyli jednego z wytworów kartografii.

2. Podstawowe informacje z dziedziny kartografii

Właściwie słowo „podstawowe” jest w tym miejscy chyba zbyt obszerne. Prawdę powiedziawszy siadając do pisania tego rozdziału miałem na celu jedynie zaznaczenie pewnych aspektów kartografii. Starając się jeszcze bardziej sprecyzować, można powiedzieć, iż chce zamieścić tu, tylko te informacje, które mogą mieć bezpośrednie znaczenie dla przyszłych prac nad projektem.

Sądzę, że niewłaściwie byłoby zamieszczać w tym miejscu, rozbudowanego tekstu dotyczącego historii, zastosowań, oraz ogółu teorii kartografii. Zbyteczne informacje już na wstępie, niepotrzebnie rozbudowałyby niniejszy tekst, sprawiając, iż potencjalny czytelnik znudziłby się nią bardzo szybko.

W związku z tym, na kolejnych kilku stronach zamieszczone będą jedynie podstawowe informacje i maleńka część teorii dotycząca problematyki związanej z mapami fizycznymi. Zwrócę też uwagę na tworzenie kolorów na tych mapach, gdyż to będzie miało kluczowe znaczenie dla tej pracy.

2.1. Odzwierciedlanie terenu

W tym miejscu pozwolę przytoczyć sobie, odrobinę historii i teorii, związanych z tworzeniem samych map, ich zawartością, ze szczególnym naciskiem na mapy fizyczne.

Rzeźba terenu i jej cechy odgrywają ważną rolę. Wysokość na lądzie (a na mapach akwenów wodnych – głębokość) wzbogaca treść mapy, ożywia obraz powierzchni Ziemi, jest zasadniczym elementem składowym mapy i podnosi jej estetykę.
Pierwsze pomiary wysokości związane były z odkryciem barometru rtęciowego. Tak zapoczątkowane było tworzenie map wysokościowych. Zapożyczone z metod projektowania rysunku półperspektywicznego, pozwalało za pomocą ukośnych linii obrazować rzeźbę terenu. Metoda ta połączona z uwzględnieniem oświetlenia terenu dawała bardzo plastyczny efekt rysunkowy, ale tylko na obszarach górskich. Duże obszary o niezróżnicowanej rzeźbie pozostawały na mapach białymi plamami.
Od końca XVIII wieku zaczęto stosować w przedstawieniu rzeźby terenu na mapach metodę, opracowaną przez majora armii saskiej J.G.Lehmana, polegającą na uzasadnionym w sposób matematyczny przedstawieniu oświetlenia za pomocą grubości i odstępów kresek, rysowanych na mapach zgodnie z kierunkiem spływu wody.
Dalszym rozwinięciem tej metody jest przedstawienie rzeźby terenu na mapach za pomocą cieniowania naśladującego fotografię terenu przy oświetleniu bocznym. Kolejny krok w rozwoju metody cieniowania to przedstawienie terenu za pomocą oświetlenia skośnego.
W dalszych próbach uplastycznienia rysunku poziomicowego rzeźby terenu na mapach, stosowano pogrubienie lub zagęszczenia poziomic na stromych zboczach, oraz wykorzystanie różnych odcieni poziomic w zależności od oświetlenia. Równocześnie zaczęto stosować barwoplastykę kartograficzną, wykorzystując plastyczne oddziaływanie barw na widzenie stereoskopowe ludzi.

2.2. Reprezentacja wysokości

Opracowana przez Konrada Peuckera w 1911 roku skala barw rzeźby terenu na mapach (z niewielkimi modyfikacjami stosowana współcześnie) - oparta jest na zasadzie: im wyżej, tym kolory powinny być jaśniejsze i cieplejsze (efekt bliskości), a im niżej - tym ciemniejsze i chłodniejsze.
Pod koniec XIX i w XX wieku dalszy rozwój metod przedstawienia ukształtowania powierzchni na mapach to próby połączenia metod kreskowej Lehmana i cieniowania w postaci systemu punktowego (M.Eckert, 1898). Plastyczny obraz rzeźby terenu na mapie odwzorowany jest za pomocą siatki zróżnicowanych pod względem wielkości i jasności punktów. Liczba i jasność punktów oraz ich rozmieszczenie podlegały ścisłym regułom matematycznym, wynikającym z relacji pomiędzy nachyleniem terenu i jego oświetleniem.

Z tych informacji wynika, iż mapy (zwłaszcza fizyczne – dla potrzeb tej pracy) zawierają informacje o wysokości, reprezentowane przez kolory. Plastyczne przedstawianie form terenu, będzie dla mnie bardzo istotne, gdyż to dzięki niemu możliwe będzie późniejsze tworzenie odpowiedników cyfrowych.

Dodatkowo muszę zapoznać się z pozostałymi informacjami umieszczanymi na mapach fizycznych, tak by w pełni móc zrozumieć zagadnienie i stworzyć pewien obraz, możliwych przyszłych problemów, przed których rozwiązaniem, przyjdzie mi stanąć.

2.3. Informacje zawarte na mapach fizycznych

Przyglądając się dowolnej mapie fizycznej można rozróżnić, oprócz już wspomnianych barw, cały szereg dodatkowych informacji.
Najbardziej rzucającą się w oczy, będzie dodatkowa ramka, umieszczana na drukowanych mapach, zawierającą skalę mapy, która będzie istotna dla formatu cyfrowych danych, oraz skalę barw.
Dodatkowo, możliwym problemem, będzie rozpoznanie i prawidłowa interpretacja szeregu linii. Na większości map, również fizycznych, umieszczana są siatki geograficzne, pozwalające określić położenie danych obszarów, lub obiektów. To podczas prac nad rozpoznawaniem zawartości mapy, może stanowić problem.
Podobnie rzecz będzie się miała z całą masą opisów i nazw. Sądzę, że dla moich potrzeb nie będą one potrzebne. Tworząc dane cyfrowe, nie będę projektował algorytmów, rozpoznających nazwy i teksty, gdyż to inne zagadnienie. Zastanawiam się w związku z tym, czy nie będzie niezbędne pewne przygotowanie map do procesu analizy. To jednak rozwiąże się podczas samego projektowania i implementacji.

Głównymi jednak informacjami będą kolory. Reprezentujące wysokości, będą interesować mnie najbardziej.

By dokładniej zrozumieć złożoność zagadnienia map, muszę poznać sposoby pozyskiwania danych do ich tworzenia. To z kolei pozwoli, zacząć kreować konkretniejszy obraz czekającego mnie zadania.

3. Cyfrowe modele terenu

Dane przestrzenne są w wielu opracowaniach określane jako „ogół danych opisujących przestrzeń geograficzną". Jednak dla potrzeb tej pracy, podana definicja jest zbyt szeroka. Zakłada ona przedstawianie możliwie jak największej ilości danych, dotyczących przestrzeni. Ponieważ zagadnieniem rozważanym w tej pracy, są jedynie informacje dotyczące rzeźby terenu, zawarte w prostych mapach fizycznych, należy uprościć pojęcie danych przestrzennych. W tym przypadku będzie to zestaw informacji opisujących teren, wraz z odległościami i wysokościami. Pozostałe dane będą, dla uproszczenia, albo pomijane, albo traktowane jako drugorzędne.

3.1. Źródła danych przestrzennych

Dane przestrzenne na których opierają się wszelkiego rodzaju Systemy Informacji Przestrzennej pozyskiwane są w najróżniejszy sposób. Metody dostarczania takich informacji zmieniały się wraz z rozwojem technicznym naszej cywilizacji. Marginalnie pragnę wspomnieć iż do dziś swoje miejsce mają znane od dziesiątek lat pomiary geodezyjne wykonywane w ręczny sposób przez ludzi. Jest to praca żmudna i biorąc pod uwagę dzisiejsze osiągnięcia myśli technicznej, niezbyt wydajna.

W tym miejscu zajmować się będę krótkim opisem współczesnych metod pozyskiwania danych przestrzennych. Nowoczesne urządzenia pozwoliły przyspieszyć zbieranie, gromadzenie i analizowanie nieprzebranych ilości takich informacji. Do najbardziej wyrafinowanych i spektakularnych będą należały obrazy satelitarne. Jednak ważne miejsce będą zajmować nadal zdjęcia lotnicze, zdjęcia naziemne (tzw. stereoskopowe), pomiary odbiornikami GPS, oraz automatyczne stacje pomiarowe. Pewną część poświęcić należy także najprostszym, a wspomnianym już rodzajom pomiarów, to znaczy geodezyjnym, pracom i pomiarom terenowym, a również wykorzystaniem planów i map. To ostatnie będzie bezpośrednim nawiązaniem do tematu niniejszej pracy.

3.1.1. Obrazy satelitarne

Obrazy satelitarne od pewnego czasu stały się jednym z podstawowych i szeroko wykorzystywanych źródeł danych o wysokiej rozdzielczości i wszechstronnym zastosowaniu dla Systemów Informacji Geograficznej. Z tego więc względu zdecydowałem się pokrótce opisać je na wstępie. Ich najważniejszą cechą, a jednocześnie zaletą, jest zastosowanie technologii obrazowania multispektralnego.
Wykorzystywane są zakresy kanałów widzialnych (panchromatycznych i barwnych), podczerwieni, oraz obrazowania radarowego.
Obrazy satelitarne cechuje bardzo wysokie parametry, decydujące o ich szerokiej przydatności i wszechstronnym zastosowaniu, wśród którego do najważniejszych należą:
- zasięg pojedynczego obrazu,
- wielkość piksela,
- oraz wspomniana wcześniej multispektralność;

Oczywiście podstawowe zdjęcia satelitarne, by mogły być w pełni wykorzystywane muszą przejść dokładną obróbkę i analizę. Jednak ich upowszechnienie i podane wyżej własności stanowią o niezaprzeczalnej wartości, jako źródeł danych dla systemów GIS.

3.1.2. Zdjęcia lotnicze

Kolejnym znanym źródłem danych przestrzennych są zdjęcia lotnicze. Wykonywane z samolotu za pomocą specjalnej kamery lub skanera, zawierają podobnie jak zdjęcia satelitarne, informacje radiometryczne i geometryczne.
Ich zasadniczy podział dotyczy zastosowanego filmu. Mogą to być materiały:
- nieczułe,
- czarno – białe,
- podczerwone,
- barwne,
- do fotografii spektrostrefowej;

Zdjęcia lotnicze mają wiele zastosowań i nadal są stosunkowo prostym sposobem zdobycia informacji. Ich specyfika pozwala jednak badać znacznie mniejsze obszar, niż w przypadku wykorzystania satelitów.

3.1.3. Zdjęcia naziemne (stereoskopowe)

Zdjęcia naziemne są bardzo dobrym źródłem danych przestrzennych dla małych powierzchni. Ich zaletą jest możliwość gromadzenia bardzo szczegółowych, a zatem precyzyjnych, danych geograficznych, przy użyciu specjalnie skalibrowanych aparatów cyfrowych. Ponieważ te urządzenia stały się bardzo popularne, pozyskiwanie materiałów jest teraz znacznie prostsze i tańsze.

3.1.4. Odbiorniki GPS

Global Positioning System, oparty na 24 sztucznych satelitach ziemskich, pozwala określać pozycję naziemnego odbiornika GPS, oraz dokładnego czasu i prędkości, jeśli ten jest w ruchu.
System ten może być doskonałym źródłem specyficznych danych geograficznych. W połączeniu ze specjalistycznym oprogramowaniem, o którym mowa będzie dalej, pozwala nanosić pozycje o znacznej dokładności. Dzięki temu możliwe jest zbieranie punktowych informacji w samym terenie i ich późniejsze wykorzystanie w systemach GIS.

3.1.5. Automatyczne stacje pomiarowe

Źródłem danych mogą być również automatyczne stacje pomiarowe. Umożliwiają one bezobsługowe gromadzenie, różnorodnych danych geograficznych na przestrzeni, tygodni, miesięcy, a nawet niekiedy lat. Czas ich bezobsługowej pracy związany jest bezpośrednio z pojemnością pamięci, oraz źródeł zasilania. Dane pozyskane w ten sposób pozwalają na bezpośrednie wykorzystanie w systemach GIS, dzięki numerycznemu sposobowi ich zapisu.
Ich zaletą jest długotrwałe działanie, nie wymagające interwencji człowieka, co sprawia, że mogą byś instalowane w niedostępnych miejscach, których nie brak na powierzchni Ziemi.

3.1.6. Inne źródła danych

Przedstawione w pierwszej kolejności źródła danych, posiadały pewną wspólną cechę, która była ich niezaprzeczalną zaletą. Mam tu na myśli, iż owe źródła dostarczały informacje w postaci numerycznej, a więc bezpośrednio nadającej się do wykorzystania w GIS. Jednak stanieje cały szereg sposobów pozyskiwania danych analogowych, które po analizie i obróbkach mogą doskonale nadawać się do zastosowanie w różnego rodzajach Systemach Informacji Przestrzennych.

Takim źródłem prostych danych, mogą być prace i obserwacje terenowe. Choć tradycyjne notatki na papierze i mapie, zastępowane są cyfrowymi urządzeniami, to sama istota jest niezmienna. Pomocnym źródłem informacji są również pomiary geodezyjne.

Jednak najbardziej istotnym, nie cyfrowym, źródłem danych przestrzennych, z punktu widzenia tej pracy, są mapy i plany. Ich różnorodność zapewnia obrazowanie dowolnych terenów i zjawisk geograficznych. Mnie jednak najbardziej interesowały mapy fizyczne. Więcej informacji o tych mapach zamieszczone zostało w rozdziale drugim.

3.2. Rodzaje danych przestrzennych

Ewolucja potrzeb i możliwości wykorzystania komputerów, przyniosła ogromny postęp w dziedzinie analizy różnorakich danych, w tym przestrzennych, a co za tym idzie, oprogramowania dedykowanego różnym rodzajom analiz. W rezultacie każdy współcześnie dostępny pakiet oprogramowania GIS umożliwia wykonywanie prostych i bardziej złożonych analiz danych przestrzennych, a z każdą nową wersją dostępne są coraz bardziej zaawansowane narzędzia analityczne. Ten podrozdział omówi podstawowe cztery rodzaje systemów GIS i przedstawi ich uproszczone charakterystyki.

3.2.1. GIS

Skrót GIS (ang. Geographic Information System) oznacza, w wiernym przetłumaczeniu: System Informacji Geograficznej. Zdawać by się mogło, że załatwi to sprawę/ Jednak nie jest to takie proste. Jak przeczytać można, w literaturze fachowej, ewolucja tego typu systemów doprowadziła do powstania wielu bardziej wyspecjalizowanych, jak: SIP, SIG, czy SIT. Po ich dokładniejszy opisy odsyłam do książki: „Systemy Informacji Geograficznej", autorstwa Leszka Litwina i Grzegorza Myrda.

By opisać zasadę działania GIS najlepiej przytoczyć jedną z definicji:

„GIS jest to system informatyczny zaprojektowany do pracy z danymi, które są odniesione do przestrzennych lub geograficznych współrzędnych. Innymi słowy GIS jest zarówno systemem bazodanowym z możliwością przechowywania przestrzenni odniesionych danych, jak i zbiorem funkcji przeznaczonych do przetwarzania danych" (Star J., Estes J.: „Geograpfic Information Systems: An Introduction. Prentice Hall, 1990)

Definicja ta, determinuje nasze postrzeganie zagadnienia. Będzie ono ewoluowało i zmieniało się subtelnie wraz ze specjalizacją w kierunku wykonywania pewnych zadań. W tym jednak miejscu chciałbym zaznaczyć pewien fragment z opisywanych systemów, dotyczący analizy danych, a mający olbrzymi wpływ na rozwój niniejszej pracy. Mam tu na myśli opis numerycznych modeli danych przestrzennych. Wspomniane modele tworzone są w celu dokonywania na nich analiz, bezpośrednio nie dostępnych na danych źródłowych. Poniżej przedstawione zostaną bardzo pobieżnie dwa modele: TIN i GRID, które są najczęściej wykorzystywane w celach analitycznych.

TIN

„Ang. Trianbulated Irregular Network Data Model, jest specyficznym modelem danych, w którym powierzchnia ciągła przedstawiona jest w postaci zbioru trójkątów. Wierzchołki poszczególnych trójkątów, umiejscowione są w przestrzeni, stanowiąc punkty o współrzędnych (x, y, z).” (Litwin L., Mydra G., „Systemy Informacji Przestrzennej: Zarządzanie danymi przestrzennymi w GIS, SIP, SIT, LIS, Helion 2005)

GRID

„GRID jest rastrowym modelem danych, w którym dwuwymiarowa przestrzeń podzielona jest na szereg komórek w kształcie kwadratów o identycznych wymiarach, z których każda odzwierciedla fragment przestrzeni. GRID podobnie jak mapa przedstawia charakterystykę wybranego obszaru i jego relatywną pozycję w przestrzeni. W celu jak najdokładniejszego odzwierciedlenia cech powierzchni komórki powinny mieć możliwie najmniejsze wymiery.” (Litwin L., Mydra G., „Systemy Informacji Przestrzennej: Zarządzanie danymi przestrzennymi w GIS, SIP, SIT, LIS, Helion 2005)

Przedstawione powyżej dwa modele będą w przyszłości, podstawą do rozwoju oprogramowania tworzonego w ramach tego projektu.

3.2.2. WebGIS

Zasadniczo rzecz ujmując WebGIS, to taki system, który można obsługiwać za pomocą przeglądarki internetowej. W tym układzie przeglądarka jest klientem, a dane pobierane są z serwera. Występuje więc tu struktura klient – serwer. Przesyłana przez Internet mapa jest w mniejszym, lub większym stopniu przetwarzana przez przeglądarkę.

WebGIS ma swoje wady i zalety. Niewątpliwą zaletą jest swobodny dostęp i łatwość korzystania. Wadą jest pewne ograniczenie funkcjonalności. Wydajność obliczeń ściśle związana jest z możliwościami serwera i łącza. To jednak może się zmienić. Jeśli przeglądarka służy jedynie do prezentacji wyników odpytań, bardzo ogranicza to funkcjonalność całego serwisu, gdyż obliczenia dokonywane są na serwerze. Stosowanie jednak aplikacji internetowych, tworzonych w Javie, lub innych językach

dostosowanych do potrzeb sieci, odciąży prace serwera. Część operacji i analiz wykonywana jest na komputerze klienckim, przez co znacznie wzrasta wydajność.
W związku z tym rozróżnia się dwa typy WebGIS. Oparty na technologii „cienki klient", gdzie operacje wykonuje serwer, oraz „gruby klient", gdzie część (czasem zdecydowana), operacji i analiz, wykonywana jest przez oprogramowanie klienckie.
Z całą pewnością systemy WebGIS są przyszłością. Ich stosowanie ma niezaprzeczalne zalety, a nieustanne prace nad nimi, zmniejszają ich wady.

3.2.3. 3D GIS

Trójwymiarowa prezentacja obiektów, na stałe zagościła w naszych komputerach. Wrażenie wizualizacji bardzo znacząco wpływa na nasze sposób obserwacji. Współcześnie komputerowe technologie 3D spotykamy praktycznie na każdym kroku: w filmach, animacjach, czy grach komputerowych. Technologia ta dotarła z czasem również do świata oprogramowania GIS. Pojawiły się algorytmy pozwalające budować, trójwymiarowe modele terenu, oraz poddawać je analizie. Funkcje te stały się z czasem nawet bardziej wyrafinowane, niż proste dwuwymiarowe. Powstał zatem, niejako nowy rodzaj Systemów Informacji Geograficznej, określany jako 3D GIS.
Jest on szeroko wykorzystywany w licznych dziedzinach gospodarki, a co za tym idzie, w licznych profesjach z nimi związanych. Przykładami mogą być: architektura, urbanistyka, kartografia. 3D GIS ściśle związany jest z dodatkowym oprogramowaniem, wspomagającym projektowanie, CAD.
Wykorzystanie technologii 3D GIS pozwala w praktyce przejście do płaskiej mapy papierowej do fotorealistycznych modeli trójwymiarowych i wirtualnych map.

3.2.4. Mobile GIS

Jedną z metod pozwalających na dostęp do danych w terenie jest użycie urządzeń przenośnych, oraz wykorzystanie bezprzewodowych sieci łączności. Jest to kolejna innowacja (po WebGIS), w świecie GIS dla urządzeń przenośnych. Niewątpliwie do rozwoju Mobile GIS przyczyniło się upowszechnienie odbiorników GPS. Większość zastosowań GIS w urządzeniach przenośnych wykorzystuje właśnie tą technologię, ze względu na naturalną możliwość automatycznej nawigacji na mapie, lub określania współrzędnych obiektów. Zastosowania i potencjalne zalety, są wręcz nieocenione. Rozwój urządzeń przenośnych wyposażonych w systemy znane z PC, pozwolił bardzo ułatwić pracę, jak również zwiększyć wydajność działania. Stosowane oprogramowanie, takie jak Polska AutoMapa, dynamicznie się rozwija, gromadząc olbrzymie grono zwolenników.

Dostęp do informacji przestrzennych stał się łatwiejszy i przede wszystkim przenośny. Zalety Mobile GIS docenią wszyscy, którzy często są w drodze, a potrzebują stałego dostępu do informacji przestrzennych.

3.3. Formaty cyfrowych danych

Formaty wykorzystywane podczas pracy z systemami GIS, są bardzo różnorodne. Ich mnogość zależy od źródła i przeznaczenia w których zostaną użyte. Jest to również związane z przewidywanym zapotrzebowaniem i operacjami wykonywanymi na nich. W tym miejscu zostaną przedstawione jedynie nieliczne, spośród wielkiej liczby możliwych. Pozwoli to w kolejnych krokach tworzenia projektu, w znaczącym stopniu uprościć wybór akceptowanego formatu danych, lub pomóc w stworzeniu własnego. Tak więc, przedstawicielami popularnych formatów są:

Pliki typu shape.

Jednym z formatów, najczęściej spotykanych podczas pracy z systemami GIS, oraz najłatwiejszymi w użyciu jest format shape. Pliki tego typu mogą reprezentować obiekty punktowe (miasta), liniowe (rzeki, autostrady), lub obiekty powierzchniowe (pola, gminy). Pliki zawierają dane dotyczące położenia geograficznego obiektu, oraz atrybuty, które opisują dany obiekt.
Zbiory typu shape zawierają w sobie przynajmniej trzy inne pliki (podobnie jak przy kompresji), które muszą się znajdować w tym samym katalogu. Całość składa się na opisany format.

Geobaza

Geobaza używana w powiązaniu z relacyjną bazą danych służy jako magazyn różnego rodzaju danych przestrzennych, atrybutów, oraz informacji na temat wszelkich złożonych relacji pomiędzy tymi danymi. Wykorzystując Geobazę możliwe jest utworzenie rozbudowanych modeli danych GIS, które prezentują skomplikowany układ występujący w rzeczywistości.

Warstwy ArcINFO

Innym popularnym formatem, z którym można się spotkać, są tzw. warstwy ArcINFO. Podobnie jak pliki typu shape, warstwy ArcINFO przechowują obiekty geograficzne, takie jak punkty, linie, poligony, połączone z ich atrybutami. Niektóre warstwy mogą zawierać więcej niż jeden z tych obiektów.

4. Analiza wybranych systemów informacji geograficznej

Po zapoznaniu się ze źródłami informacji geograficznych, ich potencjalnym wykorzystaniem i formatami danych cyfrowych, przyszła kolej na omówienie kilku programów, które są ściśle związane z Systemami Informacji Geograficznej.

Zasadniczo podział dokonany będzie na oprogramowanie komercyjne i bezpłatne. Ma to swoje uzasadnienie. Choć więcej jest płatnych i wyspecjalizowanych programów, to ich odpowiedniki Open Source mogą być ciekawą alternatywą, udostępniając podobne funkcje, a jednocześnie, ze względu na dokładną dokumentację, być dobrym wzorem dla aplikacji powstającej w ramach tego projektu.

4.1. Przykłady oprogramowanie komercyjnego

W chwili obecnej zdecydowana większość programów służących zastosowaniom GIS, podzielona jest na niezależne, ale współpracujące ze sobą, części. Firmy projektujące tego typu aplikacje zdecydowały się na to posunięcie ze względów praktycznych. Okazało się bowiem, że wielkie, złożone oprogramowanie, umożliwiające setki operacji, nigdy nie zostaje w pełni wykorzystane. Dodatkowo jest ono bardzo drogie. Tak, wiec tworzy się całe zbiory aplikacji, z których użytkownik wybiera, mu odpowiadające, a jednocześnie obniża koszty wdrożenia całego systemu.

4.1.1. ESRI ArcGIS

Przykładem wyżej opisanej tendencji, jest rodzina programów ArcGIS. Firma ESRI zdecydowała się tworzyć niezależne programy, które stały się swego rodzaju wzorami. Omówię tu, pokrótce trzy z nich.

Właściwą „głową rodziny" jest, rozbudowany ArcInfo. Aplikacja zawiera wiele narzędzi umożliwiających realizację dowolnego zadania w zakresie obsługi mapy cyfrowej. Dodatkowym atutem jest zgodność zarówno z platformą Windows jak i UNIX.

Kolejną aplikacją, wchodzącą w skład ArcGIS, jest przeglądarka map ArcView. Należy tu jednak zaznaczyć, iż uproszczenie opisu jedynie do funkcji przeglądania, byłoby wielkim nieporozumieniem. W obecnych wersjach jest to program funkcjonalnością mogący rywalizować z ArcInfo.
Jest zgodny z wieloma używanymi na całym świecie formatami GIS i CAD, takimi jak: DGN, DWGm DXF. Dodatkową zaletą jest wbudowanie języka VBA, który pozwala dostosowywać środowisko do potrzeb bardziej zaawansowanych użytkowników.

Ostatnim modułem będzie ArcIMS.
Program służy do dystrybucji danych i aplikacji GIS, za pośrednictwem Internetu, lub intranetu. Jego prostota i przydatność sprawiają, że jest doskonałym uzupełnieniem funkcjonalności, dwóch większych braci.

4.1.2. Intergraph GeoMedia

Czołowym produktem związanym z GIS, firmy Intergraph jest oprogramowanie GeoMedia. Jego główną zaletą jest możliwość analizowania i wizualizacji danych z różnych źródeł i standardów, bez konieczności specjalnego przygotowanie (np. importu lub konwersji). Współpracuje z takimi standardami jak MGE, MicroStation, AutoCad, czy wspomnianym już ArcGIS. Dodatkowo możliwe jest tworzenie baz dla znanych systemów bazodanowych (np. MS Access, czy MS SQL).
W skład pakietu wchodzi program GeoMedia WebMap, pozwalający na wizualizację i analizę danych w środowisku przeglądarki internetowej. Dzięki temu możliwe stało się uniezależnienie, od systemów operacyjnych komputerów klienckich.

4.1.3. MapInfo

Ostatnim z prezentowanych pakietów oprogramowania jest MapInfo. Główny produkt MapInfo Professional pozwala na tworzenie geograficznych baz danych, oraz wykonywanie ich analiz. Program pozwala pracować na warstwach, co ułatwia tworzenie i edycję baz, a przez możliwość importu z innych środowisk, realna staje się współpraca z innymi formatami. Wielką zaletą jest możliwość tworzenia wtyczek, co znacznie poszerza potencjał pakietu.

Dodatkowo wspomnieć należy o MapXtreme. Oprogramowanie to, oparte na technologii .NET pozwala prezentować mapy w Interencie.

4.2. Przykłady oprogramowania Open Source

Zaletą oprogramowania typu Open Source, czyli oprogramowania dostępnego bezpłatnie łącznie ze źródłami, jest nie tylko koszt, ale również możliwość dokładnego dobrania i dopasowania odpowiednich systemów, lub ich części do wymagań użytkownika. Międzynarodowa współpraca wielu programistów w rama projektów, wśród których jest też wiele projektów GIS, sprawia, że prace nad ustanowieniem standardów w zakresie zapisu i wymiany danych przestrzennych przybrały realne kształty.

4.2.1. GRASS

System GRASS, przeznaczony jedynie dla środowisk UNIX\Linux, to zbiór modułów, z których każdy wykonuje określone czynności. Obsługa programu, jak przystało na środowisko UNIX-owe, może być dwojaka. Wykonywana z wiersza poleceń, lub przy pomocy zintegrowanego interfejsu graficznego. Obsługa pakietu może przysporzyć problemów zwłaszcza początkującym i nie przyzwyczajonym użytkownikom. Innowacją jest specyficzna organizacja danych. Wszystkie gromadzone są w specjalnym katalogu. System rozwija się jednak dynamicznie i chętnie wykorzystywany jest w celach naukowych.

4.2.2. POSTIGS

Oprogramowanie POSTIGS jest doskonałym przykładem, świetnego systemu bazodanowego dla potrzeb GIS. Pracuje na specjalnych bazach PostgreSQL. Jego specyfikacja jest dość złożona, wiec nie sądzę, by było celowe jej opisywanie. Dla bardziej zainteresowanych polecam opis w książce: „Systemy Informacji Geograficznej", Leszka Litwina i Grzegorza Myrda. Oprogramowanie jest jednak istotne ze względu na wydajność analizy i wizualizacji, zwłaszcza dużych zbiorów danych.

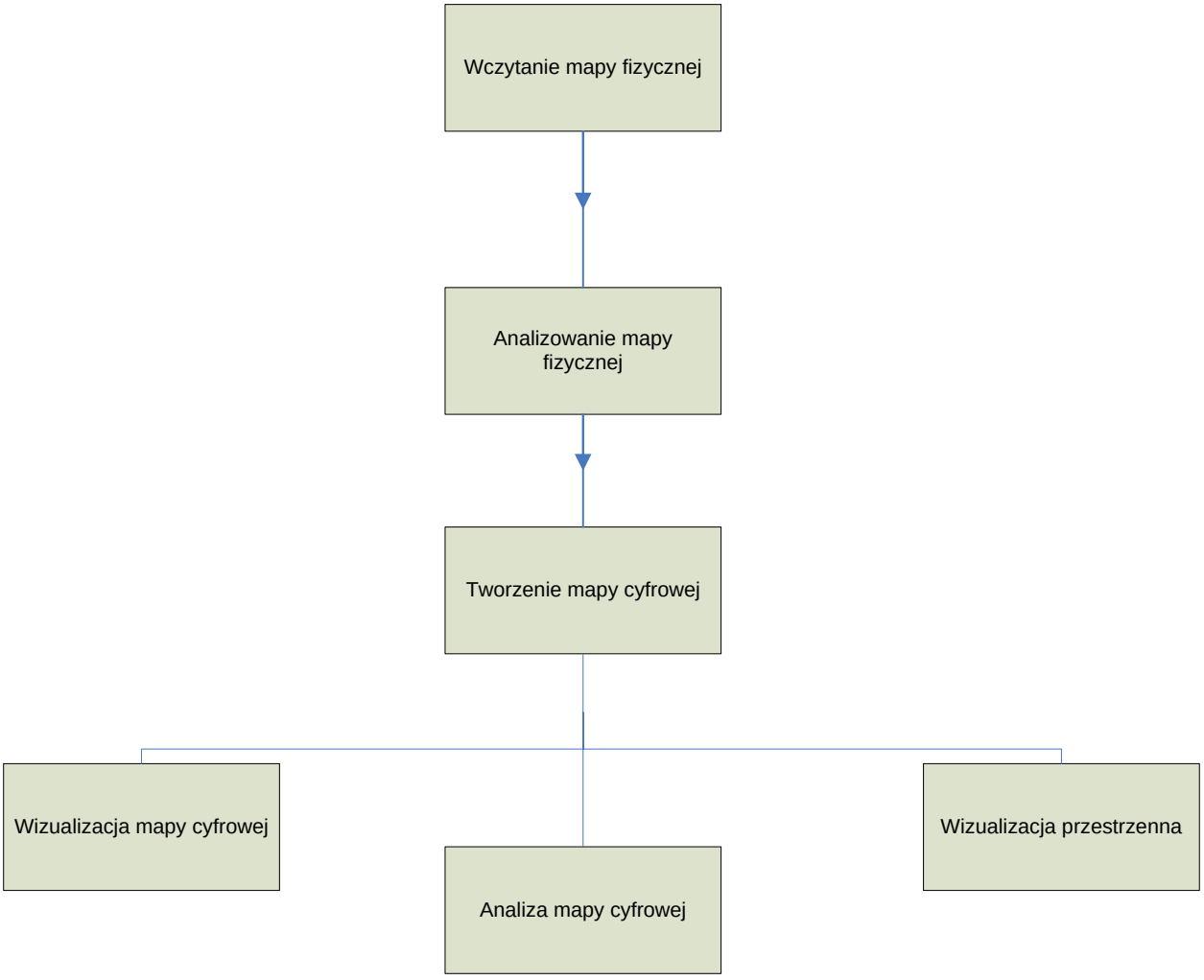
5. Projekt koncepcyjny

Realizację każdego projektu z prawdziwego zdarzenia, rozpoczyna proces tworzenia koncepcyjnego. W tym miejscu powstają założenia, które następnie będą implementowane, by w końcu powstał program.

W fazie tworzenia projektu koncepcyjnego, analizuje się dogłębnie przedmiot pracy i wyciąga wstępne wnioski. Pozwala to uniknąć szeregu błędów, a co za tym idzie, najczęściej kosztów podczas implementacji w wybranym środowisku. Stworzony odpowiednio zarys jest wyznacznikiem działania całego projektu. W tym rozdziale zajmę się analizą problematyki związanej, z tworzeniem, przechowywaniem i analizowaniem danych cyfrowych. Przedstawię główne problemy i wstępnie dokonam teoretycznych prób ich rozwiązania. Pozwoli to w kolejnych rozdziałach rozwijać przedstawione niżej myśli, by ostatecznie doprowadzić do powstania, działającego, wydajnego programu.

5.1. Ogólna koncepcja projektu.

Realizowany program ma za zadanie odzwierciedlenie w postaci cyfrowej, mapy fizyczne. Przy tworzeniu, główny nacisk będę starał się położyć na prostotę działania oprogramowania. Intuicyjny interfejs działający zgodnie z zasadami użytkowania innych aplikacji systemów rodziny Windows, powinien sprawić, że nawet pierwszy kontakt nie będzie sprawiał problemów dla większości użytkowników.
Ogólny zarys działania programu przedstawia poniższy schemat:



Rys. 01

Wczytywanie mapy fizycznej.

Choć temu zagadnieniu poświecony będzie kolejny podpunkt rozdziału, należy zaznaczyć w tym miejscu, iż część ma odpowiadać za pobieranie możliwie najbardziej precyzyjnych danych zapisanych w plikach graficznych. Program powinien obsługiwać jak największą ilość znanych formatów, kładąc nacisk na te najbardziej popularne.

Sądzę, że obsługa map bitowych i kilku wyjątkowo znanych formatów kompresji w pełni zadowoli większość użytkowników. Dodatkowo należy zadbać w tym miejscu o prosty i czytelny interfejs, ułatwiający użytkownikowi pobranie mapy do dalszej obróbki.

Analizowanie mapy fizycznej.

Wczytany plik graficzny należy poddać bardzo szczegółowej analizie. Temu zagadnieniu poświęcony będzie cały rozdział, gdyż problem jest wyjątkowo złożony. Właściwie z punktu widzenia programistycznego to właśnie ta część, będzie przykuwała największą uwagę. W tym miejscy każdy piksel mapy bitowej będzie poddawany analizie, by na jego podstawie określić dodatkowe parametry dla odzwierciedlenia cyfrowego. To właśnie w tej części działania programu kolor mapy zostanie logicznie zamieniony na wysokości, odpowiadające kształtowi terenu, który prezentowany jest na mapie.

Tworzenie mapy cyfrowej.

Po uzyskaniu, w poprzednim kroku, informacji dotyczących rzeźby terenu należy je przechować i utrwalić. Mapa wraz z wysokościami będzie reprezentowana w postaci tablicy dwu-wymiarowej, przechowywanej w pamięci operacyjnej komputera. Jednak istotna kwestią tej części działania programu będzie jej składowanie na nośnikach pamięci. Specjalnie do tego celu opracowany zostanie format danych cyfrowych, oraz odpowiednie algorytmy do jego stosowania. Temu zagadnieniu zostanie poświecone wielu uwagi. Wcześniej opisywane były znane i stosowane na świecie formaty map cyfrowych. Na podstawie wiedzy zdobytej wcześniej, stworzony będzie nowy rodzaj zapisu.

Analizowanie mapy cyfrowej

Ta część będzie odpowiedzialna za wykorzystanie map cyfrowych. Właściwie dopiero mając dane o odległościach i wysokościach w postaci cyfrowej można mówić o analizowaniu mapy. Ta część ma zawierać procedury i funkcje, które pokażą, że zdobyte dane mogą być przetwarzane, a więc dla przykładu możliwe będzie tworzenie statystyk dotyczących terenu przedstawianego przez mapę. Określanie rodzaju terenu: czy jest górzysty, czy przeważają depresje. Ta część ma za zadanie pokazać potencjalne możliwości zarówno samych map cyfrowych jak i zaproponowanego formaty danych.

Wizualizacja mapy cyfrowej

Ta część związana będzie z reprezentacją danych. W pewien sposób będzie to wsteczne przetwarzanie. By pokazać, że dane w postaci cyfrowej są odpowiednikami mapy fizycznej program na podstawie danych o wysokości i położeniu danego punktu będzie tworzył kolorowy dwu-wymiarowy obraz mapy. Na mapie tej będzie można porównać cyfrowy model z jego fizycznym odpowiednikiem.

Wizualizacja przestrzenna.

Najbardziej widowiskowym i najlepiej oddziaływującym na wyobraźnię użytkownika wizualizowaniem terenu jest jego prezentacja w trzech wymiarach. Temu problemowi poświęcony będzie jeszcze cały obszerny materiał mówiący o rodzaju silnika graficznego i technologii tworzenia struktury terenu. Istotą zagadnienia jest, jednak przestrzenna prezentacja cyfrowych danych wyodrębnionych z graficznego pliku, mapy fizycznej.

5.2. Koncepcja modułu wczytywania map fizycznych

Pierwszym krokiem w rozpoczęciu procesu analizowania mapy fizycznej i zamiany przetwarzanych danych na model cyfrowy jest właściwe wczytanie pliku graficznego. W tym miejscu należy wspomnieć o zasadzie składowania informacji w plikach graficznych. Zasadniczo rozróżniamy dwa rodzaje zapisu graficznego: grafikę wektorową i grafikę rastrową.

Grafika wektorowa, która nas nie będzie interesowała, przechowuje informacje o zawartości rysunku w postaci zbioru kształtów geometrycznych, a więc kwadratów, prostokątów, prostych i punktów. Ma ona swoje wady i zalety, o czym pisać tu nie sposób w związku ze złożonością problemu.
Drugim rodzajem, jest tzw. grafika rastrowa. Ta metoda przechowuje informacje o grafice zapisując dane o każdym punkcie osobno. Każdy punkt, nazywany pikselem posiada informacje o barwie, nasyceniu, ogólnie rzecz ujmując, o swoim kolorze.
To bardzo istotna informacje. Ponieważ, jak wcześniej zostało powiedziane, wartości wysokości danego punktu terenu, reprezentowane są przez odpowiadające im kolory, dane opisujące poszczególne piksele są informacjami o położeniu w przestrzeni danego punktu.

Oczywiście w tym miejscu, nie rozpisując się zbytnio należy powiedzieć, iż zapis rastrowy ma wiele swoich odmian. Istnieje bardzo wiele różnych rodzajów plików graficznych, tworzonych w formacie rastrowym. Oprócz podstawowego zagadnienia zapisu poszczególnych punktów grafiki, należy wspomnieć o kompresji, która to może mieć pewne logiczne znaczenie dla rozwoju myśli tego projektu.
Otóż najprostszym sposobem kompresji jest określenie przestrzeni w których dane piksele są identyczne, lub podobne (przy kompresji stratnej, gdzie pewna ilość informacji jest tracona podczas procesu kompresji) i opisanie tego obszaru, co w niektórych przypadkach znacząco zmniejsza ilość informacji.

Mając podstawową wiedzę o formatach plików graficznych, które będą przechowywały zeskanowane, lub w inny sposób zdigitalizowane mapy fizyczne, należy rozpatrzyć możliwości wczytywania tych plików.

Oprogramowanie tworzone w ramach tego projektu będzie używało standardowych okien dialogowych umożliwiających wczytywanie wybranych formatów. Najprostszym i najwygodniejszym będzie oczywiście obsługa nie skompresowanych plików graficznych, formatu BMP.

Pewnym problemem może być wczytywanie plików, formatów używających niektórych formy kompresji. Oczywiście w tym miejscu pragnę zaznaczyć, iż wykorzystanie nawet większość formatów nie jest możliwe i właściwie nieistotne dla samego meritum problemu pracy. Skupię się na najpopularniejszych formatach, a więc jak już wspomniałem na BMP, który choć już nie popularny to bardzo prosty do implementacji i znanego wszystkim standardu JPEG, który jest wyjątkowo popularny, co zapewni kompatybilność z większością plików zawierających mapy fizyczne.

5.3. Koncepcja formatu cyfrowego modelu terenu.

Wynik działania programu, będący wynikiem rozwoju projektu, powinien być przechowywany, powstaje potrzeba przechowywania ich na nośnikach danych. Nie będzie istotne rozpatrywanie jakiego rodzaju będą to nośniki. De facto mnie interesować będzie jedynie sposób zapisu i przechowywania wygenerowanych informacji i ich umieszczenie w pliku fizycznym.

Rozpatrując istniejące formaty danych dowiedziałem się, iż możliwości zapisu informacji cyfrowych jest bardzo wiele. Najbardziej popularne jest zapisywanie w postaci wektorowej (pewien wstęp do tego zagadnienia był w poprzednim podrozdziale). Ma to swoje konkretne uzasadnienie. Format wektorowy umożliwia dowolne skalowanie i przetwarzanie obrazu bez straty jego jakości. W zapisie rastrowym jest to nieosiągalne. Ponieważ jednak formatem wejściowym w moim projekcie jest format rastrowy postanowiłem zastosować podobne rozwiązanie na wyjściu. Podobne techniki są wykorzystywane do niektórych celów, więc mając zestaw informacji o takich rozwiązaniach mogę pokusić się o stworzenie własnego formatu danych. By tego dokonać muszę przeanalizować moje potrzeby i możliwości późniejszej implementacji.

W pierwszej fazie planowania formatu danych rozpatrzę jakiego typu informacje będą mi niezbędne do stworzenia prostej mapy cyfrowej.

Następnie przeanalizuję pod tym kątem dostępne i używane formaty. Ponieważ jest ich stosunkowo sporo, postaram się wybrać jak najkorzystniejszy z punktu widzenia tego projektu.

Ostatecznie dokonam wyboru zgodnego formatu danych lub zaproponuje własny wraz ze wstępnym opisem przydatnym do późniejszej implementacji.

5.3.1. Analiza stosowanych formatów danych

Przedstawione w podrozdziale 3.3. formaty cyfrowych danych, pokazują jak złożoną sprawą jest właściwe ich wykorzystanie. Jednocześnie należy zauważyć iż przedstawione formaty w zasadzie opierają się na dwóch różnych typach danych: wektorowych i rastrowych.
Opisywane dodatkowo bazy danych, wykorzystywane jako zbiory informacji wzbogacających mapę są w tym miejscu jedynie uzupełnieniem. Sądzę, że dla potrzeb tworzonego projektu, będą one niewykorzystane. Zależeć mi będzie najbardziej na wybraniu tych cech, które w miarę prosty, do późniejszej implementacji, sposób pozwolą stworzyć szybki i wydajny, własny format danych.

Przeglądając opisy stosowanych na świecie form źródłowych i docelowych map, doszedłem do wniosku, iż ich różnorodność ma za zadanie zapewnić pokrycie wszystkich możliwych kombinacji wystąpień. Analizy dokonywane są zarówno na zdjęciach lotniczych, satelitarnych, jak i na zwykłych mapach. Formaty wynikowe to bogactwo, chyba jeszcze większe. Zapewniają obrazowanie dwu i trójwymiarowe. Uwidocznianie całości terenu, jak również tylko jego wybranych elementów.

Trudno więc byłoby w tej pracy dostosować się do tak wielkiej wiedzy, gromadzonej przez pokolenia. Dla celów tej pracy najważniejsze będzie zapoznanie się z formą przechowywania źródłowych (często zeskanowanych) map fizycznych i określeniu, jakie dane, pobrane podczas ich analizy, będą zatrzymane, stanowiąc podstawę dla nowego formatu.

5.3.2. Opis wybranego formatu danych

Ponieważ mowa tu o przedstawianiu w postaci cyfrowej, modelu terenu, należy określić zakres informacji, które będą przechowywane przez proponowany format danych. Dzięki temu podczas procesu implementacji możliwe będzie pewne tworzenie kodu generującego pliki z mapami.

Pierwszą grupą niezbędnych elementów będą dane opisujące rzeźbę terenu. Ponieważ rozpatrywana będzie mapa fizyczna w postaci pliku graficznego, poszczególne piksele będą odpowiadały miejscom w terenie. Można więc przyjąć za odniesienie dwu-wymiarową tablicę, której elementy będą odpowiadały poszczególnym pikselom mapy. Tablica ta powinna natomiast, zawierać dane o wysokości danego punktu, obliczonej prze odpowiednie algorytmy programu na podstawie kolorów mapy źródłowej.

Mając te informacje mamy właściwie całkowity obraz terenu. Dodatkowo format danych powinien zawierać informacje o wielkości (długość i szerokość) tablicy. Informacje te będą miały znaczenie przy tworzeniu tablicy pamięciowej, jak również przy ewentualnych operacjach matematycznych, obliczających wielkości mapy.

Żadna mapa nie może istnieć bez skali. Posiadanie informacji o wysokości danego punktu nie przedstawia żadnej wartości bez skali mapy. Choć informacje osiągnięte dzięki obliczeniu wysokości na podstawie przedstawionego na mapie koloru będą niezależne, ich połączenie logiczne z mapą nastąpi dopiero przy powiązaniu ze skalą danej mapy. Problem ten będzie szerzej opisywany i wyjaśniany podczas implementacji oprogramowania. Tak, więc format danych zawierać musi również skalę w której mapa została wykonana. Muszą to być wartości rzeczywiste opisujące zależność miedzy pojedynczym pikselem, a jego rzeczywistym odpowiednikiem, będącym miarą odległości w terenie.

Ponieważ zasadniczo teren możemy podzielić na cztery strefy (depresje, niziny, wyżyny i góry), co będzie miało również uzasadnienie w adaptacyjnym rozpoznawaniu barw i przypisywaniu im wartości wysokości, format powinien również przechowywać informacje o zakresach wysokości dla poszczególnych stref terenu. Dzięki temu na przykład, będzie jednoznacznie można określić wartość poziomu zerowego, znajdującego się na połączenie między strefami depresji i nizin.

Proponowany format danych opisany jest graficznie na poniższym schemacie:

Rys. 02

Jak widać ze schematu, główna część informacyjną będzie przechowywać ostatni moduł formatu. Przypisane wysokości do poszczególnych punktów dwu-wymiarowej tablicy, stanowić będą faktyczny opis rzeźby terenu. Pozostałe moduły odgrywać będą funkcje dodatkowe, synchronizujące działanie formatu. Taki zapis pozwoli na proste generowanie plików wynikowych i ich szybkie przetwarzanie.

5.4. Analiza mapy fizycznej i generowanie modelu cyfrowego

Choć by stworzyć projekt, będzie niezbędne pokonanie wielu problemów, zarówno małych, jak i poważniejszych, to jednak najistotniejsze będzie opracowanie metody analizy wczytanej z pliku mapy fizycznej i wygenerowanie na jej podstawie modelu cyfrowego. Generowanie komputerowej postaci mapy to problem jeszcze dość odległy. W pierwszej fazie należy się zastanowić jak będzie rozpoznawana zawartość źródła fizycznego. Tym właśnie problemem zajmie się w głównej mierze ten podrozdział.

Początkowo planowałem tworzyć model terenu, na zasadzie znanej z historii. Chodzi mi mianowicie o triangulację. Przed laty, była to znana i ceniona metoda. Zakładała ona tworzenie siatki połączonych ze sobą w trójkąty, punktów rozmieszczonych w terenie. Wierze triangulacyjne, są już dziś zapomniane, ale gdzieniegdzie na terenie Polski można znaleźć niedobitki, lub ich pozostałości, opierające się czasowi. Tyle jeśli chodzi o historię. Chciałem ją jednak przytoczyć, by pokazać, skąd wziął się pierwotny pomysł.
Jego logiczne rozwinięcie zapoczątkowało testy nad tworzeniem siatki trójkątów na wczytanej mapie. Wierzchołki trójkątów (dla uproszczenia były to trójkąty prostokątne), miały być punktami orientacyjnymi, na podstawie których wyliczana była wysokość. Pomysł w fazie eksperymentów sprawdzał się dość dobrze. Jednak technologia nie okazała się zbyt wydajna, gdyż nawet niewielkie trójkąty, tworzące dość gęstą siatkę, powodowały tracenie, bardzo wielkiej ilości danych. Wystarczy sobie wyobrazić obszar, będący polem danego trójkąta. Staje się on niewiadomą, białą plamą, gdyż jedyne informacje dostarczane są z jego wierzchołków. Co prawda sam pomysł nie jest zły i powstała myśl by jeszcze go wykorzystać, przy tworzeniu modelu przestrzennego pewnej wizualizacji, jednak do celów tworzenia, jak najdokładniejszej mapy cyfrowej nie nadawał się, praktycznie wcale.

Analizując moc, współczesnych komputerów, łatwo dojść do wniosku, iż przechowanie opisu nawet znaczącej ilości punktów, nie będzie stanowić problemu.
Tak więc, zacząłem rozwijać myśl, której głównym wątkiem, było zbadanie i opisanie każdego piksela na analizowanej mapie.

5.4.1. Analiza pikseli mapy fizycznej

Każdy wczytany plik zawierający bitmapę, na której znajduje się mapa fizyczna, to czworobok. To banalne, wręcz śmieszne stwierdzenie, od razu pozwoliło wymyślić sposób poruszania się po obrazie, a dalej tworzyć miejsce by przechowywać informacje o nim. W tym jednak momencie, najważniejsze jest, iż dwie pętle będą poruszać się po mapie, dostarczając współrzędnych X i Y, każdego piksela. Mając już te dane mogę postarać się zdobyć informacje o zawartości mapy. Wiem, że opisany wyżej sposób, pozwoli mi dostać się do dowolnego miejsca na powierzchni mapy.

Teraz niezbędne jest by właściwie zrozumieć, jakiego rodzaju informacji poszukuje na mapie, a dokładniej rzecz ujmując, na każdym jej pikselu. Ponieważ wysokość na mapach fizycznych obrazowana jest przez kolor, będę musiał opracować sposób wydobycia informacji o kolorze, opisującym piksel. Tu pomocne informacje znajdować będę w książce „Delphi 4 dla każdego", w rozdziale 12, poświęconym programowaniu grafiki i multimediów. Książka odnosiła się do starszej wersji środowiska Delphi, ale przyznam się, iż często wracam, do świetnie opisanych prostych w swojej istocie algorytmów.

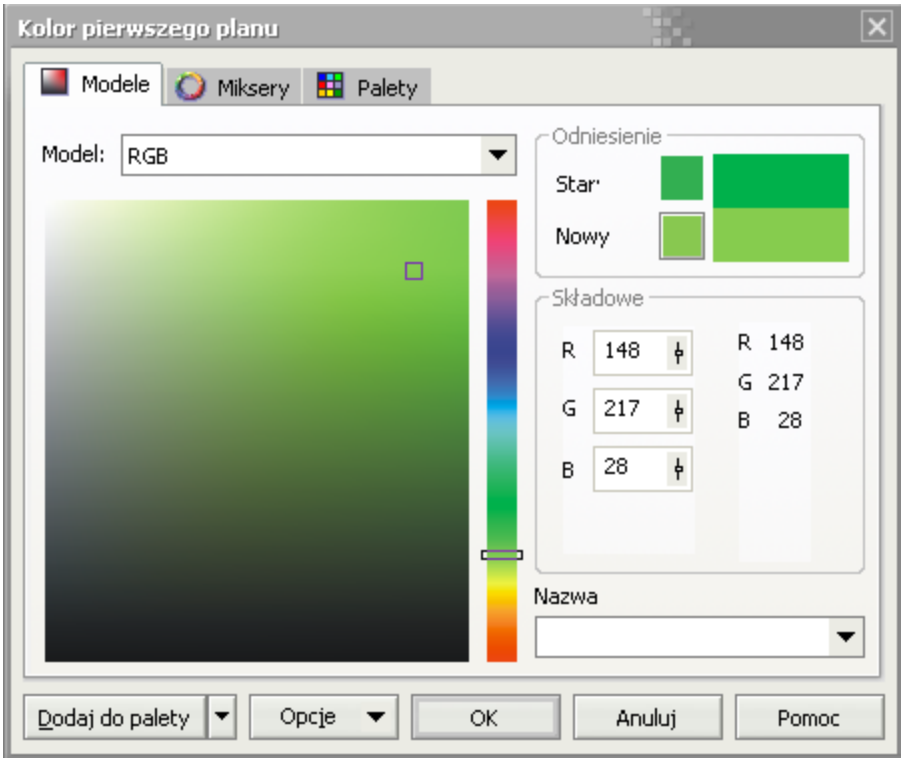
Mając dostęp do poszczególnych pikseli mapy i uzyskując informacje o kolorze mogę przystąpić do obliczania wysokości. Tym zajmę się jednak w kolejnym podpunkcie, należy jednak jeszcze rozwiązać problem opisu koloru w Delphi. Kolor opisany jest wartością całkowitoliczbową. Wspomniana książka pokazuję jak wykorzystać mechanizmy, zamiany tej wartości na znany w świecie komputerów schemat RGB.

5.4.2. Rozpoznawanie kolorów

Są to najważniejsze, dla projektu, przemyślenia. Właściwe rozpoznanie kolorów przysparzać będzie wielu problemów, których części w sposób definitywny niestety nie uda się rozwiązać, w tej w gruncie rzeczy, uproszczonej pracy. Jednak należy zacząć od początku.

W rozdziale drugim omawiane były podstawowe metody odzwierciedlania terenu. Jedną z nich było kolorowanie mapy, tak by właściwe natężenie i jasność określały wysokość. Zasada jest prosta. Najniżej będą znajdować się ciemne odcienie, zimnego niebieskiego. Stopniowo przechodzić będą w zieleń. Ta z kolei przejdzie w żółć, a ona w coraz ciemniejsze i cieplejsze czerwienie.

Tak więc wzrost wartości wysokości terenu, określają kolory coraz cieplejsze i coraz ciemniejsze, dla danego obszaru.



Rys. 03

Przedstawione okno, „zrzucone” zostało z programu Corel DRAW. Prezentuje paletę kolorów. Opisana wcześniej zasada kolorowania, odpowiada paskowi tęczy po prawej stronie, zielono - czarnego kwadratu, zaczynając od niebieskiego, a dążąc do czerwonego.

Właściwe określenie parametrów koloru jest niezbędne dla jego analizy.
W związku z tym należy podzielić uzyskany wcześniej kolor piksela, na jego składowe: R (ang. Red) – czerwony, G (ang. Green) – zielony, B (ang. Blue) – niebieski. Są one podstawą tworzenia kolorów w, chyba nie przesadzę, najbardziej znanej dla komputerów metodzie. Natężenie ich wartości (zakres od 0 do 255 w liczbach całkowitych) daje możliwość stworzenia tysięcy różniących się od siebie kolorów. To ma wielkie znaczenie dla tego projektu. Przedstawione wyżej okno, okazało się niezastąpione, gdyż przesuwając się po obszarze palety koloru, pokazywało ono, równocześnie składowe jego barw. Zasada może okazać się przydatna i dobrze by było wykorzystać ją podczas tworzenia oprogramowania. Mam tu na myśli, liczbowe przedstawianie składowych RGB danego koloru.

Oglądając zmieniające się liczby, podczas wielogodzinnych prób i przemyśleń, spostrzegłem, że opisana skala kolorów jest bardzo złożona. Jednocześnie uwidocznił się ogromny problem, dotyczący samych map fizycznych. Otóż odzwierciedlenie kolorów, zwłaszcza dokładne jest bardzo trudne. Choć wszystkim producentom i twórcom map znane są zasady ich barwienia, to praktyka przysparza wiele problemów. By dokładniej zapoznać się z problemem skontaktowałem się ze znajomym fotografem, zajmującym się grafiką komputerową od lat i będącym ekspertem w Adobe Photoshop. Opowiedział mi On wiele na temat problemów z właściwym przedstawianiem gamy kolorów. Sam, często zauważał iż kolorystyka, która wydawała się znakomita na ekranie komputera, po wydrukowaniu, traciła już na jakości, a po ponownym zeskanowaniu była, już zdecydowanie różna od projektowanego oryginału. W tym miejscu pojawił się problem z poprawnym rozpoznaniem kolorów. Poddawane analizie mapy fizyczne, to najczęściej skany, lub grafiki komputerowe pobrane z zasobów Internetu, z najróżniejszych źródeł. Zauważalny efekt problemu, to dwie mapy, przedstawiające ten sam teren, które już gołym okiem, znacząco różnią się ze względu na kolorystykę, zwłaszcza nasycenie koloru i jego kontrast.

Powstała więc myśl by przeprowadzać wstępną analizę, której zadaniem będzie rozpoznanie stref terenów. Zasadniczo sprawa jest prosta. Wiemy, że depresje oznaczane są odcieniami koloru niebieskiego, doliny zielonego, wyżyny żółtego, a góry

czerwonego. Przyglądanie się wartościom zmiennych RGB pozwoliło określić jakie składowe dominują przy danym kolorze.
Zależności przedstawia tabela:

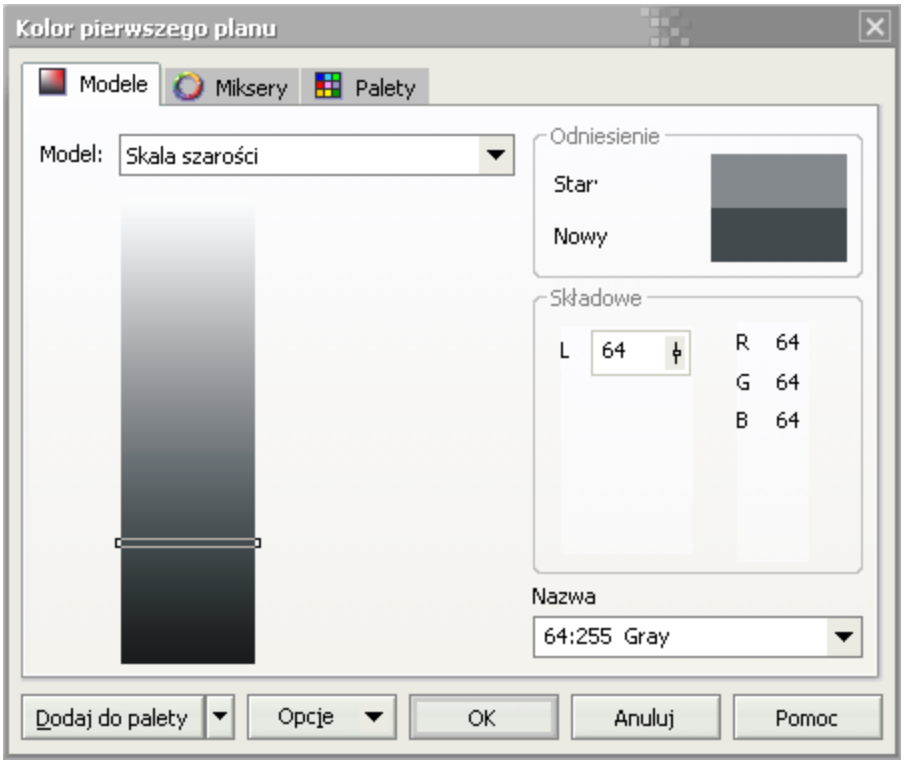
Depresje	Jeżeli niebieskiego jest więcej niż zielonego i czerwonego.
Doliny	Jeżeli zielonego jest więcej niż niebieskiego i czerwonego.
Wyżyny	Jeżeli czerwonego jest więcej niż niebieskiego, a różnica między składowymi czerwieni i zieleni jest większa niż 20
Góry	Jeżeli czerwonego jest więcej niż niebieskiego i zielonego

Jak widać z tabeli większość kolorów, a tym samym stref, łatwo określić. Pewnym wyjątkiem jest strefa wyżyn, gdyż opisujący ją kolor żółty jest uzyskany, ze złożenia czerwieni i zieleni. Wartość 20, która jest wyznacznikiem koloru powstała na drodze eksperymentalnych obserwacji zmiany wartości składowych kolorów RBG.

Tak więc określenie strefy wysokości, nie powinno stanowić już problemu. Inaczej sprawa mieć się będzie z konkretnymi wysokościami. Będą one zależne od strefy i natężenia koloru.

5.4.3. Uproszczenie wartości skali RGB

By podołać zróżnicowaniu kolorystyki i odzwierciedlania barw po wielu przemyśleniach doszedłem do wniosku, iż najlepiej byłoby uprościć trzy składowe i dopiero wówczas je analizować. Teoretyczne próby arytmetycznego ich łączenia i wyszukiwania zależności, nie przyniosły większych rezultatów.
Znów myśl, która przyniosła rozwiązanie, podpowiedział Corel DRAW. Zawiera on palety różnych kolorów i różnych systemów ich odzwierciedlania. Również skali szarości. To właśnie ona będzie podstawą upraszczania.



Rys. 04

Jak widać znacząco upraszcza to sprawę. Z trzech zmiennych RGB, odpowiadających poszczególnym kolorom składowych, powstaje jedna zmienna L będąca wyliczonym odpowiednikiem danego koloru. Jej zakres również leży pomiędzy 0, a 255, co nie zmienia stosunków miedzy poszczególnymi kolorami. Zasadniczo pomysł, opiera się na obserwacji, iż zamieniony na skalę szarości obraz pozostaje w swoim zarysie nie zmieniony. W związku z tym, jedyną do rozwiązania pozostałą przeszkodą, będzie opracowanie podczas implementacji sposobu zamiany koloru na jego „szary” odpowiednik.

5.4.4. Analiza wysokości na podstawie kolorów

Potrafiąc stwierdzić do jakiej wysokości będzie należał kolor, czy będzie to depresja, nizina, wyżyna, czy góra. Mając jednocześnie możliwość uproszczenia składowych kolorów z trzech do jednej nie pozostaje mi nic innego, jak wyliczyć wysokość. Zakładając, że posiadam informacje o zakresach wysokości dla danej strefy, mogę wyliczyć szukaną wartość. Będzie to prosta arytmetyka.
Mam wartość składowej szarości, która umiejscawia się gdzieś pomiędzy 0, a 255.
Mam również zakresy, czyli najmniejszą i największą wartość dla danego zakresu, który z kolei został rozpoznany po kolorze. Teraz wystarczy, dzięki prostym równaniom arytmetycznym wyznaczyć wartość wysokości odpowiadającą wartości ze skali szarości, by zakończyć kłopotliwy, ale najistotniejszy problem.
Rzecz jasna rozumiem, iż podczas implementacji wynikowy kod, będzie zapewne musiał dokładniej opisywać zagadnienie, tak by jego działanie było, bez zarzutu. Jednak wydaje się, iż przedstawiony tu szkielet, będzie dobrze się spisywał.

By więc rozpoznać kolor, muszę, przeszukiwać wszystkie piksele na mapie bitowej. Następnie pobrać z danego piksela jego kolor i określić go jako niebieski, zielony, żółty lub czerwony. Dzięki temu w kolejnym kroku będę wiedział do jakiej strefy geograficznej należy analizowany punkt. Kończąc muszę zamienić kolor na skalę szarości, co upraszcza mi trzy składowe barwy, do jednej, a następnie wykonując obliczenia matematyczne, wyznaczyć wartość wysokości.

5.4.5. Przechowywanie informacji o wysokości

Gdy uda mi się stworzyć sprawny algorytm obliczający wysokości, na podstawie pobranego z piksela koloru, będę musiał właściwie go przechowywać. By mieć pełen zakres informacji o mapie, będę potrzebował współrzędnych punktu i jego obliczoną wysokość. Początkowo myślałem nad tablicą rekordów, która przechowywała by te informacje. Jednak doszedłem do wniosku, że prosta tablica, która swoją wielkością odpowiadała by mapie fizycznej, mogłaby przechowywać jedynie informacje o wysokości. Punkt o danych współrzędnych na mapie, byłby reprezentowany przez punkt w tablicy, a jego zawartością byłaby wysokość.

Takie rozwiązanie znacznie uprości i zmniejszy zakres tablicy, a tym samym przyspieszy jej przeszukiwanie, analizowanie i ogół późniejszych zadań wykonywanych na niej.
Ostatecznie, również dzięki temu, możliwe będzie łatwe składowanie całej tablicy do pliku fizycznego i zapisywanie rezultatów pracy przyszłej aplikacji.

Choć cały program będzie składał się z wielu algorytmów, których większość będzie wymagała pokazania twórczości umysłu, to najważniejszy został tu przedstawiony. Pozwoli on stworzyć to, co jest głównym celem projektu, a więc zamienić mapę fizyczną na cyfrowy odpowiednik.

W kolejny podrozdziale omówię planowane dodatki, które mają wzbogacić całość tworzonego oprogramowania.

5.5. Pozostałe funkcje systemu

Choć tak jak napisane było na poprzednich stronach, głównym zadaniem programu będzie poprawne rozpoznanie zawartości mapy fizycznej i jej zamiana na cyfrowy odpowiednik, to bez dodatkowych funkcji i możliwości, praca będzie zbyt uboga, by mogła zaciekawić użytkownika. Dodatkowe możliwości, pozwolą pełniej skorzystać z zaproponowanego formatu danych, a tym samym pokazać jego poprawność.

Choć zapewne większość rozwiązań ewoluuje podczas ich implementacji, to niezbędna jest pewna myśl, temu przewodnicząca. Ten niewielki w rozmiarach podrozdział, będzie właśnie temu poświecony.

5.5.1. Generowanie modelu przestrzennego terenu

Myśląc o dodatkowym wzbogaceniu projektu, nie przychodziło mi początkowo na myśl nic bardziej efektownego i przemawiającego do wyobraźni użytkownika, jak prezentacja terenu w postaci trójwymiarowej. Przedstawienie rzeźby powierzchni w postaci, zwłaszcza siatki, pozwala zapoznać się, z jej złożonością. Dodatkowymi aspektami mogą być pokrywanie siatki powierzchniami o odpowiedniej kolorystyce: bądź prezentującej wysokości, bądź rodzaje gleby.
By jednak tego dokonać, w czasie implementacji będę musiał znaleźć odpowiednie narzędzia, a przede wszystkim wydajny silnik graficzny, który zapewni mi prosty i wydajny mechanizm tworzenia obrazu.

Sama zasada tworzenia siatki powierzchni, wydaje się nie być zbyt złożona. Ponieważ opis terenu, zawarty będzie w tablicy pamięciowej, wystarczy z niej skorzystać. Najważniejsze będzie opracowanie metody właściwego pobierania punktów z tablicy. Pamiętam, przy tym, że będą to już trzy współrzędne X,Y,Z, umiejscawiające dany punkt w przestrzeni.
Tak więc mechanizm powinien działać następująco:
Dwie pętle, przeszukiwać powinny całą tablice i pobierać współrzędne. Odpowiednie procedury łączące punkty w figury geometryczne (trójkąty, lub czworokąty), będą tworzyły siatkę. Następnie, będzie ona wypełniana i odpowiednio kolorowana. Dzięki temu powstanie trójwymiarowy model terenu.

Jednocześnie, dla większego efektu, trzeba będzie zaprojektować mechanizmy, umożliwiające manipulowaniem stworzoną bryłą. Mam tu na myśli, choćby podstawowe działania, jak obracanie, skalowanie i przesuwanie położenia kamery.
Jednocześnie nasunęła mi się drobna myśl, która może jednak mieć ciekawy efekt. Przemieszczanie kamery wokół terenu, powinno być ograniczone, tak by teren widziany mógłby być jedynie z boków i góry. To zapewni dość naturalne obrazowanie i narzuci użytkownikowi pewien sposób prezentacji.

Podczas praca nad tworzeniem terenu, chciałbym również rozwinąć kilka procedur związanych z tworzeniem odpowiedniego środowiska. Oświetlenia, tworzenia wody itp. Jednak tego typu rozwiązania pozostawię do czasy implementacji projektu.

5.5.2. Analiza mapy cyfrowej

Po wygenerowaniu mapy, jej zapisaniu na dysk, oraz umożliwieniu jej prezentacji, przyjdzie moment, by dodać proste algorytmy pozwalające zademonstrowanie potencjału formatu, lub właściwie samych danych cyfrowych.
Pewną tego formą jest już sama wizualizacja przestrzenna, która analizuje wysokości, tak by móc umieścić dane punkty w przestrzeni.
Jednak w tym miejscu mam na myśli nieco inne działania.

Na przykład sprawdzenie jakie wysokości są największe w całej tablicy, pozwoli odnajdować szczyty wzniesień lub gór. Analogicznie sprawa będzie się miała z wyszukiwaniem głębin.

Dodatkowo zliczanie powierzchni terenu, ilości wód i lądów, pozwoli tworzyć proste, ale konkretne statystyki. Pozwoli to w sposób znacznie bardziej czytelny, a przede wszystkim dokładny opisywać prezentowany teren.

Analizą może być również wyszukiwanie błędów, które mogą powstać podczas tworzenia samych danych przestrzennych. Nieprawidłowe rozpoznanie wysokości, może być wykryte poprzez wykonywania złożonych procedur analizy. Pozwoli to wykonać jeszcze precyzyjniejsze modele, pozbawione rażących defektów.

Tak więc całość projektu została obmyślona. Choć zdaję sobie doskonale sprawę z tego, że podczas implementacji mogą zrodzić się nowe problemy, których w tej chwili nie jestem w stanie przewidzieć, to sądzę, że zgromadzenie podstawowej wiedzy ma kluczowe znaczenie dla końcowego efektu.

6. Realizacja projektu

Po przeanalizowaniu całości problemu, zapoznaniu się z istniejącymi rozwiązaniami, oraz określeniu własnych wymagań, co do tworzonego oprogramowania, przyszła kolej na konkretną realizację projektu. Zdobyta wiedza, pozwoliła wytyczyć prawdopodobny przebieg pracy, a to z kolei pozwoli zaopatrzyć się w niezbędne materiały i oprogramowanie.

Tak więc, w pierwszej kolejności zapoznałem się z dostępnymi językami programowania, wybrałem jeden, który z różnych względów najbardziej mi odpowiadał, a następnie zgromadziłem dodatkowe narzędzia i zasoby.

Przy, już dość zaawansowanych pracach nad projektem doszedłem jednak do wniosku, iż zebranie wszystkiego w jednym programie, będzie niewygodne, o czym będę jeszcze pisać. W związku z tym powstała myśl o rozdzieleniu wielu operacji, między trzy niezależne, ale jednocześnie powiązane ze sobą aplikacje. Ostatecznie powstałe moduły rozrosły się i osiągnęły zamierzony poziom.

To właśnie tym decyzjom i pracom nad nimi, będzie poświęcony niniejszy rozdział. Opisując zasadę działania i same moduły, będę dążył do rozbudowanego zakończenia, jak najdokładniej opisując algorytmy i ich funkcjonowanie, dla poszczególnych aplikacji.

6.1. Wybór środowiska programistycznego

Jest to bardzo istotne zagadnienie. Wybranie odpowiedniego dla potrzeb, języka programowania, zapewni wydajność aplikacji, jej uniwersalność i wygodę pracy nad projektem. Niestety, prawda jest jednak taka, iż opisany wybór, jest najczęściej kompromisem. Należy zdecydować przy jakich stratach, osiągnie się jak najlepsze rezultaty. Programy, barwne i ciekawie prezentujące się wizualnie, najczęściej będą znaczących rozmiarów i pochłaniające niesamowicie zasoby systemowe. Jest to ściśle związane z wyborem środowiska w którym będzie się pisało. Do tego jednak dojdziemy. Najpierw należy się jednak, zapoznać z dostępnymi językami.

6.1.1. Analiza znanych języków programowania

By wybrać odpowiednie dla siebie środowisko pracy musiałem przejrzeć kilka znaczących języków programowania, wiele mniej znanych, jak również całą masę tak zwanych generatorów aplikacji.
Te ostatnie już na starcie można odrzucić. Choć pozwalają w szybki i efektowny sposób tworzyć ciekawe aplikacje, to ich wadą jest wyspecjalizowanie. Mam tu na myśli, iż generator aplikacji działa na zasadzie przewidzianych przez twórcę zdarzeń, a rezultat, jest w pewnym sensie ich zespoleniem. Choć większość z nich umożliwia dodawania prostych skryptów, to jest to zdecydowanie niewystarczające do sprostania celom, które przede mną stały.
W dalszej kolejności przyglądałem się niektórym językom. Choć znane i cenione, jak na przykład Java, której maszyna wirtualna zapewnia zgodność, z wieloma systemami operacyjnymi, to jednak chyba nie do końca będzie odpowiadała moim potrzebom. Ja szukam języka, właściwie całego środowiska, który jak najwydajniej byłby dostosowany do środowiska Windows.
Właściwie moja uwaga skupiła się jedynie na dwóch:
- Borland C++ Builder
- Borland Delphi

Pierwszy, jak sama nazwa wskazuje, tworzy aplikacje w szybkim i wydajnym języku C++, który umożliwia tworzenie niewielkich w rozmiarach kompilacji. Efekt ten osiągnięto poprzez, dołączanie tylko wymaganych kontrolek i bibliotek.
Drugim jest Delphi, również firmy Borland. Wersja 7 jest mi znana ze studiów, gdyż była prze kilka lat podstawą do pracy i zaliczania przedmiotów związanych z programowania. Ma pewną zaletę. Język Object Pascal jest nieco prostszy, niż C++.

Oba środowiska umożliwiają tworzenie złożonych aplikacji dla systemów rodziny Windows, a to mi jest potrzebne.
Zasadniczo rozpatrywałem oba języki pod kątem:
- znajomości danego języka i łatwości jego wykorzystania,
- możliwości środowiska i bogactwa podstawowych komponentów (przyśpiesza to proces tworzenia okien, a tym samym całej aplikacji, na przykład wykorzystanie komponentów dialogów),

- ilości dostępnej literatury i pomocy on-line,
- ewentualnych, dodatkowych rozszerzeń i ergonomii pracy,

Zarówno Delphi, jak i Builder udostępniają szereg podobnych komponentów, a ich interfejsy są do siebie zbliżone. Jest to z całą pewnością spowodowane tym, iż są wytworem jednego producenta, który starał się je ujednolicić.
Ponieważ pomoc, dla obu środowisk, jest po porostu znakomita, zarówno dołączona do programów, jak i dostępna w Internecie, wykorzystanie danego języka nie byłoby problemem.
Końcowy wybór uzależniony został od przyzwyczajeń i wygody pracy.

6.1.2. Uzasadnienie wybrania Delphi

Tu sprawa jest naprawdę prosta. Po przeanalizowaniu języków programowania (nie było nawet sensu myśleć o generatorach aplikacji), wybrałem Delphi. Choć jego zastosowanie ma kilka wad, sądzę, że świetnie się sprawdził, a tworzona aplikacja osiągnęła zamierzony poziom.
Dla mnie bardzo istotne, a wręcz przeważające było, swobodne poruszanie się w tym środowisku. Delphi w wersji 7 jest swego rodzaju klasycznym językiem programowania. Udostępnia już wiele nowoczesnych narzędzi i struktur, jak obiektowość, czy wykorzystane tu z całym rozmachem dynamiczne tablice, ale nie jest już tak nowoczesny jak jego kolejne wcielenia, ściślej związane z platformą .net.
Nie będzie tak elitarnym i precyzyjnym językiem jak Borland C++ Builder, którego kod, bez większych przeszkód byłoby można przenosić na inne systemy operacyjne, a jego wynikowe aplikacje, mogą być znacznie większe niż wygenerowane przez Builder. Jednak wygoda i doświadczenie skłaniają mnie do niego.

Jak napisałem na początku tego rozdziału, decyzja ta jest kompromisem.
Jednak ostateczny efekt okazał się dość ciekawy, a łatwość poruszania się po środowisku Delphi, była wielkim plusem.

6.1.3. Wybór dodatkowych narzędzi

W trakcie realizacji projektu niezbędne okazują się dodatkowe narzędzie i zasoby. Należy pamiętać, iż sam program, to nie tylko kod źródłowy (przynajmniej nie w tym wypadku) to również szereg dodatków, takich jak grafika, ikony, czy dźwięki. By móc wykorzystać te multimedia, należy użyć dodatkowego oprogramowania.
Zasadniczo problem sprowadza się do dwóch typów oprogramowania:
- edytora tekstów,
- programu graficznego.

Edytor tekstów niezbędny jest przy tworzeniu jakichkolwiek dokumentów, również niniejszego. W samym projekcie, znajduje się sporo wyjaśnień i notatek, których skład najwygodniejszy jest na komputerowym programie do edycji tekstu.
Niewątpliwie niezastąpionym okazał się Microsoft Word, choć równie dobrze, można by wykorzystać bezpłatny OpenOffice.
Tak wiec jedno dodatkowe narzędzie zostało wybrane i wykorzystane.

Kolejnym niezbędnym software jest program do przetwarzania grafiki rastrowej. Nie skorzystam z grafiki wektorowej ponieważ nie mam gdzie. Format który opracowuje, choć właściwie, jest pokrewnym do grafiki rastrowej, w wykorzystaniu bardziej przypomina grafikę wektorową, ponieważ piksele opisane są właściwością (wysokość), a dowolne przekształcenia, rozwiązywane są za pomocą matematycznych obliczeń.
Rozterka wiązała się właściwie z wyborem między, tylko dwoma, ale za to niezwykle znanymi produktami: Adobe PhotoShop, a Corel DRAW Photo-Paint.
Obydwa programu pracują na grafice rastrowej i oba do moich prostych potrzeb nadają się, aż nadto. Bez większych i moim zdaniem, nie potrzebnych wynurzeń, napisze, iż zastosowana w projektach grafika wykonywana była w Corel Photo-Paint.
Program umożliwił stworzenie tła i przystosowanie zdjęcia oka, do potrzeb projektu, oraz wykonania innych prostych grafik.

Dodatkowo przy realizowaniu programów w ramach tej pracy, wykorzystane zostały inne zasoby, choćby tak drobne, jak ikony, które pobrane zostały w niewyczerpanych zasobów Internetu.

6.2. Podział projektu na moduły logiczne

Wstępne prace rozpoczęte wiele miesięcy temu, rozwijały kolejne pomysły i ewoluowały tworząc nowe podejścia do zagadnienia tego projektu. Początkowo projektowana była jednolita aplikacja, która miała przejąć, wykonywanie wszystkich zamyślonych operacji, na siebie. Jednak wkrótce po rozpoczęciu implementacji stało się jasne, że potrzebne będą mechanizmy upraszczające kod, gdyż ten znacząco, w miarę dodawania nowych procedur i funkcji, rozrastał się. Choć to działanie nie było niczym nieoczekiwanym, to jednak znacząco utrudniłoby w przyszłości swobodne poruszanie się po kodzie źródłowym. Powstał wiec pomysł rozbicia kodu na mniejsze części i ich powiązanie ze sobą. Początkowo rozważałem tworzenie dynamicznie dołączonych bibliotek, ale zasadniczo nie zmieniłoby to całości problemu. Projekt został więc rozbity na oddzielne programy. Początkowo od głównego projektu odłączony został moduł wizualizacji. Prace nad rozbudową kodu generującego obraz przestrzenny w OpenGL bardzo go „rozdmuchały" i rozdzielenie stało się niezbędne. Programy jednak współpracują ze sobą. Moduł wizualizacji został tak zaprojektowany, by można było go wywoływać spod okna programu generatora, wraz z parametrem mówiącym o tym jaki plik ma otworzyć. Oczywistym również powiązaniem jest sam format danych, który obowiązuje we wszystkich trzech aplikacjach.
Ostatnią częścią, było stworzenie, modułu do analizy powstałych map cyfrowych. Pracujący oczywiście na wspomnianym formacie i powiązany z modułem wizualizacji, stał się integralną częścią projektu.

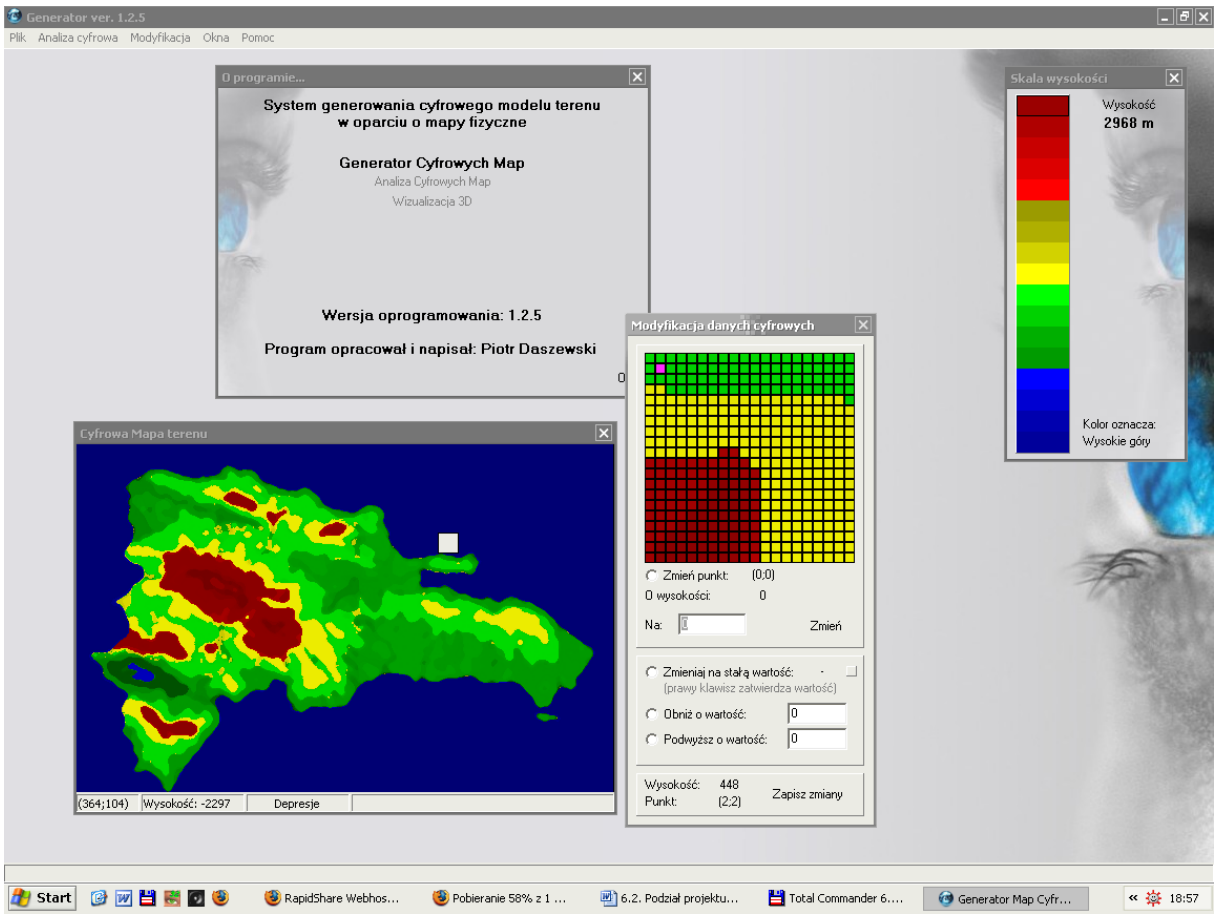
Poniżej znajdą się krótkie opisy poszczególnych aplikacji, składających się na całość pracy. Pozwoli to wstępnie zapoznać się z problematyką i możliwościami danych modułów. Więcej informacji o ich działaniu znaleźć można w kolejnych podrozdziałach.

6.2.1. Moduł generowania map cyfrowych

Główna aplikacją, składającego się z trzech modułów projektu, jest Generator. Jest to najbardziej rozwinięty program z całej trójki. Zawiera pomysłowe algorytmy, mające na celu wykonanie głównego założenia pracy, czyli właściwego rozpoznania mapy fizycznej i stworzenia jej modelu cyfrowego. Całość ma być w założeniu jak najbardziej zautomatyzowana. Analiza map zawierających setki tysięcy punktów to złożona sprawa i musi być wykonana jak najdokładniej. Jednocześnie ma zapewnić wygodę pracy i swego rodzaju przyjemność, która z całą pewnością zostanie doceniona przez przyszłych użytkowników.

W związku z tym należy stworzyć przyjemny wizualnie interfejs, zasadą działania i stylistyką nie odbiegający od znanego ze środowiska Windows. Dodatkowo przez całość projektu, głównie jednak w tym module, przewijać się będzie przyjemny i delikatny motyw kobiecego oka. Mam nadzieję, że wyróżni ono tworzone aplikacje i urozmaici zwykłe okna systemowe.

Całość prezentuje się następująco:



Rys. 05

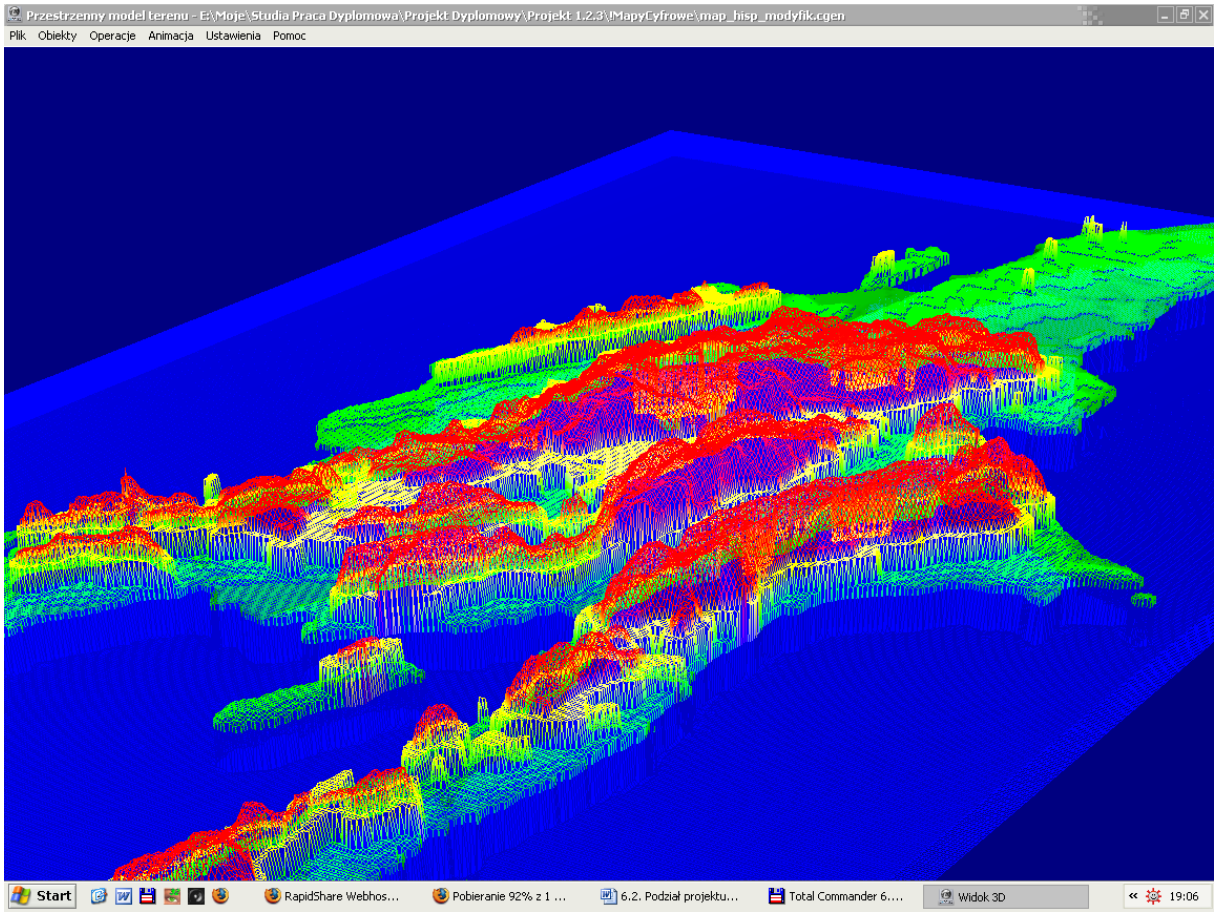
6.2.2. Moduł wizualizacji przestrzennej

Jak już na początku tego podrozdziału było stwierdzone moduł wizualizacji przestrzennej powstał, niejako na drodze ewolucji całego projektu. Początkowo udostępniał jedynie generowanie prostej, jednobarwnej siatki, bez dodatkowych wzbogaceń, jednak z czasem stał się w pełni funkcjonalną aplikacją.

Trzeba przyznać, że choć ze względów programistycznych, jest on znacznie mniej złożony, niż moduł generatora, to jest zdecydowanie bardziej efektowny. Jest to właściwie jego niezaprzeczalna siła. Całość oparta na OpenGL prezentuje świetną grafikę i ciekawi użytkownika. Po stworzeniu mapy, dosłownie korci by przyjrzeć się jej w postaci 3D. Ma to swoje całkowite usprawiedliwienie, gdyż łatwiej nam jest wyobrażać sobie teren widząc jego modelowaną przez program powierzchnię.

Dlatego główny nacisk przy tworzeniu tego modułu położony był na jak najdoskonalszą formę generowania obrazu, przy oczywiście prostym jego wykorzystywaniu.

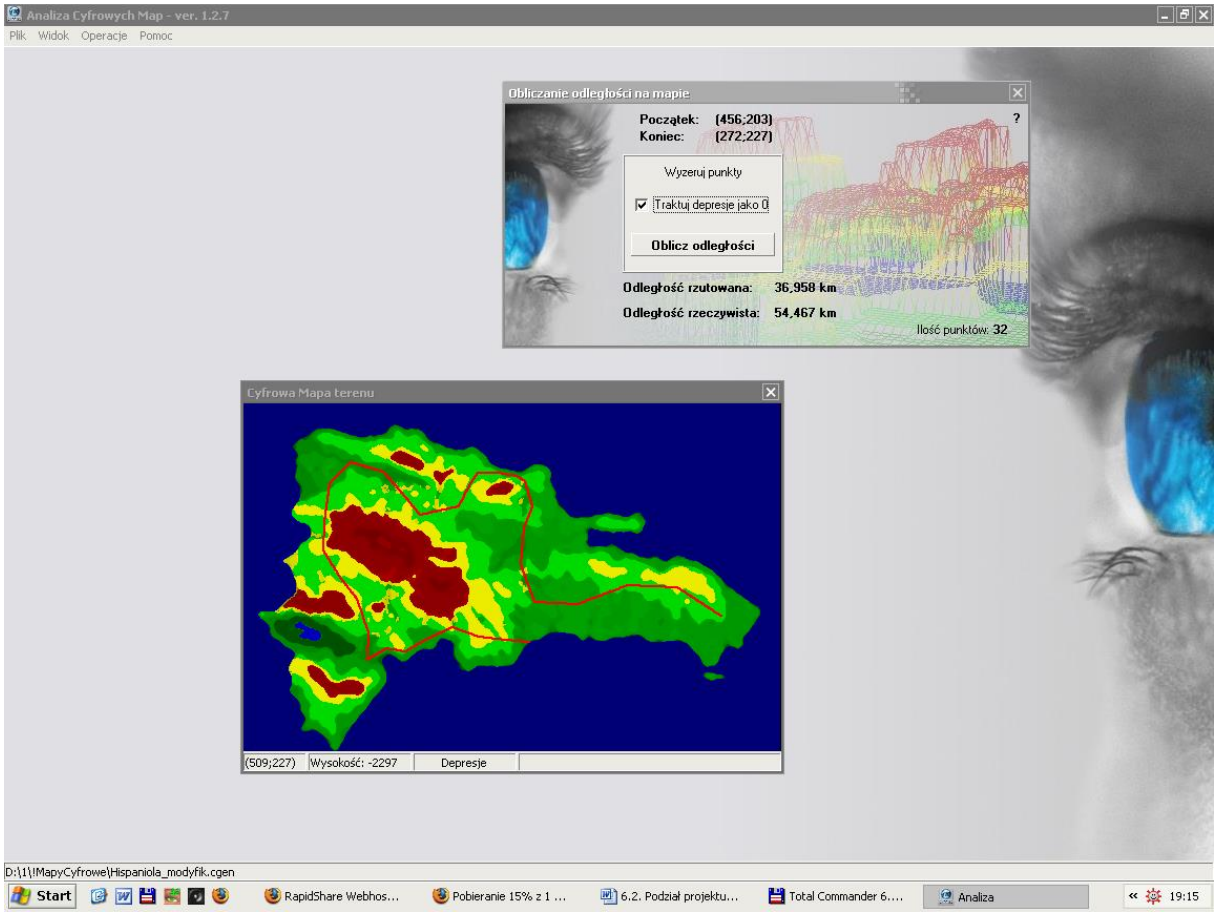
Program jednak najlepiej prezentuje się sam:



Rys. 06

6.2.3. Moduł informacyjny

Ostatnią aplikacją wchodzącą w skład projektu jest moduł analizy tworzonych map cyfrowych. Choć nie jest tak reprezentatywny jak jego poprzednik, to ma swoje niezaprzeczalne zalety. Stworzone, przez generator mapy, są właściwie tylko modelami i prostymi odpowiednikami map fizycznych. Bez dodatkowego oprogramowania nie mogą reprezentować nic więcej, a przecież jak zostało powiedziane na początku pracy, technologia tworzenia cyfrowych systemów informacji przestrzennej, daje wiele możliwości. W związku z tym powstał moduł analizy. Udostępnia on niewielką, ale dopracowaną gamę narzędzi tworzących statystyki i wykonujących proste obliczenia na podstawie wczytanych map cyfrowych.
Przedstawiony poniżej zrzut ekranowy prezentuje obliczanie trasy opisanej przez użytkownika na mapie:



Rys. 07

Trzy aplikacje i wiele, czasem złożonych obliczeń.

6.3. Implementacja generatora map cyfrowych

Po stworzeniu odpowiednich zarysów teoretycznych i zgromadzenia niezbędnej wiedzy, przyszedł czas na tworzenie konkretnych programów. Ich podział został teoretycznie i częściowo praktycznie przemyślany, a główny nacisk położony na moduł do generowania cyfrowych map, zgodnych z zaplanowanym standardem formatu danych.

Tworzenie odpowiedzialnych za rozpoznawanie zawartości mapy fizycznej, algorytmów przysporzyło wiele problemów. Wynikały one z istoty samych map, a opisane zostały podczas przemyśleń teoretycznych.

„Generator", bo tak nazwałem moduł do tworzenia cyfrowych map terenu, został napisany po wielu miesiącach modyfikacji. Teoretycznie opracowane algorytmy zostały poprawiane pod wpływem praktycznych testów, które wykazały ich błędy i nieprawidłowe działanie.

Tu przedstawione zostaną opisy i wyjaśnienia zasad działania ostatecznych wersji algorytmów, tworzących ten moduł.

6.3.1. Algorytmy analizy map fizycznych i wysokości

Opisana tu grupa algorytmów jest właściwie sercem całego projektu. To właśnie ona odpowiada za poprawne rozpoznanie i przetworzenie mapy fizycznej. Pozostałe elementy, są niejako dodatkami, umożliwiającymi testowanie i operowanie na wygenerowanych mapach.
Najważniejszy z tych algorytmów, odpowiada za analizę poszczególnych pikseli mapy fizycznej i rozpoznanie koloru. Jednak sprawę należy zacząć omawiać po kolei.

W pierwszym kroku powstała procedura umożliwiająca wczytanie z pliku mapy przedstawiającej fizyczny opis terenu. Wstawienie odpowiednich kontrolek dialogów i przypisanie im niezbędnych właściwości to tylko formalność. Wczytany z pliku obraz zostaje przekazany do komponentu Image, którego odpowiednie oprogramowanie zapewnia właściwą prezentację mapy. To umożliwia nam bezpośredni dostęp do mapy, a dokładniej, do każdego jej piksela. Opisany komponent Image, znajduje się w oddzielnym oknie programu, wiec odwołanie do niego, a tym samym do jego zawartości, odbywać się będzie przez wywołanie jego macierzystego okna.
Właściwie już w tym momencie powstał pierwszy prosty, algorytm określający z jaką strefą wysokości mamy do czynienia. Jest to bardzo ważne, gdyż dotyczy już bezpośredniego rozpoznawania kolorów, a przyda się bardzo, przy analizowaniu mapy. Wywoływanie algorytmu przypisane jest obsłudze myszki. Gdy porusza się ona po komponencie Image, określana jest jej pozycja (X i Y) i pobrany jest kolor piksela który ona wskazuje. Następnie zmienna przechowująca kolor, „rozbijana” jest, przez specjalną procedurę na składowe, dając opis R, G, B – czyli czerwonego, zielonego i niebieskiego składnika koloru. Opis zasady tworzenia kolorów znajduje się przy rozważaniach teoretycznych. Realizację tego zapewnia kod:

```
RGB:=Mapafiz.Image1.Canvas.Pixels[x,y];
 R := GetRValue(RGB);
 G := GetGValue(RGB);
 B := GetBValue(RGB);
```

Gdzie pierwsza linijka to pobrany kolor (zmienna nazywa się RGB) i jego podzielenie na składowe czynniki.

Analizując zależności między natężeniem poszczególnych składowych, a należy pamiętać, że natężenie jest wyrażone wartością w zakresie 0-255, można stwierdzić z jakim przedziałem wysokościowym mamy do czynienia.
Teoria, tworzenia map fizycznych, dla przypomnienia, mówi, że im wyżej, tym kolor jest ciemniejszy i cieplejszy. Tak więc strefy przechodzą od głębokich i ciemnych odcieni niebieskiego, przez zielony, żółty, aż do coraz ciemniejszych czerwieni.

Algorytm sprawdza którego składnika jest więcej i określa zakres wysokości:

Depresje	Jeżeli niebieskiego jest więcej niż zielonego i czerwonego.
Doliny	Jeżeli zielonego jest więcej niż niebieskiego i czerwonego.
Wyżyny	Jeżeli czerwonego jest więcej niż niebieskiego, a różnica między składowymi czerwieni i zieleni jest większa niż 20
Góry	Jeżeli czerwonego jest więcej niż niebieskiego i zielonego

Jak widać dość problematyczną strefą były wyżyny. Wynika, to ze specyfiki koloru żółtego. Zawiera on w sobie znaczną część składowej czerwieni, co utrudniało identyfikację. Jednak doświadczalne testy wykazały, że badanie różnic składowych czerwieni i zieleni zapewni poprawność rozpoznania tej strefy.

Całość analizy wykonuje kod:

```
if (B>R) and (B>G) then kolor:='Depresje';
if (G>R) and (G>B) then kolor:='Doliny';
if (R>B) and (R>G) then kolor:='Góry';
pom:=R-G;
 if pom<0 then pom:=-pom;
 if (R>B) and (pom<20) then kolor:='Wyżyny';
```

Dla sprawdzenia, a potem dokładniejszego działania programu, wyniki działania algorytmu zostały przedstawiane użytkownikowi, w pasku stanu, na dole okna.

Gdy już wiedziałem jak rozpoznawać dane obszary wykorzystałem teorie opracowaną specjalnie na potrzeby tego projektu. Dzięki niej powstał algorytm, wywoływany odpowiednim poleceniem z menu górnego, który dokonywał analizy poszczególnych pikseli (jak powyżej) i przypisywał im konkretne wartości wysokości.

Pierwszą częścią jest stworzenie odpowiedniej tablicy pamięciowej.
Delphi, umożliwia tworzenie tak zwanych dynamicznych tablic. Okazały się w tym przypadku niezastąpione. Klasycznie tworzy się tablice o znanych podczas programowania wymiarach. W tym jednak przypadku nie mogę określić jakiej wielkości ma być tablica, a chciałbym by miała tyle elementów ile ma pikseli wczytana mapa fizyczna. Dzięki temu będę mógł, każdemu pikselowi mapy przypisać obliczoną wysokość i przechować ją w tablicy. Tworzenie dynamicznych tablic odbywa się następująco (chce w tym miejscu to opisać, gdyż będzie stosowane we wszystkich modułach, nawet kilkakrotnie).
Deklaruje zmienną reprezentującą dynamiczną tablice:

```
tabcyf : array of array of integer;
```

W moim przypadku jest to zmienna globalna, działająca w obrębie całego programu. Pozwoli to korzystać z niej, w sposób pośredni lub bezpośredni, przez wiele procedur i funkcji całego modułu.
Następnie podczas rozpoczęcia analizowania mapy fizycznej przydzielana jest dla tej tablicy odpowiednia pamięć z zasobów komputera. Realizuje to kod:

```
SetLength(tabcyf,w+1,h+1);
```

Gdzie, zmienne w i h to długość i szerokość mapy bitowej. Dodane jedynki zapewniają mi zgodność między rozmiarami bitmapy i tablicy, ponieważ współrzędne mapy zaczynają się od (0;0), a pierwsze miejsce w tablicy, określa (1;1);
Dzięki temu w pamięci komputera mam zarezerwowaną tablicę dokładnie takiej wielkości jak wczytana z pliku mapa fizyczna. Teraz mogę przejść do jej analizy.

Tworzę pętle która „skacze” po kolejnych pikselach przechowywanych przez komponent Image. Poniższy kod zapewnia pełne przejście po bitmapie, pobranie kolorów i ostateczne zapisanie wysokości do tablicy:

```
for y:=0 to h-1 do
 Begin
  for x:=0 to w-1 do
   Begin
    RGBI:=Mapafiz.Image1.Canvas.Pixels[x,y];
    R := GetRValue(RGBI);
    G := GetGValue(RGBI);
    B := GetBValue(RGBI);
    ……
    tabcyf[x+1,y+1]:=wysokosc;
  End;
 End;
```

Zaprezentowany szkielet analizuje wszystkie piksele, pobiera z każdego kod koloru, dzieli go na składowe, przelicza, a ostatecznie zapisuje obliczoną wysokość do tablicy. Mam gotowy schemat rozpoznawania wysokości.
W tym momencie zastosowana będzie teoria (odsyłam do punktu 5.4.) dotycząca zamiany koloru na skalę szarości i obliczenie wysokości na podstawie zakresów wysokości i ich wartości.

Prosty kod dokonuje rozpoznania zakresu wysokości, dokładnie na tej samej zasadzie, co opisana na początku podrozdziału i przypisaniu specjalnej zmiennej odpowiedniej wartości, reprezentującej zakres.
Gdy zakres jest już określony kolor z pobranego piksela zostaje zamieniony na skalę szarości:

```
R2:=((R*RConst + G*GConst + B*BConst) shr 8);
```

R2, jest to zmienna (liczba całkowita) będąca wartością zastępującą każdą składową koloru. Będzie niezbędna do obliczeń, a jej zastosowanie wynika z teoretycznych rozważań.

Następnie dla danego zakresu dokonywane jest obliczenie konkretnej wysokości:

```
if opis='zie' then
      Begin
       pomR:=(maxzie-minzie)/255;
       pomR:=pomR*R2;
       pomV:=pomR;
       pom:=pomV;
       wysokosc:=pom;
      end;
```

Podany kod oblicza wysokość dla nizin, co zapewnione jest przez identyfikowanie zmiennej „opis". Zasada wyliczania wysokości jest prosta. Wiem, że zakres wartości koloru to 0-255. Jednocześnie znam zakresy wysokości (o czym będzie pisane w dalszej części tego podrozdziału – zmienne minzie i maxzie). Proste porównanie i odpowiedni kod, zapewniający działania arytmetyczne, wyszukuje mi odpowiadającą natężeniu danego koloru wartość wysokości.

W ten sposób określiłem wysokości dla każdego piksela mapy bitowej i wprowadziłem je do specjalnie w tym celu tworzonej tablicy. Trzeba jednak zaznaczyć, że sam kod procedury jest znacznie bardziej rozbudowany. Dotyczy to obsługi błędów i wymuszania, prawidłowego postępowania, przez użytkownika, przy generowaniu mapy cyfrowej. Po wykonaniu pętli, wywoływana jest procedura generująca obraz, już mapy cyfrowej i określana są komunikaty ewentualnych błędów. Jednak sam sens rozpoznawania i tworzenia wysokości dla mapy cyfrowej zamyka się w tych kilku stronach.

6.3.2. Składowanie danych do plików fizycznych

Już podczas projektowania formatu danych, przewidywałem jak będą wyglądały pliki zawierające informacje o mapie. Zebrane w pamięci operacyjnej komputera dane, to znaczy: tablica z wysokościami, informacje o zakresach, wymiarach mapy i jej skali, należy docelowo składować. Sensu nie trzeba wyjaśniać. Po co tworzyć program do generowania map, skoro nie można ich zapisać. Teoretyczny zarys formatu już istnieje. Należy go tylko wykorzystać.

W głównym oknie programu, zgodnie z przyjętymi w środowisku Windows, zasadami w menu Plik, Zapisz mapę cyfrową, znajduje się procedura składująca dane z pamięci operacyjnej do plików fizycznych.

Dodanie odpowiednich kontrolek, było formalnością, zarówno dla odczytu jak i zapisu. Pliki map cyfrowych tworzone w ramach projektu mają rozszerzenie .CGEN.

Są to jednak zwykłe pliki tekstowe i można je otwierać, oraz edytować w dowolnym programie do obsługi tekstów, nawet w Windows-owym Notepad.

Procedura tworzy pliki i wpisuje do nich dane w następującej kolejności:

- szerokość mapy,
- długość mapy,
- skalę mapy – to jedyna wartość rzeczywista,
- zakresy wysokości (w poszczególnych, kolejnych wierszach, zapisywane jest minimalna i maksymalna wartość danego zakresu)
- końcowa część pliku to zapis całej tablicy (każda wartość znajduje się w oddzielnej linii);

Format został przyjęty dla całości projektu i będzie wykorzystywany we wszystkich modułach.

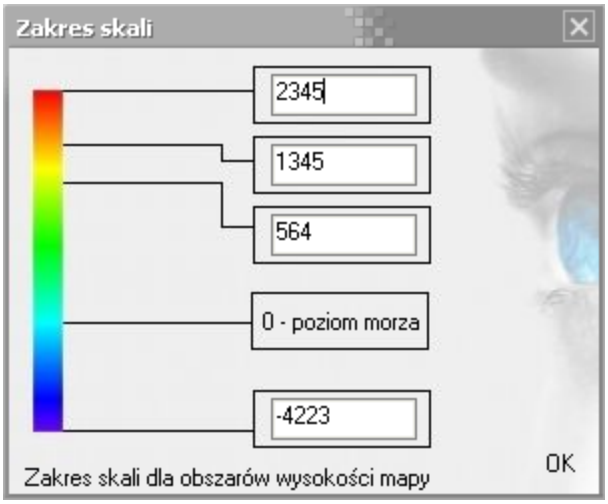
Odczytanie informacji z pliku, odbywa się w sposób odwrotny. Plik otwierany jest w trybie, tzw. „do odczytu”, w przeciwieństwie do trybu „nadpisywania”, stosowanym podczas tworzenia, a dane pobierane są po kolei, linijka po linijce.

Ilość danych podstawowych, to znaczy długość, skala itp. jest stała, wiec zmienne przypisywane są zawsze tak samo, natomiast tablica tworzona zostaje dynamicznie, a odczytane z pliku wiersze, wprowadzone do jej odpowiednich komórek.

6.3.3. Skalowanie map i zakresy wysokości

Ważnym problemem który należało rozwiązaćm było odpowiednie przygotowanie zakresów wysokości i skali mapy. Te pierwsze jak już wiemy, były niezbędne do obliczania wysokości, na przykład wyżyn. Natomiast skala mapy pozwoli wiernie odwzorowywać rzeczywisty układ terenu przy wizualizacji i okaże się niezbędna przy obliczeniach wykonywanych w module informacyjnym.

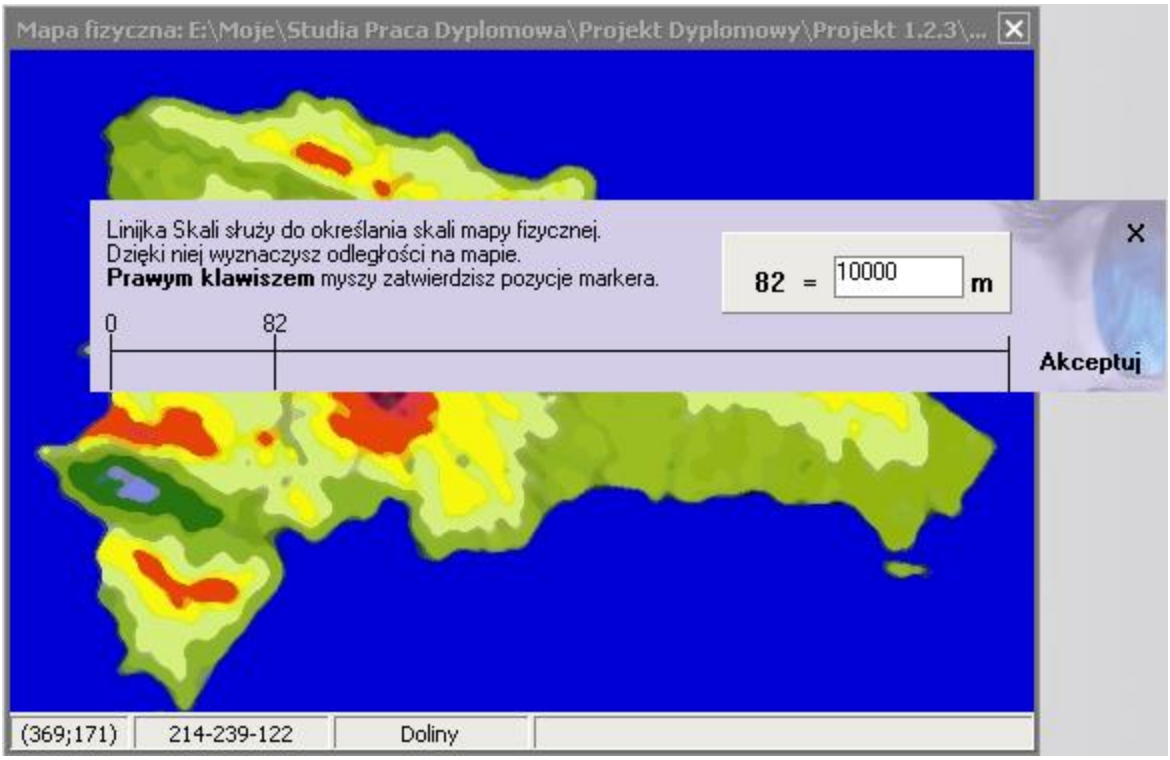
Obydwa problemy zostały rozwiązane najprostszą z możliwych dróg. Uzupełnienie tych danych powierzone zostało użytkownikowi. Pierwotnie zamysłem było wyliczanie wysokości automatycznie, ale okazało się to bardzo trudne do wykonania, a ze względu na złożoność niektórych map, wręcz niewykonalne.
Do określania minimalnych i maksymalnych wysokości dla danego zakresu, zostało stworzone specjalne okno, prezentowane poniżej:



Rys. 08

Tęcza kolorów, w sposób intuicyjny podpowiada użytkownikowi z jakim zakresem ma do czynienia. Podaje on wartości liczbowe danych poziomów, a drobne algorytmy sprawdzają ich wystąpienie i poprawność. Na przykład, gdy użytkownik poda maksymalny poziom głębin, jako wartość dodatnią, program wyświetli komunikat o nieprawidłowości wprowadzonych danych.
Potwierdzenie kliknięciem przycisku OK, spowoduje zamknięcie okna pomocniczego (które można później swobodnie wywołać) i zapisaniu danych o zakresach.

Podobnie rzecz się ma w przypadku skalowania mapy.
Stworzone zostało specjalne okno, w którym zmodyfikowano obsługę formatek, tak by można było nim swobodnie poruszać:



Rys. 09

Służy ono do określenia ilości pikseli na mapie i przypisania im wartości wyrażonej w metrach. Skala powstaje przez podzielenie ilości metrów i ilości pikseli. Uzyskana wartość mówi nam ile metrów odpowiada jednemu pikselowi.

W ten prosty sposób rozwiązane zostały problemy dostarczenia podstawowych danych, niezbędnych przy dokonywaniu obliczeń wysokości, a tym samym procesu generowanie cyfrowych map terenu.

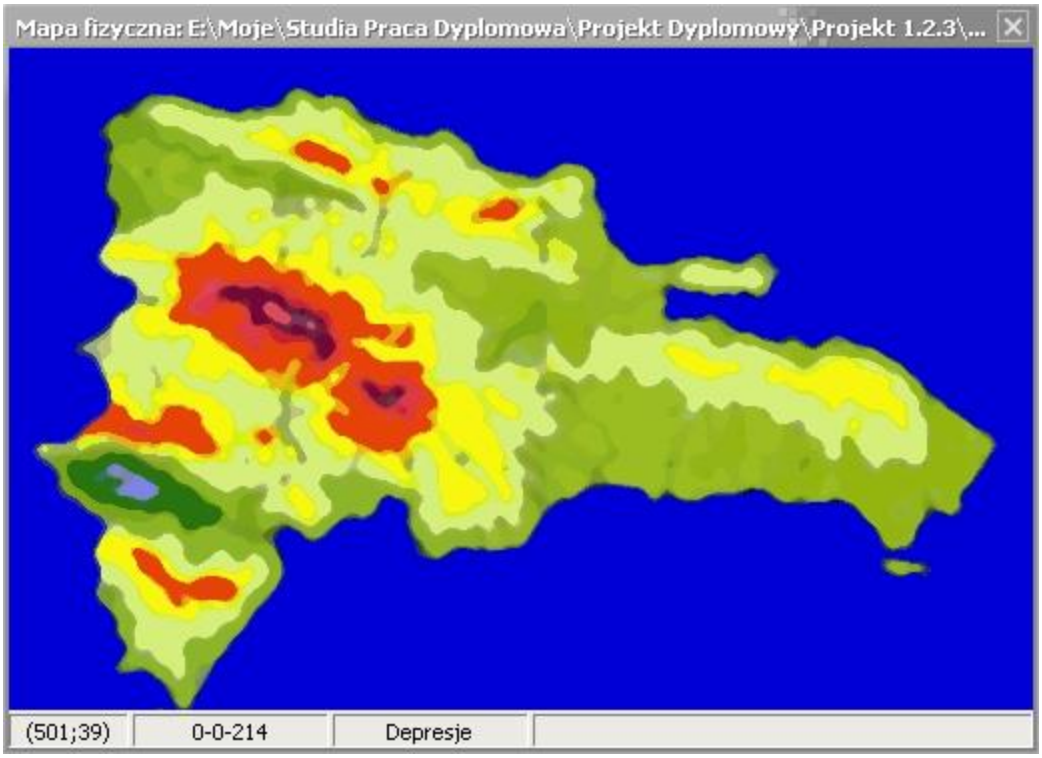
6.3.4. Kolorowanie cyfrowych map terenu

Ponieważ przetrzymywanie wysokości w tablicy, nie jest dla użytkownika widoczne, powstała myśl by uzyskane dane wizualizować w sposób odpowiadający tworzeniu map fizycznych. Należy jednak pamiętać, że choć efekt końcowy przedstawionych poniżej działań i algorytmów jest podobny do zwykłej mapy, to jest on ujednolicony i kolorystyka nie stanowi już problemu, o czym była mowa w podrozdziale 5.4. Jest jednakowa dla wszystkich map, niezależnie od tego jak wyglądały ich źródłowe odpowiedniki.

Zasadniczo problem kolorowania mapy na podstawie wysokości jest bardzo prosty w rozwiązaniu. Skorzystałem z już nabytej wiedzy. Mając wysokość, muszę sprawdzić do jakiego należy przedziału, w jakim zakresie wysokości się znajduje, a tym samy jakich kolorów muszę użyć. Następnie obliczyć wartość natężenia odpowiedniej składowej koloru, wygenerować kolor i umieścić piksel o tym kolorze na komponencie przedstawiającym dwuwymiarowy obraz mapy cyfrowej.
Całość opiera się jak zwykle na dwóch pętlach, pobierających z tablicy informacje o wysokości. Wysokość zapisywana jest do odpowiedniej zmiennej, a składowe kolory zerowane – ma to swoje uzasadnienie, ponieważ są one przechowywane w pamięci, pozwala to skasować stare ustawienia.
Wewnątrz pętli znajdują się warunki sprawdzające do jakiego zakresu wysokości należy dana wartość. Jest to istotne, ponieważ dany zakres determinuje jaką składową koloru będę obliczał. Dla przykładu powiem, iż dla nizin, których kolorystyka opiera się na zielonym i jego odcieniach, obliczaną składową będzie G (z gamy RGB).
Obliczenie następuje przez porównanie, analogiczne do tego które przedstawiane było na początku podrozdziału. Wysokości mają pewien zakres (min i max), natomiast kolor określa się przez wartości od 0 do 255. Prosty algorytm arytmetyczny oblicze wartość danej składowej. Dla depresji jest to B – niebieska, dla nizin G – zielona, dla gór R – czerwona. Wyjątkiem są wyżyny, których składowe to G i R. Obliczone i podstawione wartości składowych zostają zamienione na kolor, który z kolei zostaje przypisany pikselowi na mapie.

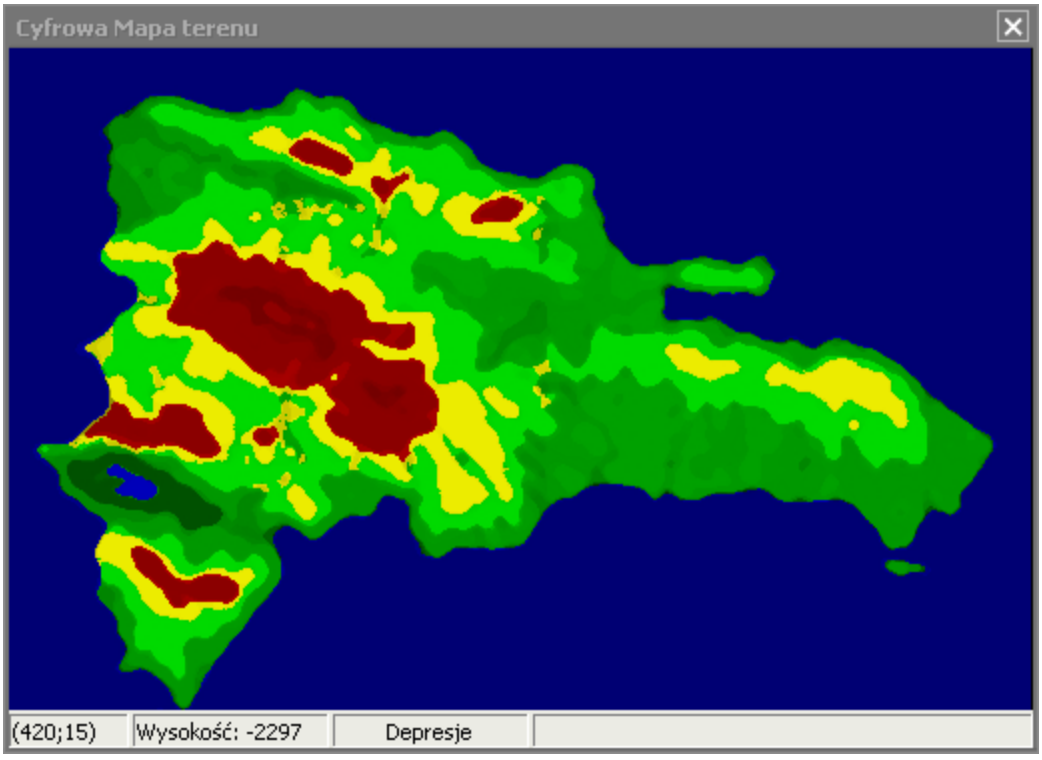
Poniżej prezentowane są dwie mapy. Obie przedstawiają ten sam teren.

Pierwsza jest obrazem zeskanowanej mapy fizycznej:



Rys. 10

Kolejna przedstawia tą samą mapę po przeanalizowaniu jej i wygenerowaniu dwuwymiarowego odpowiednika cyfrowego:



Rys. 11

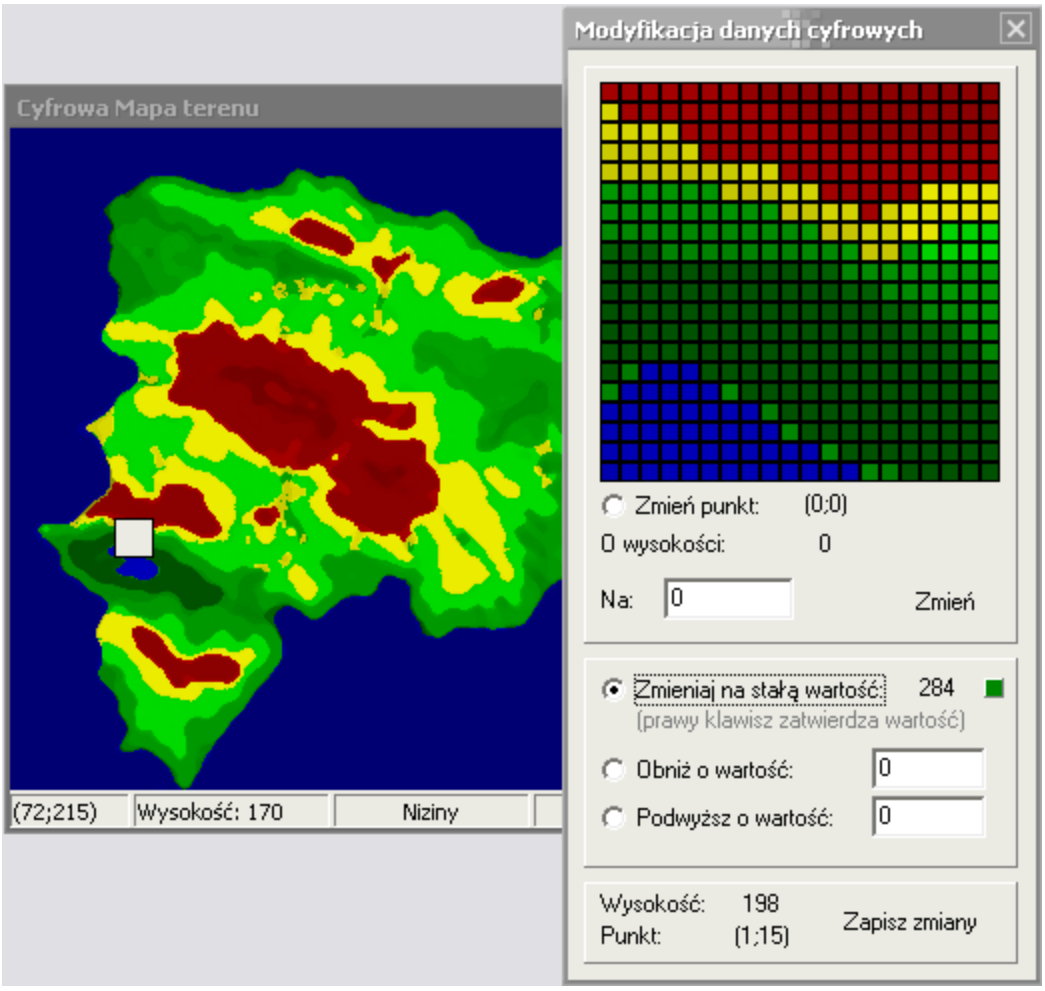
Jak widać choć zarys i struktura terenu nie uległy zmianie, a całość została poddana gruntownej przebudowie, mapa prezentuje się znakomicie. Zmieniona kolorystyka jest już stała dla wszystkich map, co sprawia, iż gromadzone zbiory są jednolite pod względem jakości i formy.
Procedury odpowiedzialne za kolorowanie sprawdziły się znakomicie i w podobnej konstrukcji zostaną użyte w kolejnych modułach, zarówno dla obrazowanie dwu, jak i trójwymiarowego.

6.3.5. Modyfikacja map cyfrowych

Choć algorytmy rozpoznawania koloru, obliczania wysokości i kolorowania mapy cyfrowych spisują się bardzo dobrze, to nie są wolne od błędów, których niestety w tym stadium rozwoju projektu nie dało się usunąć.
Problemem są „niechciane" piksele na mapie fizycznej. Jeśli na terenie nizinnym, gdzie przeważa kolor zielony wystąpi piksel o barwie w której przeważa składnik czerwony, kolor może zostać rozpoznany jako należący do wysokości większej niż otoczenie, a w rezultacie zapisany jako na przykład wyżyna. Choć bezpośrednio tego nie widać, zwłaszcza przy obrazowaniu dwuwymiarowym, już podczas modelowania przestrzennego powstają iglice – błędnie rozpoznane wysokości. Problem ten może dotyczyć zwłaszcza plików bitmap poddanych kompresji, lub źle zeskanowanych.
By temu zaradzić zaprojektowałem odpowiednie procedury i funkcje do ręcznego modyfikowania cyfrowych map. Zostały one zawarte w specjalnie do tego celu stworzonym oknie, uruchamianym z górnego menu „Modyfikacje".

Pomysł polegał na tym, by do nowego okna przenieść wybrany fragment mapy i umożliwić poddanie go modyfikacjom. Realizacja tego pomysłu opierał się na skopiowaniu fragmentu tablicy głównej do małej pomocniczej tablicy, stworzonej na potrzeby modyfikacji. Tablica pomocnicza ma rozmiar 40x40. Tak wiec maksymalnie mogę przekazać 1600 pikseli. Jednak pomysł rozwinął się bardziej i tworząc dodatkowe zmienne mówiące ile pikseli przekazuje, mogę modyfikować trzy różne wielkością obszary: 10x10, 20x20 i 40x40. Daje to możliwość powiększania wybranego obszaru i dokładniejszego wglądu w jego zawartość.

Po zaznaczeniu na mapie cyfrowej obszaru, odpowiednie algorytmy przechwytują zakres do tablicy pomocniczej i wyświetlają jej zawartość w nowym oknie:



Rys. 12

Poszczególne punkty tablicy pomocniczej są pobierane podczas wykonywania się dwóch pętli, a współrzędne przeliczane, na odpowiednie obszary w komponencie Image, znajdującym się w oknie Modyfikacji danych cyfrowych. W zależności od użytej wielkości obszaru, obliczane są różne wielkości pojedynczych komórek. Zapewnie to specjalnie napisana procedura „Odswiez", która analizuje wysokości znanymi już metodami, a następnie tworzy na komponencie Image, kwadrat o odpowiednio wyliczonych bokach:

```
kolor:=RGB(R,G,B);
    Image1.Canvas.Brush.Color:=kolor;
    xp:=x*wielkoscx; yp:=y*wielkoscx;
    Image1.Canvas.Rectangle(xp,yp,xp+wielkoscx,yp+wielkoscx);
```

Jak widać z powyższego rysunku okno modyfikacji umożliwia sporą gamę prostych narzędzie umożliwiających i przyspieszających modyfikacje pobranego obszaru.

Moim zdaniem najwygodniej jest używać opcji „Zmieniaj na stałą wartość".
Umożliwia ona zamianę danych (klikniętych) pikseli na wartość określoną przez użytkownika. Ponieważ najczęściej modyfikacja polega na dostosowaniu do otoczenia, wystarczy kliknąć prawym klawiszem by pobrać wartość wysokości jako podstawę do modyfikacji.
Następnie klikając już lewym klawiszem, bądź trzymając go i poruszając się po obszarze, zmieniamy wartości wysokości.

Oprócz tej opcji dostępne są również:
- obniżanie o daną wartość, co powoduje przeliczenie klikanego piksela i zmniejszenie jego wartości o dany stopień,
- podwyższanie o daną wartość, analogiczne jak wyżej,
- ręczne nadanie wartości wysokości dla danego piksela;

W tym miejscu chcę zauważyć, że zmiany są przeprowadzane na tablicy pomocniczej, a podgląd ich wyników, możliwy jest dzięki ciągłemu odświeżaniu komponentu Image. Daje to również jeszcze jedną korzyść. Wprowadzone zmiany nie muszą być zaakceptowane. Jeśli z jakiś względów zmiany nie są korzystne wystarczy zamknąć okno i całość odejdzie w niepamięć.

Tak więc generator cyfrowych map, może być ciekawym i wydajnym programem, do tworzenia modelów terenu, zgodnych z zaproponowanym formatem. Umożliwia on pełną swobodę działania, automatyzację rozpoznawania mapy fizycznej i ewentualne modyfikacje błędów powstałych podczas tego procesu.

6.4. Implementacja modułu wizualizacji przestrzennej

Początkowo moduł ten był integralną częścią generatora. Dostępny w dodatkowym oknie, głównej aplikacji nie umożliwiał zbyt wielu opcji. Jego rozdzielenie pozwoliło znacząco go rozwinąć, jak również nakierować projekt na ostateczny tor. Tak zrodziła się niezależna aplikacja o nazwie „Wizualizacja", realizująca modelowanie cyfrowego terenu w przestrzeni. By tego dokonać musiało powstać i ewoluować wiele drobnych, a czasem nawet dość złożonych algorytmów. Ich ogólny opis i zasada działania znajdują się poniżej. Realizują one takie działania, jak tworzenie przestrzennej siatki na podstawie danych o położeniu i wysokości danego punktu, wypełnianiu tej siatki i jej odpowiednim kolorowaniu. Dodatkowo, aplikacja tworzy środowisko do wizualizacji terenu i udostępnia niewielką gamę narzędzi do pełniejszego przedstawiania jego rzeźby i obrazowania zjawisk fizycznych.

6.4.1. Wybór silnika graficznego

W przypadku realizowania tego projektu wybór silnika graficznego, odpowiedzialnego za tworzenie przestrzeni trójwymiarowej, był niezwykle prosty. Na zajęciach z grafiki komputerowej używaliśmy, wraz z kolegami z grupy, OpenGL do tworzenia prostych figur i animacji. Sprawa była więc jasna. Dzięki zajęciom uzyskałem niezbędną podstawową wiedzę, którą podczas tworzenia aplikacji znacząco poszerzyłem, czego efektem będzie ostateczna jakość programu. Dodatkowym atutem okazało się mnogość przykładów zastosowania OpenGL w Delphi, znajdujących się w Internecie.
Podstawę do tworzenia przestrzeni zaczerpnąłem z takich stron jak:

- www.delphi3d.net
- www.sulaco.co.za

Obydwie strony są anglojęzyczne, ale opisy i przykłady są naprawdę znakomite.

Całość okazała się dobrym wyborem i pozwoliła w znacznym stopniu zrealizować założenia projektu.

6.4.2 Algorytmy wizualizacji terenu

Zasadniczy szkielet kodu tworzącego przestrzeń w OpenGL pobrałem z przykładów zamieszczonych na wyżej przedstawionych stronach. Zawierały on procedury takie jak:

setupPixelFormat – ustalającą podstawowe parametry wyświetlania,
GLInit – inicjującą wyświetlanie okna w OpenGL,
Draw – będącą podstawą do wywoływania obiektów w oknie;

W dodatkowym pliku (obiekty.pas) umieszczone są najważniejsze procedury i funkcje, wywoływane przez procedurę Draw programu głównego, a zawierające algorytmy odpowiedzialne za tworzenie terenu. To właśnie tym algorytmom poświecony będzie niniejszy podpunkt.

Jedną z podstawowych spraw jest przekazanie do modułu wizualizacji, danych o terenie, czyli prościej mówiąc wczytanie pliku z opisem mapy cyfrowej. Zasadniczo w tym miejscu można napomnieć, że program do wizualizacji ma możliwość dwojakiego uruchamiania się i pobierania danych o pliku, co jest dość ważne dla całości projektu. Wizualizacja może być uruchomiona jako niezależna aplikacja, a mapa wczytana „ręcznie", spod poziomu menu programu, lub przez wywołanie aplikacji, z parametrem przekazującym dane o pliku. Stanowi to element łączący trzy aplikacje. Moduły Generator i Analiza są w stanie, podczas pracy wywołać Wizualizację i wczytać do niej przerabiany plik. Daje to możliwość bezpośredniego dostępu do formy przestrzennej.

Tak więc po uruchomieniu wczytywania pliku z mapą cyfrową, jedną z pierwszych procedur jest przywołanie zmiennych opisujących mapę i wypełnienie nimi tablicy. Procedura ta jest identyczna jak w pozostałych aplikacjach, ale w tym przypadku znajduje się w oddzielnym pliku, dołączanym podczas kompilacji do głównego programu.

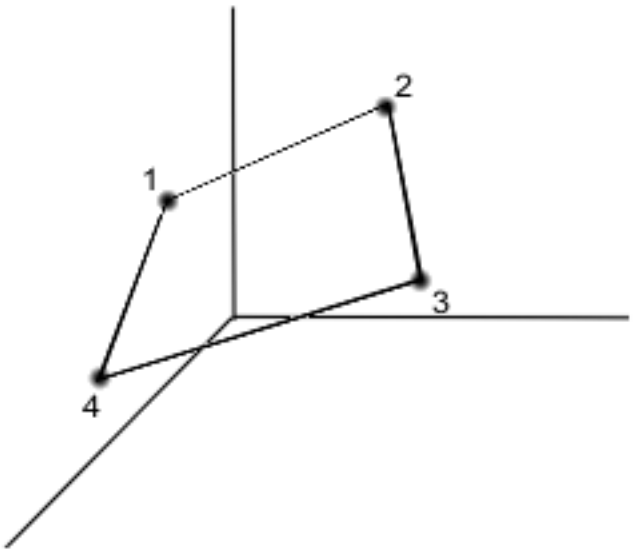
Tworzenie terenu

Najważniejszą procedurą, choć zawierającą na pierwszy rzut oka niewiele kodu, jest „teren". Służy ona do obliczania i tworzenia siatki terenu złożonej z wielu specjalnie obliczonych kwadratów, umieszczonych w przestrzeni.

Pomysł tworzenia jest bardzo prosty. Ponieważ tablica zawiera szereg punktów, odpowiadających wysokościom poszczególnych pikseli, można pogrupować je po cztery, przylegające do siebie i umieścić je w przestrzeni. Samo wygenerowanie takiego kwadratu jest dość proste. OpenGL umożliwia tworzenie obiektów przestrzennych na podstawie podanych punktów. W przypadku projektu wykorzystane zostały dwa składniki:

- GL_QUADS – dla powierzchni wypełnianych,
- GL_LINE_STRIP – dla samej siatki;

Wykorzystanie tych dwóch funkcji OpenGL jednocześnie zdeterminowało sposób pobierania i analizowania punktów, jak również pozwoliło tworzyć efektowną scenę.



Rys. 13

Zasada umiejscawiania punktów w przestrzeni jest prosta. Dwie pętle „przemieszczają" się po dwuwymiarowej tablicy, zasada opisana była wcześniej już kilkakrotnie, pobierają informacje o punkcie (odpowiada on informacji o położeniu w tablicy) i wczytując wysokości. Właściwie mamy już trzy współrzędne X, Y, Z, ale powstały tu dwa problemy.

Pierwszy to umiejscowienie terenu na środku okna, w którym ma być wyświetlany. Dla OpenGL środek okna, jest również środkiem układu współrzędnych, ale dane w tablicy były pobrane w płaszczyzny, gdzie początkiem układu współrzędnych jest lewy, górny róg. Tak więc należy ustalić środek tablicy i odpowiednio przesunąć wszystkie piksele. W tym celu powstały dwie specjalne zmienne, określające zero bezwzględne dla przedstawianego terenu. Ich wyliczenie to wyśrodkowanie, przez prostą arytmetykę, tablicy. Zapewnia to następujący kod:

```
zerox:=-(w div 2);
zeroy:=-(h div 2);
```

Przy czym „w" i „h" są zmiennymi przechowującymi informacje o szerokości i długości tablicy.

Określenie środka terenu i zastosowanie prostego przesunięcia punktów pozwoli wyśrodkować w oknie programu, tworzoną siatkę.

Kolejnym problemem było odpowiednie przeskalowanie wysokości. Należy pamiętać, że przy tych obliczeniach powstała pewna niezgodność. Piksele reprezentujące poszczególne punkty mapy były pozostawione same sobie. Natomiast wysokość była wartością wyrażoną w jednostkach odległości. Powstawał tu konflikt zastosowanych jednostek. Gdyby zapisać wysokość w wartości pobranej bezpośrednio z tablicy, powstałby efekt, określany przez wielu, w sposób wyjątkowo swobodny, jako „rozjeżdżanie się" grafiki. Poszczególne piksele mapy w poziomie byłyby obok siebie, ale ich przesunięcie w górę lub w dół nabierałoby niebotycznych wartości. Tak więc by zachować proporcjonalność każda wysokość została podzielona przez skalę – zmienną przechowywaną w pliku mapy cyfrowej i dopiero wstawiona jako jedna ze współrzędnych.
Cały kod ustawiający piksel mapy wygląda następująco:

```
obl:=tabcyf[(x*s)+s,(y*s)]/skala;
    pom:=obl; wysrz:=pom;
    glVertex3d(zerox+(x*s)+s, wysrz, zeroy+(y*s));
```

Tak wiec by podsumować. Dwie pętle przeszukują dwuwymiarową tablicę i ustawiają w przestrzeni, korzystając ze specjalnych kodów, piksele po cztery. Tak stworzony czworobok jest jednym z wielu „okienek" tworzących siatkę, lub wypełniony model terenu.
W tym miejscu należy dodać jeszcze jedno wyjaśnienie. Przewijająca się przez powyższy kod, zmienna „s", jest związana z ustalaniem precyzji tworzenia terenu. Aplikacja umożliwia określanie w jakiej dokładności ma być tworzony model przestrzenny terenu. Im mniej „okienek" tym model jest mniej dokładny, ale trzeba, to zaznaczyć, znacznie szybciej jest przeliczany. Ma to swoje uzasadnienie i korzyści. Jeśli komputer, na którym przedstawiany jest trójwymiarowy teren, nie posiada szybkiej karty graficznej, zmniejszenie dokładności terenu przyspieszy działanie wizualizacji.
Za ten efekt odpowiedzialna jest właśnie zmienna „s". Jest to liczna całkowita, która określa co ile ma zostać pobrany kolejny punkt tablicy, do stworzenia rogu czworoboku. Im większa wartość tej zmiennej tym większy jest skok po tablicy, większe „okienko" siatki, ale mniejsza dokładność prezentowanego terenu.

Na tym zasadniczo kończy się proces tworzenia siatki, lub wypełnionej bryły reprezentującej teren. Należy jednak zaznaczyć, że dodatkowy kod, zapewnia możliwość przełączania się pomiędzy funkcjami OpenGL, opisanymi na początku, co umożliwia generowanie raz terenu w postaci siatki, raz przy jej wypełnianiu. Jednak te zależności są na tyle proste, iż chyba nie ma sensu przedstawiać tu kodu i zasady ich działania.

Kolorowanie terenu

Kolejnym istotnym krokiem jest odpowiednie pokolorowanie terenu, a dokładniej mówiąc poszczególnych czworoboków. Pełny kod ustawiający piksele w przestrzeni wygląda następująco:

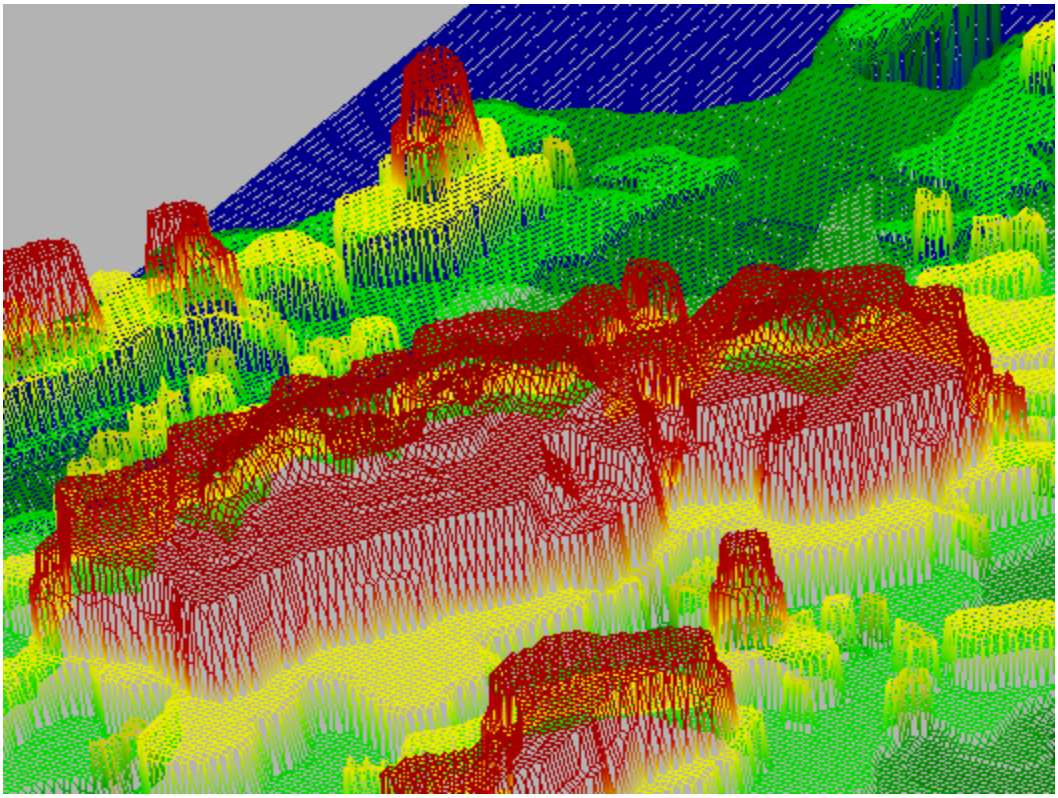
```
Ustaw_kolor(tabcyf[(x*s)+s,(y*s)]);
    obl:=tabcyf[(x*s)+s,(y*s)]/skala;
    pom:=obl; wysrz:=pom;
    glVertex3d(zerox+(x*s)+s, wysrz, zeroy+(y*s));
```

Pierwsza linia, nie prezentowana wcześniej zawiera wywołanie procedury „Ustaw_kolor". Jest ona odpowiedzialna za określenie jakiego koloru ma być piksel danego roku czworoboku. Ma to olbrzymie znaczenie, dla końcowego efektu. Załóżmy, że dane, pojedyncze „okienko" leży w takim położeniu, że jeden z jego rogów znajduje się na wysokiej, stromej górze, a pozostałe należą do dolin. Taki przypadek szczególnie może mieć miejsce, gdy zastosowana jest mała dokładność tworzonego terenu, a czworoboki, są bardzo rozległe. Wówczas, odpowiednie kolorowanie poszczególnych rogów pozwala OpenGL stworzyć przejścia kolorystyczne wewnątrz pola. Zapewnia to ciekawe efekty i płynność wypełniania w generowanym terenie.

Przejście do podglądu procedury odpowiedzialnej za „kolorowanie mapy", powodować może niechęć. Procedura jest rozbudowana i niezbyt czytelna. Jednak została zbudowana tak, by zapewnić odpowiednie działanie zarówno podczas pełnego kolorowania siatki, jak również podczas wybierania funkcji wyróżniania tylko wybranych obszarów.
Tak wiec procedura podzielona jest na dwie zasadnicze części: kolorowania wszystkich pikseli, lub kolorowania tylko wybranych obszarów.

Oczywiście może ona wcale nie reagować jeśli użytkownik nie zaznaczył opcji kolorowania. Wówczas, choć jest ona wywoływana nie dokonuje żadnych obliczeń.

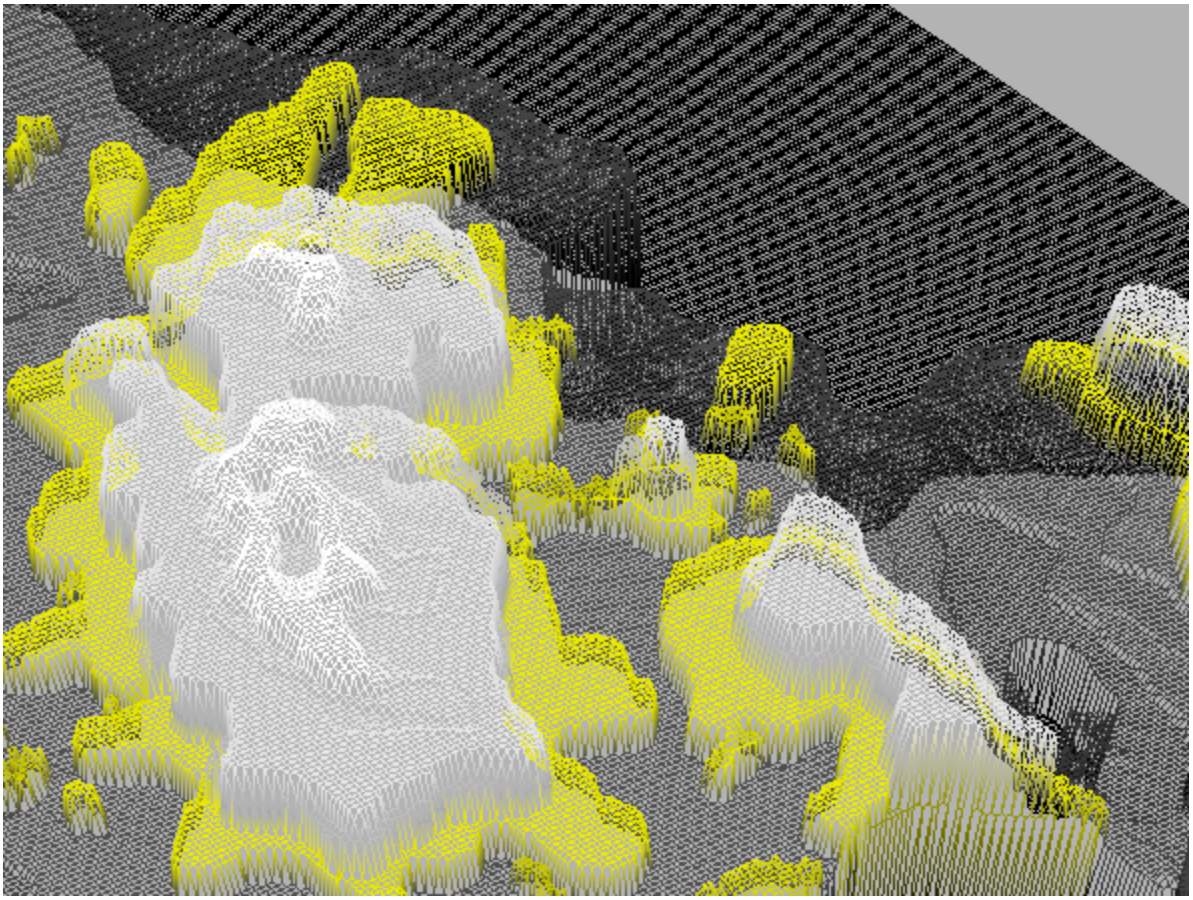
Pierwszą możliwością jest kolorowanie wszystkich pikseli. Zmienną wejściową dla procedury jest „wysp" typu Integer – przekazująca wysokość danego punktu. Wysokość ta jest porównywana z przetrzymywanymi przez zmienne globalne, zakresami dla danych terenów, a następnie po rozpoznaniu, jako: depresja, nizina, wyżyna lub góra, wyliczany jest odpowiedni kolor. Sens obliczania koloru wyjaśniany był podczas omawiania implementacji generatora map cyfrowych w punkcie 6.3. Jedyną różnicą jest sposób zapisu wartości poszczególnych składowych koloru. W poprzednim module wykorzystywane są wartości całkowite w zakresach od 0 do 255. Tu natomiast wartości te, są liczbami rzeczywistymi (analogicznie przedstawianymi). Dla przykładu: 225 odpowiada wartość 2.25. Wynika to z formatu opisu kolorów w OpenGL.



Rys. 14

Drugą możliwością oprogramowaną w procedurze, jest kolorowanie tylko wybranych obszarów. Dla uproszczenia moduł umożliwia tylko wyróżnienie depresji, nizin, wyżyn lub gór. Zasadniczo część ta nie różni się od poprzedniej. Jedynie, po stwierdzeniu, iż

badana wysokość piksela znajduje się w zaznaczanym zakresie jest obliczany kolor odpowiadający danym poziomom, w przeciwnym wypadku jest obliczana skala szarości dla danego piksela. Całość dokładniej opisana została przy okazji tworzenia moduły analizy i znajduje się w części 6.5. Końcowy efekt działania jest jednak naprawdę ciekawy:



Rys. 15

W ten sposób kolorowane są poszczególne piksele tworzące czworoboki, składające się na modelowany teren cyfrowy. W zależności od sposobu kolorowania, wypełnienia i poziomu precyzji, uzyskujemy zaskakująco różne efekty dla tego samego terenu. Całość spełnia znakomicie funkcję zabawy i obrazowania cyfrowej mapy.

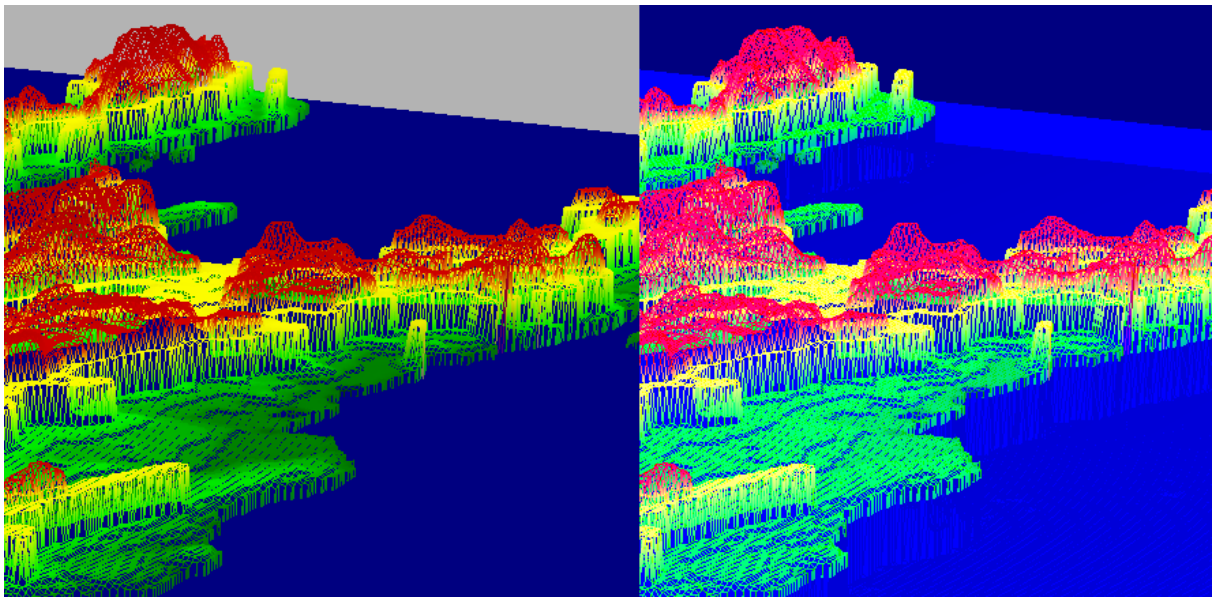
Dodatkowe procedury i funkcje

Ponieważ zasadniczą role odgrywały algorytmy przedstawione na wcześniejszych kilku stronach, można by przyjąć, że projekt modułu, jest już opisany. Należy jednak wspomnieć o kilku dodatkowych możliwościach, które wzbogacają całość.

Jedną z małych dodatkowych procedur znajdujących się w pliku obiekty.pas jest „poziom_zerowy". Prosta procedura, która tworzy prostokąt, o wymiarach odpowiadających, całemu terenowi, a umiejscowionym na zerowej wysokości. Pomaga on oznaczyć poziom morza. W większości przypadków jest to wyjątkowo wygodne, gdyż znacząca część ukształtowania terenu znajduje się pod wodami. Dodatkowym, atutem jest ciekawy efekt przezroczystości, który uaktywnia się podczas włączania oświetlenia.

Mówiąc już o oświetleniu nie sposób pominąć procedury odpowiedzialnej za generowanie tego efektu. Choć całość została pobrana razem ze szkieletem sceny w OpenGL, to drobna modyfikacja umożliwia zmiany wyglądu otoczenia. Ponieważ właściwości materiału, z którego tworzony jest teren, uaktywniają się przy dodatkowym oświetleniu, różnice mogą niekiedy przynosić przyjemne wizualnie efekty.

Przykład zastosowania sceny z włączonym i wyłączonym dodatkowym oświetleniem:



Rys. 16

Jak widać zastosowanie oświetlenia zmienia w sposób znaczący wygląd i dynamikę sceny.
Na zakończenie opisu algorytmów, należy wspomnieć o kilku prostych operacjach, mających za cel tworzenie interaktywności prezentowanej sceny.
Wykorzystując macierze w OpenGL i transformacje na nich, możliwe stało się przybliżanie, oddalania oraz obracanie sceną. Nie ma chyba sensu się nad tym rozpisywać, gdyż są to dobrze znane czynności, których analizę i wykonanie przejmuje na siebie silnik graficzny.
Wykorzystane odpowiednio transformacje, pozwoliły stworzyć animację obracania sceny, względem płaszczyzny.

Podsumowując całość opisu moduły wizualizacji przestrzennej, można powiedzieć, iż choć był projektowany jako dodatek, stał się niezależną, w pełni funkcjonalną aplikacją, która bardzo obrazowo przedstawia wyniki działania generatora map cyfrowych. W pewnym sensie, ma ona bardzo wielkie znaczenie. Przedstawienie w postaci trójwymiarowej, terenu, znacznie bardziej działa na wyobraźnię oglądającego. Prezentacja przestrzenna modelu daje możliwość wykreowania wizerunku rzeczywistej rzeźby terenu i choć jest tylko grafiką komputerową znacznie ułatwia dotarcie do odbiorcy, zapoznającego się w cyfrowym modelem.

6.5. Implementacja modułu informacyjnego

Ostatnim modułem tworzonym w ramach projektu jest moduł informacyjny. Jest on stworzony jako niezależny program i został nazwany „Analiza". Całość pod względem wizualizacji nie odbiega formalnie od pozostałych modułów. Motyw oka znajduję się również i w tym miejscu.

Poszczególne algorytmy zostały rozmieszczone w rozwijanym z paska górnego menu. Dodatkowe okna mają ułatwiać tworzenie przejrzystego miejsca pracy i analizy.

Choć wstępne założenia dotyczące tego projektu były wcześniej zaplanowane, to podczas tworzenia powstały nowe myśli i rozwiązania.

Jednak główną siłą modułów projektu są zawarte w nich algorytmy. To one determinują właściwe działanie programu i one odpowiadają za poprawność dokonanych analiz.

Opis zastosowanych do analizy map cyfrowych, algorytmów zawarty jest na kolejnych stronach.

6.5.1. Algorytmy obliczeń danych statystycznych.

Jednym z podstawowych założeń przy tworzeniu modułu analizy map cyfrowych utworzonych w ramach projektu, było stworzenie aplikacji umożliwiającej badanie i tworzenie statystyk. W tym celu powstała grupa prostych algorytmów, tworzących raporty i obrazujących zjawiska fizyczne występujące na danym obszarze.

Statystyki mapy

Pierwszą myślą było ogólne podsumowanie zawartości wczytanej mapy. Algorytm miał za zadanie przeanalizować zawartość tablicy reprezentującej teren i stworzyć liczbowe podsumowania jej zawartości. W ten sposób mogą powstać informacje statystyczne.

Zasadniczo rzecz ujmując algorytm sprowadza się do całego szeregu nieskomplikowanych obliczeń arytmetycznych. Pierwsze informacje, które najprościej uzyskać to długość i szerokość mapy wyrażonej w jednostkach odległości, mogących obrazować zakres przedstawianego terenu. Te informacje uzyskać można już z danych zawartych w samym, zaproponowanym formacie. Wielkości tablicy odpowiadają długości i szerokości mapy wyrażonej w pikselach. Do tego dochodzi informacja o skali mapy. Pomnożenie tych dwóch składników i ich przeliczenie na kilometry to już tylko formalność:

```
sz:=(hmapy*skala)/1000;
dl:=(wmapy*skala)/1000;
pow:=sz*dl;
```

Ostatni wiersz przedstawionego u powyżej kodu źródłowego oblicza dodatkowo powierzchnię terenu, wyrażoną w kilometrach kwadratowych.

Kolejne informacje dotyczyły powierzchni jaką zajmują poszczególne sfery fizyczne. Mam tu na myśli góry, wyżyny, doliny i depresje. Rozwiązaniem było badanie każdego piksela mapy i sprawdzanie, czy jego wysokość kwalifikuje się do danego zakresu. Zakresy, dla przypomnienia zawarte są już w formacie danych dla map cyfrowych. Tak więc ogólna pętla dla wszystkich obliczeń związanych ze zróżnicowaniem terenu, sprawdza piksel po pikselu dane wysokości i kwalifikuje je do danej grupy przez dodanie do odpowiednich zmiennych. Działanie to wykonuje poniższy algorytm:

```
for y:=0 to h-1 do
  Begin
   for x:=0 to w-1 do
    Begin
     ile:=ile+1;
     wys:=wysokosc(x+1,y+1);
     if wys<0 then dep:=dep+1;
     if (wys>minzie) and (wys<maxzie) then niz:=niz+1;
     if (wys>minzol) and (wys<maxzol) then wyz:=wyz+1;
     if (wys>mincze) and (wys<maxcze) then gor:=gor+1;
    end;
  end;
```

Kolejna pętla zawarta w tym algorytmie dokonuje sprawdzenie maksymalnych i minimalnych wysokości, jednocześnie zbierając informacje o średnich wysokościach dla ogółu mapy i dla samych terenów lądowych. Pętla ta pozwala na określenie najwyższego szczytu, oraz największej depresji.

Suma wszystkich wysokości, oraz wysokości na lądzie, pozwala określić statystyczne informacje dla danej mapy. Działania te wykonują przedstawione poniżej pętle:

```
for y:=0 to h-1 do  Begin
   for x:=0 to w-1 do    Begin
     wys:=wysokosc(x+1,y+1);
     if wys<glebokosc then glebokosc:=wys;
     if wys>szczyt then szczyt:=wys;
     if wys>0 then
      Begin
       srednialad:=srednialad+wys;
      end;
     srednia:=srednia+wys;
    end;
  end;
```

Całość dodatkowo wzbogaca kod pozwalający na właściwe obrazowanie obliczeń i zapisywanie ich w specjalnie do tego celu stworzonej formatce.

Wynikiem działania tego algorytmu jest przedstawione okno:

Rys. 17

Zaznaczenia obszarów na mapie

Kolejną grupą analogicznych w działaniu i konstrukcji, a oddzielnie uruchamianych algorytmów, są obliczenia związane z wizualizacją danych stref na mapie. Wykorzystując stworzone wcześniej algorytmy obliczania i tworzenia koloru na podstawie wysokości, opracowane zostały nowe wyróżniające tylko określone wysokości. Zasadniczy problem polegał jedynie na zgraniu ze sobą dwóch algorytmów: kolorowania mapy zgodnie z zasadami znanymi z kartografii i stworzenia skali szarości z zachowaniem domyślnych obrysów.

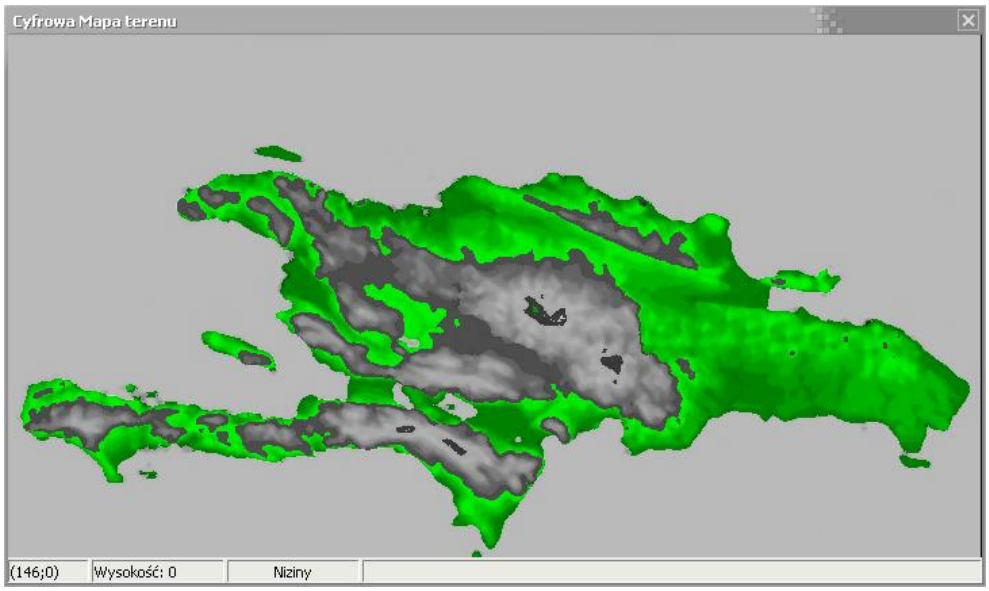
Zasada tego działania jest jednak prosta. Algorytm sprawdza kolejne miejsca w tablicy reprezentującą mapę. Przy każdym miejscy sprawdza, czy zapisana wysokość nie należy do poszukiwanego zakresu. Jeśli tak, koloruje odpowiadający tablicy piksel, zgodnie z wcześniej stworzonymi zasadami, jeśli nie, zamienia kolor piksela na skalę szarości.

By dokładniej zrozumieć zasadę kolorowania i zamieniana koloru na skalę szarości proszę cofnąć się do podrozdziału 6.3. Implementacja generatora map cyfrowych. Opis ten znajduje się w części poświęconej algorytmom analizy mapy fizycznej.

Natomiast poniższy kod prezentuje omawiany proces.

```
For y:=0 to h-1 do  Begin
  for x:=0 to w-1 do   Begin
    wys:= alcze ć(x+1,y+1);
    R:=0;
    G:=0;
    B:=0;
    if (wys>=minzie) and (wys<maxzie) then
     Begin
      pomR:=(255/(maxzie-minzie))*wys;
      pomV:=pomR;
      G:=pomV;
     end
    else
     Begin
      pomR:=(255/( alcze))*wys;
      pomV:=pomR;
      G:=pomV;
      R:=pomV;
      B:=pomV;
     end;
    kolor:=RGB(R,G,B);
    MapaCyfr.Image1.Canvas.Pixels[x,y]:=kolor;
   end;
 end;
```

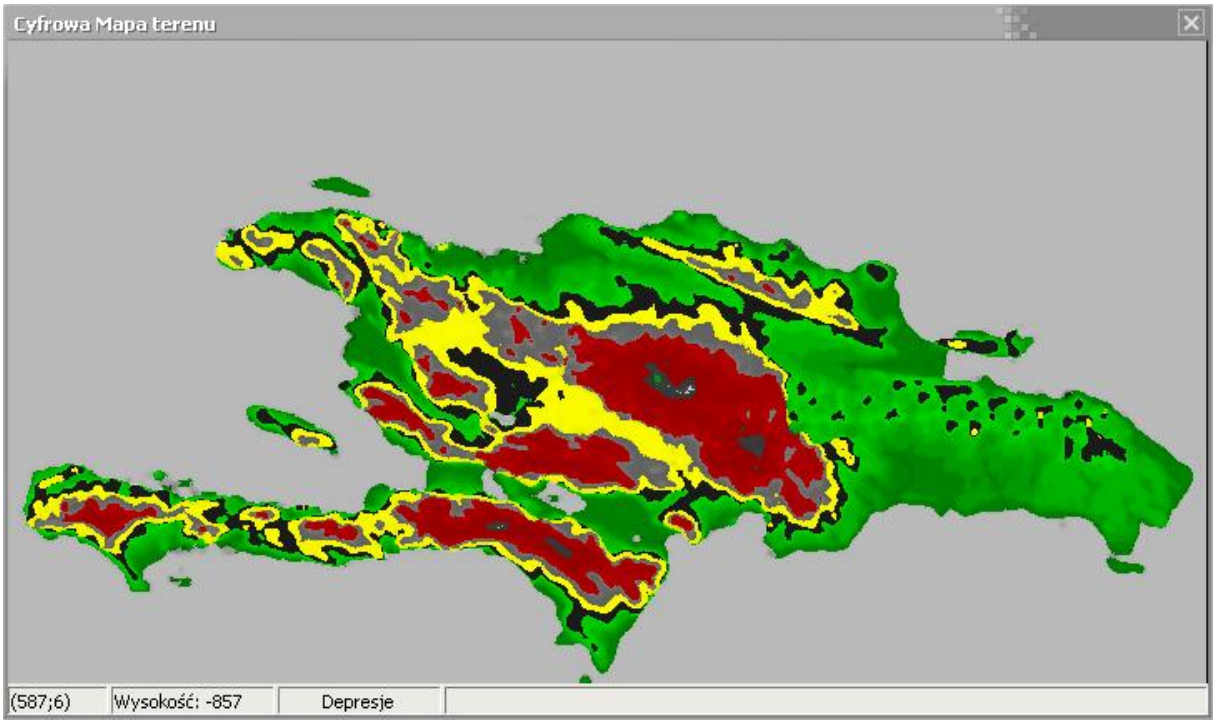
Przedstawiony kod dotyczy terenów nizinnych. Mówią o tym porównania wczytanych wysokości do zmiennych maxzie i minzie. Analogiczne działanie wykonywane jest dla pozostałych obszarów. Końcowym efektem jest zamiana kolorystyki mapy. Przykład takiej operacji ilustruje grafika.



Rys. 18

Po zakończeniu tworzenia tej grupy procedur doszedłem do wniosku, iż brakuje tu pewnej swobody działania. Program determinuje jakie obszary mogą być wyróżniane. Stwierdziłem więc, że nieco zmodyfikowany, wyżej opisywany algorytm może pozwolić na precyzyjniejsze określanie zakresu zaznaczenia.

Modyfikacja dotyczyła zmiennych, które program porównuje do pobranych z tablicy wysokości. W poprzednich algorytmach zmienne te były określone na stałe. Po modyfikacji zależały one od wartości podanych przez użytkownika. Choć zmiana była niewielka dała wymierne rezultaty, które obrazuje poniższa grafika.



Rys. 19

Zaznaczanie punktów krytycznych mapy

Ostatnimi prostymi algorytmami, którym poświecę zaledwie kilka słów, są obliczenia obrazujące maksymalne i minimalne wysokości. Algorytmy są wyjątkowo proste. Sprawdzają one poszczególne miejsca w tablicy reprezentującą mapę i wyszykują najbardziej istotne wysokości. Sam proces jest znanym algorytmem wyszukiwania pewnych wartości w tablicy. Kolejno sprawdza się wysokości i zapisuje szukane, zamieniając je dopiero wówczas, gdy znajdzie się wartość bardziej odpowiadająca kryteriom wyszukiwania. Po zakończeniu przeglądania tablicy, przechowywane wartości, są właściwymi.

6.5.2. Algorytmy obliczeń geograficznych.

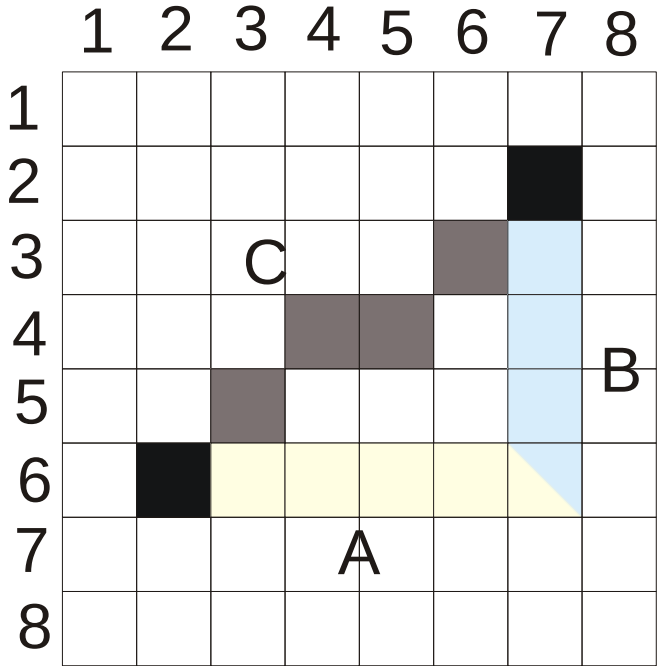
Głównym algorytmem obliczeń geograficznych jest moduł wyliczający odległości między podanymi punktami, w linii prostej.
Ponieważ mapa jest to zbiór punktów na płaszczyźnie, których właściwością jest przetrzymywana w tablicy wysokość, zaznaczone punkty obliczane są na podstawie dwóch współrzędnych X i Y. Tak więc, by dokonać niezbędnych działań matematycznych potrzebne będą dwa punkty: początkowy i końcowy. W tym momencie algorytm możemy podzielić na trzy działania:
- obliczenie drogi w pikselach od punktu początkowego do punktu końcowego,
- obliczenie zrzutowanej odległości (bez nierówności terenu),
- obliczenie odległości rzeczywistej;

Najprościej będzie obliczać odległość rzutowaną. Jest to rzut prostopadły na analizowany teren. Nie uwzględnia on wzniesień i depresji, ale jest doskonałym uproszczeniem znanym z konwencjonalnych map fizycznych. By wyjaśnić jego istotę, najlepiej napisać, że jest on ilorazem odległości między punktami na mapie fizycznej i jej skali. Podany mechanizm zastosuje również ja. Po obliczeniu ilości punktów dzielących początek i koniec, pomnożę te liczbę przez skalę mapy. Ponieważ skala mapy mówi mi ile metrów przypada na jeden piksel, zachowam tu zgodność jednostek. Da mi to wartość odległości rzutowanej wyrażonej w metrach. Całe obliczenie sprowadzi się do algorytmu:

```
rzut := ilość * skala;
```

Wynik będzie liczbą rzeczywistą, gdyż taka jest skala, stosowana w działaniu matematycznym. W taki sposób otrzymywać będę wartość odległości w rzucie prostopadłym na mapę.
By jednak tego dokonać będę musiał obliczyć, przy pomocy kolejnego algorytmu odległość, wyrażoną w ilości pikseli, pomiędzy dwoma punktami. Jest to najważniejszy i decydujący algorytm. Wykorzystany zostanie w obydwu obliczeniach odległości, to znaczy w opisanym powyżej, dla wartości rzutowanych, jak również dla wartości rzeczywistych.

By lepiej zrozumieć sens obliczania odległości między zaznaczonymi dwoma punktami, najlepiej posłużyć się będzie rysunkiem, przedstawiającym tablice danych punktów:



Rys. 20

Użytkownik zaznacza dwa dowolnie umiejscowione na mapie punkty. Nie ma dla niego znaczenia, czy pierwszy będzie wyżej, czy niżej od drugiego i jak względem siebie będą one położone. To bardzo ważna informacje, gdyż wymusza stworzenie procedur odpornych na błędy wynikające z nieprawidłowego rozpoznania punktów.

Mając dwa dowolnie położone punkty, na płaszczyźnie i chcąc obliczyć ich odległość, wyrażoną w pikselach, wykorzystam równanie znane wszystkim uczniom, męczącym się z matematyką. Twierdzenie Pitagorasa, będzie tu idealnym rozwiązaniem. Jeśli założyć, że ramiona utworzone przez pionowe i poziome przedłużenie naszych punktów będą przyprostokątnymi trójkąta prostokątnego, to szukana linia będzie przeciwprostokątną. Mając długość owej linii będziemy mieli niezbędną nam dla całego działania informację. Tak więc wykorzystując równanie Pitagorasa zakładam, że:

- a będzie przyprostokątną poziomą i obliczaną z wartości x1 i x2 (współrzędne punktów początkowego i końcowego).
- b będzie przyprostokątną pionową i obliczaną z wartość y1 i y2 współrzędnych punktów.

- c szukana wartość przeciwprostokątnej, w tym przypadku będącą odległością między podanymi punktami.

Jak napisałem wcześniej problemem jest brak spójności położenia zaznaczonych punktów odnośnie siebie. By długości boków trójkąta, a wiec a i b nie były wartościami ujemnymi, muszę sprawdzić jak punkty leżą względem siebie, tak by w działaniu: $bok = wsp1 - wsp2$ nie odejmować wartości większej od mniejszej. Prosty kod warunkowy zapewnia mi zgodność danych:

```
if x2>x1 then a:=x2-x1 else a:=x1-x2;
if y2>y1 then b:=y2-y1 else b:=y1-y2;
```

Następnie z twierdzenia Pitagorasa obliczam wartość c – czyli długość przeciwprostokątnej:

```
cp:=sqrt((a*a)+(b*b));
pom:=cp;
c:=pom;
```

Dodatkowe zmienne, rozbudowujące kod zastosowane zostały by uwzględnić dokładność obliczeń na liczbach rzeczywistych, ale ich końcowe wyniki zaprezentować w postaci całkowitej.

Gdy już obliczyłem długość przeciwprostokątnej wykorzystam ją by wyliczyć tzw. skok, czyli wartość potrzebną by od jednego punktu przenieść się do kolejnego, zbliżając się tym samym do punktu końcowego:

```
skxp:=a/c;
skyp:=b/c;
if (x2-x1)<0 then skxp:=-skxp;
if (y2-y1)<0 then skyp:=-skyp;
```

Ostatnie dwie linie kodu zapewniają by wartości skoku miały odpowiednie znaki. Chroni to, algorytm, przed podążaniem, co prawda o odpowiednią ilość pikseli, ale w złą stronę.

W tym momencie mogę zrealizować obliczenie rzutu prostopadłego, opisywanego na początku podrozdziału. Mam niezbędne dane. Rzut równy będzie ilorazowi skali,

przechowywanej w pamięci programu i odległości w pikselach (czyli ilości pikseli) pomiędzy dwoma punktami, a jest to obliczone właśnie c – długość przeciwprostokątnej.
Jednocześnie wartość c pozwoli nam wykonać określoną ilość razy pętlę zliczającą rzeczywiste odległości pomiędzy danymi punktami. Szkielet pętli będzie wyglądał następująco:

```
for i:=1 to c do
Begin
  cp:=sqrt((ap*ap)+(b*b));
  odleg:=odleg+cp;
end;
```

Wewnętrzne obliczenie określa różnicę odległości, już z uwzględnieniem wysokości, między kolejnymi pikselami i ich sumowanie dla uzyskania całkowitej odległości rzeczywistej między dwoma punktami. By uzyskać dane na temat różnic wysokości posłużyłem się ponownie równaniem Pitagorasa, traktując wysokości terenu przypisane danym pikselom jako wierzchołki trójkąta.
Przedostatnia linijka w przedstawionej powyżej pętli, daje nam ostatecznie, po wykonaniu wszystkich obrotów pętli, wartość odległości między dwoma punktami zaznaczonymi na mapie.

Końcowym etapem tworzenia algorytmu obliczeń geograficznych jest jego wielokrotne wykorzystanie. Po stworzeniu działającej procedury, obliczającej sprawnie odległości (rzeczywistą i rzutowaną) miedzy dwoma punktami, dodałem tablicę przechowującą informacje o wielu punktach. Program wyposażony został w tablicę na maksymalnie 50 punktów. Daje to spore możliwości tworzenia złożonych tras, a jednocześnie budowa algorytmu pozwala na dowolne rozszerzanie tablicy, a tym samym zwiększanie maksymalnie dostępnej ilości punktów drogi.
Tak wiec, ostatecznie, algorytm sprawdza ile ma punktów do przebadania, traktuje kolejne dwa jako podstawę do obliczeń. Stosuje wyżej opisane procedury, a ich wyniki, zarówno dla wartości rzeczywistych, jak i rzutowanych sumuje, dając na końcu, odległości tras opisanych przez użytkownika.
Dodatkowo wprowadziłem kilka linijek kodu generujące linie na mapie, co pozwala wizualizować efekt obliczeń.

Rezultat obliczeń okazał się zaskakująco dobry. Zaznaczone dwa punkty dają, w wielu przypadkach (gdy postawimy punkty między górami, a dolinami), różne odległości rzutowane i rzeczywiste. Jednocześnie, dla terenów o jednakowych wartościach wzniesień i depresji, różnice te są niewielkie, lub nie ma ich wcale. To potwierdza słuszność założeń i poprawność działania algorytmu.

6.5.3. Algorytmy kompresji formatu danych

Podczas realizacji projektu dostrzegłem iż wynikowe pliki map cyfrowych, zawierające tablice wysokości terenu, są znaczących rozmiarów. Gdy mapa fizyczna, zapisana w pliku BMP ma wielkość około 1 Mb, to wygenerowany plik mapy cyfrowej potrafi być, o kilka do kilkuset kilobajtów większy, w zależności od specyfikacji terenu. Choć dla celów dydaktycznych nie stanowi to znaczącego problemu, a dzisiejsze nośniki danych zawierają wystarczającą ilość miejsca na przetrzymanie setek tysięcy podobnych plików, to format okazuje się niezbyt wydajny. Powstała więc myśl by zastosować choćby najprostszy algorytm kompresji danych.

Zasada realizacji jest prosta. Część pliku która zawiera opis specyfikacji mapy, a więc zakresy wysokości, długości i szerokości, oraz skalę mapy, zostanie nienaruszony. Kompresja dotyczyć będzie natomiast danych z tablicy.

Pomysł kompresji danych wynika z obserwacji zróżnicowania terenu przedstawianego na mapie. Każdy piksel mapy opisany jest jej wyliczoną wysokością. Na podobnym terenie pikseli o jednakowych wysokościach jest bardzo wiele, ale przy zastosowanym formacie, każdy ma własny opis. Wykorzystując metody prostych kompresji plików graficznych, jako jedna grupa zapisywane są piksele o podobnych lub identycznych właściwościach. Tu pomysł będzie podobny. Wpisy z tablicy będą grupowane ze względu na jednakowe wartości.

Informacje o wysokości zapisywane są do pliku kolejno. To znaczy, że odczytana z tablicy wartość zajmuje cały, nowy wiersz w pliku. Po wyedytowaniu w dowolnym programie do obsługi plików tekstowych, dane o wysokościach wyglądać będą następująco:

-2384
-2384
-2370
1020
1150
1150

Pomysł kompresji opiera się na założeniu, by identyczne dane zapisywać jako połączenie wartości i ich ilości. Tak więc algorytm kompresji będzie skanował kolejne linie pliku mapy cyfrowej i zliczał identyczne wartości. Podany wyżej przykład poddany kompresji zmniejszy się o 2 linie i będzie wyglądał następująco:

-2384^2
-2370^1
1020^1
1150^2

Dekompresja będzie polegała na procesie odwrotnym. Odczytana linia skompresowanego pliku będzie podlegała analizie. Po sprawdzeniu ile razy występuje dana wartość, wykonana zostanie pętla, „obracająca” się o tą ilość i wstawiająca wysokość do tablicy.
Proces nie jest złożony i znacząco nie obciąży procesora.

Wady i zalety

Przedstawiony algorytm kompresji ma dwie zasadnicze zalety. Mianowicie, pierwszą z nich, jest znaczące zmniejszenie wielkości plików skompresowanych, w stosunku do podstawowych plików formatu. Przy sprzyjających kompresji wartościach wysokości, pliki mogą zmniejszyć się nawet o połowę lub więcej.
Kolejną zaletą jest bezstratność danych podczas kompresji. Algorytm zapisuje identyczne dane, a nie również podobne, co powodowałoby straty. Po rozpakowaniu informacje są identyczne jak sprzed wykonania kompresji.

Wadą podanego algorytmu jest brak odporności na wyjątkowo zróżnicowany teren. W przypadku gdy informacje o wysokości byłyby różne dla każdego piksela, wynikowy plik byłby nawet większy, gdyż zawierałby w każdej linii dodatkowe znaki. Jednak, taka opcja jest mało prawdopodobna, a jednocześnie możliwa do zoptymalizowania, przy kolejnych wersjach algorytmu.

Kompresja danych została dodana opcjonalnie i jest dodatkową możliwością, przydatną podczas archiwizowania plików map cyfrowych.

7. Testowanie aplikacji

Praca jest właściwie gotowa. Projekt teoretyczny uwieńczony implementacją i ewolucyjnym rozwinięciem, został właściwie zakończony. Jednak wyjątkowo istotne jest sprawdzenie aplikacji w jej działaniu pod kątem błędów i wydajności.

Należy tu zaznaczyć, iż większość testów przeprowadzana była w trakcie tworzenia oprogramowania. Nie sposób było, na przykład, prawidłowo projektować aplikacji do analizy map cyfrowych, jeśli moduł wizualizacji nie działałby prawidłowo, a jego wywołanie byłoby niemożliwe. Tak więc znacząca cześć testów, ich analiza i ewentualne rozwiązanie błędów, opisanych w niniejszym rozdziale była przeprowadzana już w czasie prac projektowych.

Mają jednak one na celu, zapoznanie z napotkanymi problemami, często ujawniającymi się dopiero podczas testowania. Rozdział ten może również posłużyć jako pewna wskazówka, przy dalszym rozwoju aplikacji, by pokazać niektóre newralgiczne punkty projektu.

7.1. Wykrywanie i usuwanie błędów

Zdecydowana większość napotkanych błędów powstała na skutek zbyt ogólnego zaplanowania pewnych posunięć. Wynikały one najczęściej, również z nieprzewidzianych efektów działań rozwijających się algorytmów.

Wykrywanie błędów podczas rozpoznawania mapy fizycznej

Operacje związane z obsługą odczytu i zapisu formatu danych, były doskonale przewidziane więc nie przysparzały problemu. Inaczej sprawa się miała z samą analizą mapy. Opisywany w rozdziałach 5 i 6, sposób określania strefy na podstawie wysokości, zawiera pewną informację dotyczącą wyżyn. Na mapach fizycznych prezentowane są one jako obszary zabarwione na żółto. Jednak żółty, powstaje z połączenia składowych R i G (po dokładne wyjaśnienie, odsyłam do punktu 5.4.2). W prezentowanym opisie jako czynnik określający kolor żółty, podana była różnica składowych R i G, która powinna być większa niż 20. Liczba ta wynikała z przeprowadzania szeregu eksperymentalnych testów i jest słabym miejscem algorytmu. Jeśli któraś z wymienionych składowych będzie odrobinę odbiegać od przyjętej podstawy, to choć kolor się zmieni zaledwie delikatnie w odcieniu, to program może go źle zinterpretować. Tak właśnie działo się podczas testów. Strefy wyżyn barwione kolorem, o składowej R, przekraczającej zakładaną wartość, rozpoznawane były jako góry. Z drugiej strony, gdy przewagę miała składowa G, punkty rozpoznawane były jako należące do nizin.
Problem przedstawiania kolorów i ich niezgodności poruszony był już podczas założeń teoretycznych, a jego złożoność pokazały testy. Rozwiązaniem okazał się eksperymentalny dobór, wartości zależności między składowymi.

Błędne piksele na mapach fizycznych

Choć problem ten był już poruszany, należy przytoczyć go jako, przykład do testowania poprawności działania algorytmów rozpoznawania terenu, przy pomocy aplikacji do wizualizacji. Mam tu na myśli, iż pojedyncze piksele źle rozpoznane na mapie fizycznej, nie były widoczne „gołym okiem", zwłaszcza przy dużych rozdzielczościach, w których pracował Windows. Pomocą przy ich identyfikacji, okazał się moduł do wizualizacji terenu. Błędne piksele, podczas umiejscawiania w przestrzeni tworzyły charakterystyczny efekt spiczastych wierzchołków wyrastających pośrodku danego terenu, lub zapadających się w niego. Było to bodźcem do stworzenia procedur umożliwiających modyfikację, już wygenerowanej mapy cyfrowej.

Rozwiązaniem tego problemu jest myśl, by stworzyć, w kolejnych wersjach oprogramowania, algorytmy przeszukujące całą tablice pod kątem anomalii terenu. Badane byłyby na przykład kilka lub kilkanaście pikseli na raz. Jeśli pośród grupy przeważającej znajdowałyby się piksele o znacznie różniących się wysokościach, byłyby one usuwane i zastępowane wartościami średnimi, dla badanego obszaru. Pomysł zautomatyzowałby wyszukiwania i usuwanie błędów, ale niesie ze sobą sporo pracy i możliwości nieprawidłowego działania, które również należałoby identyfikować i usuwać.

Błędy podczas wizualizacji terenu

Choć zasadniczo pomysł wizualizacji przestrzennej mapy, był dobrze przemyślany, wkradł się w jego realizację niewielki błąd, wyjątkowo rażący podczas wykonywania aplikacji. Dla przypomnienia i wyjaśnienia powiem, iż siatkę terenu tworzyłem, przez dobranie po cztery punkty i umiejscowienie ich w przestrzeni tworząc czworobok. Teoria wyglądała świetnie, a implementacja przynosiła ciekawe rezultaty. Jednak jednym z błędów, przeoczonych w pierwszej fazie testów było nieprawidłowe pobieranie danych z tablicy. Mam tu na myśli pobieranie danych z „powietrza". Błąd ten wynikał z braku zabezpieczenia, przed niepełną liczbą punktów w tablicy. Jeśli pobierane, po cztery, punkty doszły do końca tablicy, w której znajdowały się już jedynie dwa punkty, pozostałe były brane z pamięci operacyjnej komputera tworząc śmieci. Powstawał efekt, nierównych i poszarpanych końców mapy. Był to tym bardziej

trudny do testowania problem, gdyż nie występował na wszystkich mapach. Pojawiał się jedynie podczas wizualizacji map, w których liczba punktów była różna od którejś z potęg liczby cztery.

Rozwiązaniem okazały się drobne modyfikacje pętli pobierających punkty, tak by kończyły po stwierdzeniu, iż nie będzie już wystarczającej ilości danych do pobrania. Co prawda metoda ta traciła pewną gamę informacji, ale były to zaledwie końcowe piksele, a ich strata jest właściwie niezauważalna. Jak pisałem już w pracy, często jest to droga, wyboru kompromisów.

Błędy tworzenia środowiska wizualizacji

Sporą trudność i wiele błędów przyniosło testowanie środowiska wizualizacji terenu. Ponieważ zaczerpnięte z Internetu szkielety przestrzeni 3D były zbyt proste, powstały potrzeby ich modyfikacji i rozwijania. Niestety moja wiedza, zwłaszcza w początkowej fazie, była niewielka, co przysporzyło mi wiele problemów. Jednym z nich było nieprawidłowe włączanie oświetlenia. Oprócz głównego, związanego z oknem i inicjowanego w części GLInit, w szkielecie znajdowało się jeszcze jedno, dodatkowe źródło światłą. Jego aktywacja nie pozwalała mi właściwie prezentować kolorów.
Rozwiązanie okazało się banalne, wystarczyło włączać i wyłączać dodatkowe źródło światłą i efekty z nim związane. Pokazuje to jednak, jak często bark podstawowej wiedzy z danego zakresu, utrudnia, a czasem również zahamowuje rozwój danych myśli.

Błędy analizy geograficznej

Choć w module do analizy, założone były proste algorytmy, testy, zwłaszcza podczas projektowania pokazały, ile w ich teoretycznych przewidywaniach znajdowało się błędów.
Jednym z większych, który przysporzył wiele kłopotów był algorytm do obliczania rzeczywistej odległości na cyfrowym modelu terenu. Początkowo był on testowany tylko dla dwóch punktów (przypominam, że ostatecznie jest możliwe wykonanie obliczenia, dla znacznie większej ilości). Obliczenie ilości pikseli pomiędzy danymi punktami, działało prawidłowo, jednakże, wyliczone odległości wydawały się

niezgodne z prezentowanym terenem. Łatwo było to zauważyć, gdy podczas testów zaznaczało się punkty na równinie, a następnie, na równinie i górach. Te pierwsze dla odległości naturalnych i rzutowanych (więcej informacji w punkcie 6.5.2.), powinny być podobne, natomiast drugie powinny znacząco się różnić, tak się jednak nie działo. Początkowo, sprawdzenie poprawności algorytmu nie dało większych rezultatów. Działania arytmetyczne były poprawne, ale wyniki wychodziły błędne.

Problem rozwiązałem poprzez dodanie kodu umożliwiającego kolorowanie, analizowanych pikseli. Okazało się, że choć obliczenia arytmetyczne są rzeczywiście poprawne, to wykonywane były na złych pikselach. Źle wytyczony kierunek obliczania pikseli, prowadził do nieuniknionych błędów.
Dzięki wizualizacji postępowania algorytmu, stało się możliwe rozwiązanie błędu. Końcowy efekt okazał się dobry, a pomocny kod, kolorujący piksele drogi między punktami, pozostawiony dla podkreślenia efektu działania funkcji.

Tak wiec wykrywanie i usuwanie błędów było związane bezpośrednio z implementacją. Dzięki temu połączeniu, aplikacje wynikowe, stały się bardziej niezawodne i z całą pewnością rygorystycznie sprawdzone.

7.2. Określenie optymalnych warunków pracy

Poszukiwanie najwydajniejszych warunków dla pracy programów wchodzących w skład projektu nie było rzeczą trudną. Zasadniczo sprawa uprościła się już na poziomie wybierania środowiska programistycznego. Ponieważ Delphi tworzy oprogramowanie dla rodziny systemów Windows, dołączając wszystkie niezbędne komponenty podczas kompilacji, w tych systemach nie będzie kłopotów z uruchomieniem generatora map cyfrowych i analizy. Pomijam w tym miejscu moduł wizualizacji przestrzennych, z powodów do których zaraz dojdę.

Wykorzystane podczas tworzenia aplikacji komponenty Delphi, są jednymi z najprostszych w tym środowisku. Dzięki temu nie było konieczne dołączanie dodatkowych bibliotek, których brak mógłby zakłócić działanie programu. Sam system operacyjny Microsoftu w zupełności wystarczy.

Niektóre obliczenia zajmują jednak sporo czasu i na komputerach, wyposażonych w słabsze jednostki obliczeniowe, może nieco się wydłużyć. Należy jednak w tym miejscu zauważyć, iż przedłużenie, w żaden sposób nie wpłynie na samo wykonanie operacji.

Początkowo zastanawiałem się nad tablicami pamięciowymi. Przydzielone im w pamięci operacyjnej miejsce zależy od wielkości mapy fizycznej. Jednak doszedłem do wniosku, że proste wielkości całkowite, nie będą sprawiały dużych problemów, dla systemów nawet z małą ilością dostępnej pamięci operacyjnej.

Nieco inna sytuacja, jest z modułem wizualizacji przestrzennej. Działając w OpenGL, niezbędne staje się zapewnienie mu odpowiednich bibliotek. Jednak w nowszych systemach operacyjnych, gdzie są one bezpośrednio wgrywane podczas instalacji, problem ten przestaje istnieć. Faktem jest jednak, że obciążenie podczas generowania siatki jest dość znaczne o czym powiedziane zostanie w kolejnym podrozdziale.

7.3. Określenie wymagań środowiskowych

Tak jak, napisałem na poprzedniej stronie trzy programy wchodzące w skład projektu pracują pod systemami Windows. Testowanie aplikacji odbywało się pod Windows 98, Me, 2000 i XP. Jednocześnie przystosowanie do okienkowego trybu pracy (zwłaszcza w Windows XP) zapewnia zgodność z ewentualną zmianą samego wyglądu okien i przezroczystości.
Po testach na różnych komputerach z których najsłabszy był, wyposażony w procesor PII 300 MHz i 64 MB RAM, zauważyłem, iż jedyne problemy mogą występować tylko w module wizualizacji przestrzennej. Pozostałe dwa moduły działają podobnie na dowolnym sprzęcie komputerowym.

Wizualizacja jest jednak złożoną czynnością i ona może przysparzać problemów.
Wynika to z tego, iż siatka zbudowana jest często z tysięcy czworoboków, a obliczenia związane ze skalowaniem, obracaniem i przeliczaniem kolorystyki bardzo obciążają system. Pragnę jednak zaznaczyć, iż pomimo wolnych reakcji prezentacja 3D była generowana z zachowaniem pełnej kolorystyki i złożoności.
W tych jednak przypadkach, gdy prezentowana siatka przestrzenna jest wolno analizowana, proponuje zrobić dwie czynności:
- Włączyć wypełnianie siatki. OpenGL dokonuje obliczeń tylko widocznych części bryły, co znacznie przyspieszy proces generowania sceny.
- Zwiększyć wartość poziomu precyzji. Spowoduje to zwiększenie czworoboków siatki. To z kolei sprawi, że będzie ona znacznie mniej złożona, a przez to szybciej obliczana i generowana.

Podsumowując. Do wydajnej pracy projekt wymaga:
- systemu rodziny Windows,
- procesora z częstotliwością taktowania 1GHz lub wyższą,
- przynajmniej 128 MB RAM (w zależności od systemu, w Windows XP więcej),
- zgodności z OpenGL,
- od kilku do kilkunastu MB wolnego miejsca na dysku twardym;

8. Zakończenie

Projekt jest gotowy. Programy wchodzące w jego skład skompilowane i przetestowane. Niniejszy tekst, będący w pewnym sensie esencją nabytej wiedzy, praktycznie jest już napisany. Pozostaje więc podsumować wykorzystaną i nabytą wiedzę. Dokonać prostych analiz, by ostatecznie wyciągnąć wnioski, które stanowią podstawę naszych umiejętności,

Pora zakończyć tą pracę.

8.1. Podsumowanie realizacji projektu

Dla przypomnienia, celem pracy było opracowanie i stworzenie systemu generującego cyfrowe modele terenu na podstawie map fizycznych. Projekt okazał się sporym wyzwaniem i muszę przyznać, że jego dopracowanie wymagało dużego zaangażowania i wysiłku.

Początkowo zbierałem jak najwięcej wiadomości na temat cyfrowych modeli terenu. Wiedzę, w podstawowym zakresie, na tema map fizycznych, wyniosłem już z poprzednich lat nauki, więc teraz jedynie musiałem ją usystematyzować. Skupiłem się więc na technologiach komputerowych. Dzięki literaturze dowiedziałem się sporo na temat Systemów Informacji Geograficznej, które, choć obejmują znacznie większy zakres, dotyczą bezpośrednio zagadnienia tej pracy.

Zapoznałem się z formatami używanych danych i dużym zbiorem różnych aplikacji. Pozwoliło mi to wykreować pewną wizję późniejszych działań. Nad całością moich poczynań, początkowo dość chaotycznych, sprawował piecze, niesłychanie cierpliwy promotor, co bardzo pomagało w pracy.

Z teoretycznymi podstawami, zabrałem się do tworzenia schematów i modeli przyszłych programów. Wiele z problemów, które miały one rozwiązywać opracowane było wcześniej, a ich doszlifowania następowało podczas implementacji.

Istotne, jest również, iż część zmian i pomysłów, powstała już podczas pisania samych programów. Podział na trzy moduły, o czym była mowa w rozdziale 6, jest tego dobrym przykładem. Implementacja, była właściwie największym wyzwaniem. Trwała wiele miesięcy, ale powolne postępy, pozwalały cieszyć się rezultatami.

Końcowy efekt, spełnił pokładane nadzieje.

Choć powstałe w ramach projektu oprogramowanie, nie jest wolne od błędów, to działa dość sprawnie, spełniając postawione wymogi i założenia.

Po zakończeniu implementacji i testowania, zabrałem się za ostateczne poprawki i dopracowanie zarówno programów, jak i tekstu. Pozostało więc, przeanalizowanie prac nad projektem, przyjrzenie się aplikacjom i wyciągnięcie ostatecznych wniosków.

8.2. Analiza możliwości zastosowania projektu

Ponieważ praca stanowiła projekt akademicki, sam sens jest istnienia jest jasny. W związku z tym projekt w obecnej postaci może być jedynie wyjściem do dalszego rozwoju.

Ponieważ uzyskane dzięki analizie kolorystyki mapy, dane o wysokościach, nie są w stu procentach wiarygodne, nie może być stosowany jako typowy system GIS. Niedogodność ta wynika z błędów samej kolorystyki mapy i jej prawidłowego rozpoznawania, o czym mowa była już wcześniej. Należy jednak zauważyć, że sama rzeźba terenu, bez uwzględniania dokładnej skali, zachowuje proporcje, a tym samym może być doskonale prezentowana. Zwłaszcza dotyczy to wizualizacji przestrzennej, do czego wrócę w kolejnym podrozdziale.

Tak więc, już na tym etapie program mógłby być podstawą dla elementów obrazowania, znanych z systemów GIS.

Dodatkowo można zwrócić uwagę na fakt, iż proste obliczenia statystyczne, wyliczające informacje numeryczne o terenie, również mogą być w pełni wykorzystywane. Ta część z nich, która bezpośrednio nie zależy od obliczanej wysokości, dokonuje prawidłowych, nie obciążonych błędami operacji. Może więc dobrze służyć użytkownikowi.

Podsumowując tą prostą analizę możliwości zastosowania programów, powstałych w ramach tego projektu, można powiedzieć:

- Dobrze będą nadawały się do wizualnej prezentacji zeskanowanych map fizycznych, zwłaszcza gdy dokładne informacji o wysokościach, nie będą potrzebne. Istnieje szereg zastosowań, w których modelowanie, prezentowane przez programy tego projektu, będą przydatne, np. wszelkiego rodzaju pokazy i prezentacje omawiające rzeźbę terenu, lub położenie obiektów.

- Dobrze, lub wręcz doskonale będą tworzyć proste statystyki opisujące dane obszary mapy. Może być to ciekawa alternatywa dla biur i urzędów geodezyjnych, choć rzecz jasna jest to zbyt duże uproszczenie.

- Całość może świetnie nadawać się dla szkół zwłaszcza na zajęciach z geografii, by prezentować mapy fizyczne, w znacznie bardziej zaawansowany i efektowny sposób.

Jak więc widać, choć prostota projektu, nie pozwala stosować go w profesjonalny sposób, to jako pomoc naukowa, dla szkół, urzędów i innych instytucji, może stanowić o jego potencjale.
Łatwa obsługa i efektowne przedstawianie terenu mogą zaciekawić stosunkowo szerokie grono użytkowników.

8.3. Analiza możliwości rozwoju projektu

Zasadniczo rzecz ujmując możliwości rozwoju aplikacji, wchodzących w skład projektu, ograniczone są jedynie pomysłowością. Wzorując się jednak na szeregu programów, związanych z GIS, można zwrócić uwagę na kilka elementów, w które niewątpliwie przydałoby się rozbudować projekt. Z całą pewnością takim elementem jest baza danych, przechowująca informacje o atrybutach dowolnych punktów na mapie. Atrybuty te nie musiałyby być przypisane wszystkim punktom, lecz jedynie wybranym. Dzięki temu możliwe byłoby dodawanie, nieograniczonej grupy nazw, wartości, czy opisów. Umożliwiałoby to wzbogacenie obrazu przestrzeni w dodatkowe informacje. Mapa stałaby się bardziej przydatna.
Wykorzystując, na przykład, moduł analizy, w którym mamy możliwość obliczania odległości, oraz informacje z bazy danych, o miastach, można by wyliczyć drogę, którą mamy przebyć między tymi miastami. Dzięki bazie danych, mapa mogłaby zawierać nazwy rzek, gór, dolin. Można by umieszczać informacje o sumach opadów, lub średnich temperaturach. Jak więc widać, byłby to nieoceniony dodatek.

Kolejnym pomysłem na rozwój projektu, jest udoskonalenie modułu wizualizacji przestrzennej. Gdyby zmienić jego ogólne przeznaczenie i wzbogacić szatę graficzną, w teksturowanie, efekty pogodowe (jak mgłę, czy deszcz), mógłby powstać ciekawy program do generowania fotorealistycznych krajobrazów. Jednocześnie, pracując cały czas na ustalonym (lub zmodyfikowanym) formacie danych, mógłby on generować krajobrazy istniejących okolic, a obrazowanych przez przetwarzane mapy fizyczne.

Moduł wizualizacji przestrzennej dodatkowo wzbogacony o pewne „efekty specjalne”, mógłby stać się świetną aplikacją. Jednym z takich efektów, może być „przelot” nad terenem, opisany jakąś trasą. Łącząc możliwości modułu analizy (w którym, wspomniana procedura obliczania trasy, kreśli drogę między punktami), z wizualizacją przestrzenną, otrzymamy trasę przelotu. Stosunkowo proste przeliczenia ustawiałyby kamerę w odpowiednim miejscu i wysokości, a kontrolka „Timer”, zapewniłaby czasową zmianę pozycji, symulując lot. Ta prosta animacja, znacząco wzbogaciłaby prezentację.

Ostatecznie zmiany i ewolucja projektu dotyczyć mogą wielu mniejszych zagadnień. Omawiany w pracy format danych, mógłby zawierać opracowaną już kompresję. Jej udoskonalenie byłoby już tylko formalnością.

Jak więc widać projekt mógłby być z powodzeniem rozwijany, tak by w przyszłości stać się ciekawą i przydatną rodziną aplikacji do generowanie, wizualizacji i analizy map cyfrowych.

8.4. Wnioski końcowe

Podstawą naszego uczenia, jest zdolność wyciągania wniosków. Każde działanie, każde nowe zdolności i informacje, powinny prowadzić do wnioskowania. Dzięki temu rozwijamy się, a zdobyta wiedza pozwoli lepiej działać w przyszłości, lub unikać popełniania błędów. Tak również jest z niniejszą praca. Jest ona właściwie zakończona. Są to ostatnie linijki. Czas wyciągnąć wnioski.

Prace nad projektem rozciągały się na przestrzeni wielu miesięcy. Pokonywanie kolejnych przeszkód, oprócz nabytej wiedzy, dawały spora satysfakcję. Był to jeden z największych projektów, jakie dotychczas miałem okazję robić. Pozwolił mi on, zwłaszcza, nauczyć się systematyki pracy. Kolejne moduły i ich opracowanie, które przysparzały, czasem wielu problemów, wymusiły zaciętość. To nieoceniona zdolność, z całą pewnością mogąca przydać się przy pracach nad kolejnymi zagadnieniami, jak również w codziennym życiu.

Kończąc prace nad projektem, doszedłem do wnioski, iż chciałbym go dalej rozwinąć. Powstało wiele nowych pomysłów, dotyczących jego udoskonalenia i rozwoju. Choć jest i może zawsze będzie prostą pracą, pozwolił mi nauczyć się bardzo wiele.
Zdobyta wiedza może być przydatna przy projektowaniu przestrzennym, gdyż wiele się dowiedziałem (zwłaszcza o OpenGL), przy analizie kolorów i rozpoznawaniu obiektów na mapach bitowych.
Najważniejsze jednak jest, iż prace udało się zakończyć, odnosząc niewielki sukces, w tym rozumieniu, iż osiągnięte zostały, założone na wstępie cele.

Praca ta, dała mi wiele, czego dowodem jest ona sama.

Dodatek A

Słowniczek

CAD – (ang. Computer Aide Design); oprogramowanie służące wspomaganiu projektowania

Digitalizacja – proces przetwarzana dowolnych danych analogowych, na ich cyfrowe odpowiedniki (przypadku tej pracy, patrz skanowanie)

DWG – CAD-owski format zapisu stosowany w AutoCAD-zie

DXF – najpopularniejszy CAD-owski format wymiany danych

GIS – (ang. Geographical Information System); komputerowy system wspomagania tworzenia, przechowywania i analizy map cyfrowych

GPS – (ang. Global Positioning System); system umożliwiający dokładne określenie pozycji danego obiektu na kuli ziemskiej

Klient-Serwer – technika komunikacji polegająca na przechowywaniu danych na specjalnym serwerze, który udostępnia je klientom (komputerom w sieci) na ich żądanie

Mapa wektorowe – mapa zawierająca opis każdego elementu mapy w postaci geometrycznej (punkty, linie, figury)

Mapa rastrowe – mapa opisująca dany obszar za pomocą siatki kolorowych punktów

Shapefile – format zapisu danych przestrzennych stosowany a ArcView

Skanowanie – proces zamiany informacji przechowywanej na tradycyjnym nośniku, np. papierowym, na obraz przechowywany w postaci pliku graficznego

Dodatek B

Literatura

Boduch A.: „Delphi 7. Kompendium programisty”, HELION; Gliwice 2003

Davis D. E.: „GIS dla Każdego”, MIKOM; Warszawa 2004

Litwin L., Myrda G.: „Systemy Informacji Geograficznej; Zarządzanie danymi przestrzennymi w GIS, SIP, SIT, LIS”, HELION; Gliwice 2005

Morrisom J., Sale R., Robinson A.: „Podstawy kartografii”, Wydawnictwo Naukowe PWN; 1988

Myrda G.: „GIS czyli mapa w komputerze”, HELION; Gliwice 1997

Ormeling F., Kraak M.J.: „Kartografia – wizualizacja danych przestrzennych”, Wydawnictwo Naukowe PWN; 1998

Reisdorph K.: „Delphi 4 Dla każdego”, HELION; Gliwice 1999

Star J., Estes J.: „Geographic Information Systems: An Introduction”, Ptentice Hall; 1990