11

MINISTERSTWO NAUKI I INFORMATYZACJI
AKADEMIA OBRONY NARODOWEJ

Olga PIASNA
Andrzej STASIŃSKI
Jan ZYCH

ZAPROJEKTOWANIE I IMPLEMENTACJA MULTIMEDIALNYCH INTERFEJSÓW UŻYTKOWNIKA W SYMULATORZE OPERACYJNO-TAKTYCZNYCH DZIAŁAŃ POWIETRZNYCH

WARSZAWA 2004

MINISTERSTWO NAUKI I INFORMATYZACJI
AKADEMIA OBRONY NARODOWEJ

Olga PIASNA
Andrzej STASIŃSKI
Jan ZYCH

ZAPROJEKTOWANIE I IMPELMENTACJA MULTIMEDIALNYCH INTERFEJSÓW UŻYTKOWNIKA W SYMULATORZE OPERACYJNO-TAKTYCZNYM DZIAŁAŃ POWIETRZNYCH

WARSZAWA **2004**

Poszczególne części opracowali:

1. Olga Piasna: współautor rozdziału 3.
2. Andrzej STASIŃSKI: współautor rozdziałów: 2 i 4.
3. Jan ZYCH: wstęp, rozdział 1 i podsumowanie, oraz współautor rozdziałów: 2, 3 i 4.

Wyjaśnienie skrótów użytych w opracowaniu

AAL – ang. ATM Adaptation Layer - warstwa adaptacji ATM.

ACELP – ang. Algebraic Code Exited Linear Prediction algebraiczne CELP.

ADPCM – ang. Adaptive Differential Pulse Code Modulation adaptacyjna różnicowa modulacja impulsowo-kodowa.

AI- ang. Atriculation Index wskaźnik wyrazistości.

AON – Akademia Obrony Narodowej.

API - Application Programming Interface - interfejs dla programów użytkowych; zbiór formatów wymiany informacji i elementów języka, których aplikacje używają odwołując się do funkcji realizowanych przez inne programy, systemy komunikacyjne albo sterowniki sprzętu.

ATM- ang. Asynchronous Transfer Modl technika asynchronicznego przekazu danych.

ATMARP - ang. ATM Address Resolution Server serwer umożliwiający odnalezienie adresu ATM na podstawie adresu IP

B-ISDN - ang. Broadband-ISDN sieć szerokopasmowa stanowiąca wąskopasmowej sieci ISDN.

BUS - ang. Broadcast Unknown Server Serwer rozgłoszeniowy.

CBR - ang. Constant Bit Rate usługa o stałej wartości przepływności binarnej.

CELP - ang. Code Exited Linear Predictive liniowe kodowanie predykcyjne wyzwalane kodowo.

CIF - ang. Common Intermediate Format

CSCELP - ang. Conjugate Structure CELP CELP o zmiennej strukturze.

DCT - ang. Discrete Cosine Transform dyskretna transformata kosinusowa

DSBV - ang. Double Stimulus Binary Vote dwubodźcowa metoda oceny jakości wideo ze skalą dwuwartościową.

DSCQS - ang. Double Stimulus Continuous Quality Scale dwubodźcowa metoda oceny jakości wideo z ciągłą skalą jakości.

DSIS - ang. Double Stimulus Impairment Scale dwubodźcowa metoda oceny jakości wideo ze skalą pogorszeń jakości.

GCRA - ang. Generic Cell Rate Algorithm algorytm monitorowania zgodności napływu komórek generowanych przez źródło z parametrami zadeklarowanymi w kontrakcie

IBM PC - ang. International Business Machine Personal Computer.

HRC - ang. Hypothetical Reference Circuit hipotetyczny obwód odniesienia.

HVS - ang. Human Vision System system percepcji człowieka.

ISDN - ang. Integrated Services Digital Network sieć cyfrowa z integracją usług.

ITU- ang. International Telecommunication Union Międzynarodowa Unia Telekomunikacyjna - organizacja zajmująca się standaryzacją w dziedzinie telekomunikacji.

IP - ang. Internet Protocol protokół międzysieciowy.

LAN – ang. Local Area Network - dowolna technika sieci fizycznych, która została tak opracowana, aby sieć była zainstalowana na małym obszarze (odległość do kilku kilometrów).

LANE - ang. LAN Emulation standard emulacji sieci LAN w sieci ATM.

LDCELP - ang. Low Delay CELP CELP o małym opóźnieniu.

LECS - ang. LAN Emulation Configuration Server serwer konfiguracyjny systemu LANE.

LES - ang. LAN Emulation Server serwer sieci emulowanej w systemie LANE.

MAN - ang. Metropolitan Area Network sieć miejska.

MBit/s - prędkość transmisji danych w megabitach na sekundę.

MBS - ang. Maximum Burst Size maksymalny rozmiar paczki komórek.

MOS - ang. Mean Option Score metoda uśrednionej opinii słuchaczy.

MPEG - ang. Moving Picture Expert Group Grupa Ekspercka Obrazów Ruchomych.

MP-MLQ - ang. MultiPulse Maximum Likelihood Quantization wieloimpulsowa kwantyzacja z maksymalnym prawdopodobieństwem.

MS – Microsoft.

NATO - ang. North Atlantic Treaty Organization – Organizacja Traktatu Północnoatlantyckiego.

PCM - ang. Pulse Code Modulation modulacja kodowo-impulsowa.

PCR - ang. Peak Cell Rate wartość maksymalna szybkości generowania komórek.

PSQM - ang. Perceptual Speech Quality Measure percepcyjny pomiar jakości mowy.

PVC - ang. Permanent Virtual Connection stałe połączenie wirtualne.

RLC - ang. Run Length Coding kodowanie ciągów.

SO – system operacyjny.

SRC - ang. Sustainable Cell Rate graniczna wartość średniej szybkości transmisji komórek.

SS - ang. Single Stimulus metoda jednobodźcowa oceny jakości wideo.

SVC - ang. Switched Virtual Connection komutowane połączenie wirtualne.

TCP – ang. Transmissions Control Protocol - standard TCP/IP dla protokołu warstwy transportu. Protokół ten zapewnia pewną, w pełni dwukierunkową usługę, od której zależy wiele programów użytkownika. TCP pozwala procesowi na jednej maszynie wysłanie strumienia danych do procesu na innej maszynie.

TCP/IP - ang. Transmissions Control Protocol /Internet Protocol - zbiór sieciowych protokołów umożliwiających komunikację pomiędzy różnymi systemami pochodzącymi od wielu różnych dostawców.

TI - ang. Temporal Information informacja czasowa.

QCIF - ang. Quarter-CIF ¼ CIF.

QMF - ang. Qadrature Mirror Filter kwadraturowy filtr lustrzany.

VBR - ang. Variable Bit Rate usługa o zmiennej wartości przepływności binarnej.

VCI - ang. Virtual Channel Identifier identyfikator kanału wirtualnego.

VCL - ang. Variable Length Coding kodowanie o zmiennej długości słowa.

VO - ang. Video Objects obiekty wizyjne.

VOP - ang. Video Objects Plane płaszczyzna obiektu wizyjnego.

VPI - ang. Virtual Path Identifier identyfikator ścieżki wirtualnej.

WLOP – Wojska Lotnicze i Obrony Powietrznej.

Wstęp

Opracowanie niniejsze jest sprawozdaniem z badań z realizacji zadania 8, harmonogramu zawartego w opracowaniu: „Koncepcja realizacji projektu celowego". W ramach tych badań podjęto wysiłek adaptacji uznanych standardów multimedialnych do konstruowanego symulatora.

Naszym zdaniem, należy eksponować rolę programów z elementami multimedialnymi w kreowaniu nowych form prowadzenia ćwiczeń, treningów sztabowych, gier wojennych, ale dla komplementarności opracowania należało zaprezentować technologiczną stronę wykorzystania multimediów. Opracowanie zawiera koncepcję przeprowadzenia badań dotyczących wymiany danych mutimedialnych podczas grania, w czasie rzeczywistym, przy permanentnym monitorowaniu najbardziej istotnych procesów w grze.

Zadanie polegało na opracowaniu interfejsów z elementami multimedialnymi, w szczególności dźwiękowymi do symulatora operacyjno-taktycznych działań powietrznych. W badaniach skoncentrowano się na:

- opisaniu standardów kompresji danych multimedialnych na użytek programów komputerowych;
- opracowaniu oryginalnych skomponowanych efektów dźwiękowych;
- wykonaniu przykładowych nagrań komunikatów;
- przygotowaniu procedur do obsługi dźwięku na bazie komponentu dźwiękowego z pakietu DirectX: DirectSound.

Problem przygotowania odpowiednich procedur do obsługi dźwięku ma decydujące znaczenie z punktu widzenia technicznego (informatycznego). Użycie niewłaściwej metody implikuje wiele problemów dla wydajnego przetwarzania danych w symulatorze, jak również problemów wymiany danych między aplikacjami, będącymi częściami symulatora. Z drugiej jednak strony można zauważyć wiele pozytywów:

- odpowiednio generowane dźwięki w symulatorze będą istotnym elementem wpływającym na tzw. grywalność;

- będą przykuwać uwagę użytkownika na zdarzeniu, najważniejszym w danej chwili;
- pozwolą użytkownikom na większy komfort obsługi;
- dzięki dodatkowym efektom dźwiękowym osiągnie się wyższy standard produktu końcowego.

1. Multimedia w symulatorach i grach komputerowych

Na rynku gier i symulatorów coraz wyraźniej rysuje się tendencja nasycania tego typu narzędzi informatycznych elementami multimedialnymi. Ten etap prac miał na celu zmierzenie się z tym problemem i doposażenie konstruowanego symulatora w elementy multimedialne.

Pojęciu multimedia przypisuje się wiele znaczeń. Określenie produktów mianem multimedialnych stosowane jest jako skuteczny argument reklamowy. To co wykracza poza dziedzinę przetwarzania tekstów i liczb jest często kojarzone z pojęciem multimedia. Produkt informatyczny, gdzie administrowane jest chociażby kilka obrazków, gdzie do wybranej operacji przypisano komunikat dźwiękowy pretenduje do miana systemu multimedialnego.

Wydaje się jednak, że osiągnięto zgodność co do prawdziwości następującego stwierdzenia: dzięki technologii multimediów połączono w całość, w jedno medium, to co występowało oddzielnie. Jest oczywiste, że multimedia to coś więcej niż jednoczesna obecność różnorodnych mediów.

Bardziej znaczące jest przenikanie i równoczesny dostęp interakcyjny. Bardziej znacząca jest ich wszechobecność w nowoczesnych symulatorach zorientowanych na zastosowania edukacyjne, przede wszystkim w edukacji wojskowej. Warto upowszechniać pogląd, że są one jednym z istotnych komponentów usprawniających procesy kształcenia

Jednak wydaje się, że praktyczne problemy są niezmienne od lat. Ich istota sprowadza się do szukania odpowiedzi na pytanie: Jak uzyskać więcej (np. w kształceniu kadr, w kształtowaniu przewagi potencjału wiedzy i umiejętności) w krótszym czasie i za mniejsze pieniądze (choć nie od dziś wiadomo, że z triady dobrze-szybko-tanio maksymalnie dwa elementy mogą być jednocześnie spełnione).

Jeden z głównych nurtów zmian w kształceniu odwołuje się do odejścia od nauczania werbalnego na rzecz kształcenia multimedialnego. Nie oznacza to

rezygnacji z nauczyciela. Jednak w nawet w najbardziej renomowanych uczelniach świata, chyli się ku upadkowi era wykładów ex cathedra, gdzie głosi się autorytatywnie, w sposób niedopuszczający do dyskusji. Rynek wymusza nie tylko u nauczycieli akademickich coraz większą staranność w przygotowywaniu się do wykładów. Coraz istotniejsze staje się umiejętne zainteresowanie i skupienie uwagi studentów. Można nawet rzec, że w szczególności kadra unika zajęć sprowadzających się do podania suchych faktów, o których można przeczytać w książkach, skryptach, opracowaniach.

Właśnie nowoczesne narzędzia informatyczne z elementami multimedialnymi, poprzez możliwość oddziaływania na różne zmysły, poprzez uruchamianie wielu torów przesyłania informacji wpływają na optymalizację procesu kształcenia. Wymaga to jednak najczęściej dodatkowych kompetencji zarówno podających jak i tych, którym określone treści są podawane. Według prof. Strykowskiego kompetencje należy postrzegać jako zdolność do działania, jako syntezę trzech komponentów: wiedzy, umiejętności i postawy[1].

Zatem jak osiągnąć wysoką efektywność nauczania, zarówno w kategoriach jakościowych i ilościowych?. Osiągnięcie wysokich standardów kształcenia przy zastosowaniu tradycyjnych metod i środków staje się trudne, a czasami wręcz niemożliwe. Dlatego tak istotne stają się wszelkie pomoce dydaktyczne usprawniające przekaz, ale również umiejętności ich konstruowania i wykorzystania. Taka interpretacja to redukcja problemu i oczywiste uproszczenie, ale oddaje ogólny sens osiągania wysokich standardów kompetencji oficerów, absolwentów Wydziału Lotnictwa i Obrony Powietrznej Akademii Obrony Narodowej.

Infrastruktura telekomunikacyjna oraz potencjalne możliwości multimediów działających w czasie rzeczywistym, tych multimediów, które są adaptowane do symulatora, stwarzają jak nigdy dotąd możliwości edukacji,

---

[1] Strykowski W., Kompetencje nauczyciela szkoły współczesnej, Edukacja Medialna nr 4/2002, Wydawnictwo eMPi[2], Poznań 2002.

współpracy poprzez sieci komputerowe. Problemy wykorzystania sieci komputerowych dla potrzeb symulatora (wykorzystuje się protokół IP) nie są ani trywialne ani dobrze i wyczerpująco opisane w fachowej literaturze.

Stąd też wynikała potrzeba usystematyzowania tych zagadnień i podjęcie badań w celu określenia przepustowości sieci oraz implementacji dźwięków do symulatora operacyjno-taktycznych działań powietrznych.

Do znacznego upowszechnienia się multimediów w symulatorach mogło dojść dzięki szybkiemu rozwojowi metod kompresji danych, a przede wszystkim obrazu i dźwięku. Obecnie sygnał z danymi multimedialnymi można przesyłać już przy przepływności 28.8 [kbit/s]. Dodatkowym czynnikiem przyspieszającym adaptację elementów multimedialnych do symulatora jest standaryzacja dotycząca budowy programów komputerowych, na których mogą być odtwarzane efekty multimedialne, jak również elementów sieci niezbędnych do prawidłowego działania systemu. Z uwagi na rozproszony charakter przetwarzania danych multimedialnych wszystkie te aspekty są ważne i należy je brać pod uwagę podczas konstruowania symulatora.

Wraz z rozwojem metod kompresji sygnałów wizji i dźwięku łatwiejsze stały się prace nad metodami, które umożliwiłyby zbadanie jakości zaimplementowania kodeków, jak również ich wykorzystania, np. w symulatorach.

Sytuacja problemowa sprowadza się tu do wniknięcia w problem jakości połączeń realizowanych w sieciach IP, które zostały zestawione do testowania działania symulatora. W kolejnych rozdziałach zostanie zaprezentowany przegląd standardów kodowania danych multimedialnych wykorzystywanych w symulatorach, opis subiektywnych i obiektywnych metod badań jakości dźwięku i danych graficznych. Bez znajomości różnych metod nie byłoby możliwe dokonanie wyboru metody, która powinna być

zaimplementowana w symulatorze. Podwykonawcy do tego zadania (Olga Piasna i Andrzej Stasiński) opracowali badania jakości połączeń w symulatorze. Ponadto dołączono do opracowania najważniejsze fragmenty kodów źródłowych wraz z komentarzem, podprogramów wykorzystywanych w tym etapie prac.

2. Standardy kodowania danych multimedialnych

Rozwój nowoczesnych systemów multimedialnych, umożliwił równoczesną transmisję różnych typów informacji (obrazów, dźwięków itp.), np. w podstawowych zastosowaniach symulatorów istnieje możliwość łączenia dźwięku, danych cyfrowych, obrazów ruchomych i nieruchomych. Podstawowym warunkiem, koniecznym do pracy tych systemów jest dopasowanie urządzeń wymieniających miedzy sobą informacje. Stan pożądany to taki, gdzie urządzenia wchodzące w skład symulatora, a zorientowane na przetwarzanie danych multimedialnych spełniają międzynarodowe zalecenia dotyczące szybkości transmisji danych, protokołów komunikacyjnych i sygnalizacji.

Realizacja usług multimedialnych wiąże się z transmisją relatywnie bardzo dużej ilości informacji przesyłanych z dużą szybkością. Duże ilości informacji wymagają wydajnych technik kodowania (oraz przetwarzania informacji) stosowanych po to, aby w najkrótszym czasie przesłać jak najwięcej informacji o bardzo dobrej jakości. Szerokie możliwości rozwiązania tego problemu dają techniki kompresji sygnałów.

Metody kompresji danych można generalnie podzielić na dwie grupy: bezstratne i stratne. Kompresja bezstratna to zasadniczo zamiana danych źródłowych na dane o innej, oszczędniejszej w reprezentacji formie, dzięki której po zdekodowaniu będzie można odtworzyć sygnał oryginalny. Ten typ kompresji opiera się głównie na usuwaniu nadmiarowości, jest w pełni odwracalny i zapewnia brak zniekształceń. Inaczej jest w wypadku kodowania stratnego. Dopuszcza ono występowanie różnic pomiędzy sygnałem kodowanym i zdekodowanym, nie jest operacją odwracalną i daje w efekcie sygnał będący jedynie przybliżeniem sygnału oryginalnego. W systemach kompresji danych opartych na tej metodzie zniekształcenia

występują zawsze, ale ponieważ stopień kompresji jest bardzo wysoki, systemy te znajdują szerokie zastosowanie.

Drugi podział metod kompresji opiera się na sposobie kompresowania strumienia danych. Kompresja blokowa polega na przyporządkowywaniu kolejnym segmentom danych o stałej długości innych segmentów, też o stałej, ale mniejszej niż w oryginale długości. W danej chwili rozpatrywany jest tylko jeden blok, zaś bloki poprzednie i następne w stosunku do aktualnie rozpatrywanego nie mają żadnego wpływu na sposób kodowania. Jest to, więc metoda niewymagająca wielkich buforów na aktualnie przetwarzane dane. Natomiast kompresja nieblokowa polega na analizowaniu zachodzących na siebie segmentów danych o różnej długości. Wymagane są bufory o dużych rozmiarach oraz systemy synchronizacji przy dekodowaniu.

Jeszcze inaczej można metody kompresji podzielić na: predykcyjne i interpolacyjne. Predykcja to użycie wcześniejszych danych do przewidzenia wartości w kolejnym bloku. Interpolacja natomiast używa danych poprzednich i następnych do przewidzenia wartości próbki pomiędzy nimi. W obu przypadkach następuje porównanie pomiędzy aktualną i przewidywaną wartością w celu określenia wielkości błędu predykcji. Jeżeli jest on mały, próbka zostaje potraktowana jako nadmiarowa i jest usuwana ze strumienia danych. Jeżeli jednak błąd jest duży, próbka lub zmierzony błąd są transmitowane dalej. Zostanie to wykorzystane po stronie dekodera, gdy pojawi się tam ponownie błąd predykcji, w celu odtworzenia skompresowanego sygnału. W praktyce algorytmy predykcji i interpolacji wspólnie wykorzystują wielomiany n-tego stopnia do przewidzenia kolejnej wartości. Wielomian n-tego stopnia, na podstawie n+1 próbek, wyznacza nową wartość. Im większy stopień wielomianu, tym bardziej dokładna predykcja, ale i bardziej skomplikowana implementacja. Na szczęście okazuje się, że wielomiany stopnia pierwszego lub nawet zerowego są często

wystarczające do poprawnego przewidzenia kolejnej próbki, przez co wyżej wymienione metody są bardzo popularne.

Jeżeli dana metoda kompresji jest w stanie sama zmieniać lub modyfikować algorytm swojego działania w momencie wykrycia zdarzenia, że kompresja staje się nieefektywna lub zaczynają pojawiać się błędy, to nazywana jest adaptacyjną. Predykcyjna metoda kompresji ze zmieniającymi się w zależności od sytuacji współczynnikami wielomianu może być przykładem metody adaptacyjnej.

2.1. Standardy kodowania danych dźwiękowych

Celem algorytmów kodowania mowy jest transmisja, gromadzenie oraz synteza dźwięku przy ustalonej minimalnej ilości przenoszonej informacji. Tak, jak w przypadku innych algorytmów kodowania, również w przypadku kodowania dźwięku dąży się do usunięcia nadmiaru informacji.

Dal kodowanie sygnałów komunikatów dźwiękowych, w celu zachowania odpowiedniej jakości należy brać pod uwagę dwa najbardziej istotne czynniki. Po pierwsze, jak często jest próbkowany analogowy sygnał wejściowy. W tym wypadku, im częściej następuje próbkowanie, to znaczy im wyższa jest częstotliwość próbkowania, tym precyzyjniejszy będzie odwzorowanie sygnału wejściowego. Po drugie zmierzonym poprzez próbkowanie wartościom przypisuje się wartości z przyjętej skali. Im ta skala jest bardziej rozbudowana, tym bliższa zmierzonej wartości będzie przypisana jej wartość ze skali, a tym samym lepiej sygnał zostanie odwzorowany. Częstotliwość próbkowania musi być zatem, co najmniej dwa razy większa od częstotliwości sygnału podlegającego próbkowaniu. Dla przykładu: sygnał akustyczny o częstotliwości do 20 [kHz], konieczne należy próbkować częstotliwością co najmniej 40 [kHz]. Jest szereg narzędzi informatycznych, które zapewniają wybór odpowiednich opcji. Ważne by mieć to na uwadze i świadomie i z rozmysłem dokonywać wyboru.

W przypadku przygotowywania komunikatów dźwiękowych do symulatora ważne, by proces kodowania a następnie dekodowania utrzymywał odpowiednią jakość. Najczęściej zakłada się a priori minimalny poziom. Szerokość pasma sygnału mowy może być ograniczona do około 3,4 [kHz], bez utraty zrozumiałości przekazywanych treści. Do ograniczenia pasma można wykorzystać filtr o zadanej częstotliwości granicznej. W ten sposób obniży się na pewno ilość informacji potrzebnej, żeby opisać taki sygnał w celu jego rejestracji, a następnie transmisji.

W kolejnych podrozdziałach przedstawiony zostanie przegląd standardów kodowania wykorzystywanych w systemach multimedialnych opartych na kilku aplikacjach współpracujących ze sobą. Ważny w tym kontekście jest aspekt właściwości sposobów kodowania dźwięków.

2.1.1. Standardy organizacji międzynarodowej ITU

Międzynarodowa organizacja ITU opracowała kilka standardów, które dotyczą sposobów kodowania sygnału dźwiękowych. Zostały one opisane w serii standardów oznaczonych literą G. Ideą tworzenia standardów kodowania dźwięków było ujednolicenie stosowanych kodeków, a także jak najlepsze wykorzystanie dostępnego pasma przy jak najlepszych parametrach dźwięku.

2.2.1.1. Standard G.711

Podstawową metodą kodowania sygnału dźwiękowego jest modulacja impulsowo kodowa PCM. Sygnał wejściowy zgodnie z twierdzeniem o próbkowaniu, jest próbkowany z częstotliwością dwa razy większą od maksymalnej występującej w jego widmie. Po próbkowaniu wersja sygnału informacyjnego zostaje skwantowana, dając reprezentację sygnału dyskretną zarówno w czasie jak i w amplitudzie. Aby zmniejszyć tzw. szum kwantowania powstający w wyniku kwantyzacji, standard przewiduje użycie kwantyzatora z kwantowaniem nierównomiernym. Dla liniowej kwantyzacji sygnały o małych amplitudach podlegają większym zniekształceniom niż

sygnały silniejsze. Aby uzyskać kwantowanie nierównomierne niezbędne jest podanie próbek sygnału dźwiękowego na kompresor. Sygnał po kompresji poddawany jest kwantowaniu liniowemu. W układzie odbiorczym należy zastosować układ, o charakterystyce odwrotnej niż kompresora, zwany ekspanderem. Ze względu na niemożność uzyskania dokładnej repliki nieliniowej krzywej kompresji stosuje się odcinkami liniową aproksymację pożądanej krzywej.

Proponowane rozwiązanie przy częstotliwości próbkowania, np. 8 [kHz] i kodowaniu każdej próbki 8 bitami daje sygnał wyjściowy o przepływności 64 kbit/s. Dlatego też ten sposób kodowania jest wykorzystywany w terminalach przetwarzających dużą liczbę takich komunikatów.

2.2.1.2. Standard G.722

Taką samą przepływność jak na wyjściu kodeka G.711 uzyskamy wykorzystując kodek w standardzie G.722. Jednak kodek ten oferuje lepszą jakość dźwięku przy tej samej przepływności. Wynika to z tego, że jego sygnał wejściowy może zawierać się w paśmie od 50 Hz do 7 [kHz]. Dlatego też może być wykorzystany do różnorodnych aplikacji wymagających wysokiej jakości komunikatów dźwiękowych (przede wszystkim mowy). System ten wykorzystuje podpasmową adaptacyjną różnicową modulację impulsowo-kodową (SBADPCM) z przepływnością 64 [kbit/s]. W technice tej używane pasmo jest dzielone na dwa podpasma: wyższe (4000 ÷ 7000 [Hz]) i niższe (300 ÷ 4000 [Hz]), po czym sygnały każdego podpasma są kodowane za pomocą metody adaptacyjnej ADPCM. Metoda ta polega na adaptacyjnym kodowaniu różnicy między kolejnymi próbkami sygnału. Podziału na podpasma dokonuje się za pomocą kwadraturowych filtrów lustrzanych QMF. Podpasmo wyższe wykorzystuje przepływność 16 [kbit/s], a podpasmo niższe przepływność 48 [kbit/s], co w sumie daje 64 [kbit/s].

Standard G.722 umożliwia pracę w trzech trybach zależnie od przepływności użytej do kodowania: 64, 56, lub 48 [kbit/s]. Dwa ostatnie tryby dają możliwość wykorzystania pomocniczego kanału transmisji danych o przepływnościach odpowiednio 8 i 16 [kbit/s], będącego dopełnieniem do przepływności 64 [kbit/s], w tym celu wydzielane są bity przeznaczone dla niższego podpasma.

2.2.1.3. Standard G.723.1

Standardy opisane powyżej przedstawiają sposoby kodowania, które ze względu na konieczność współdzielenia pasma z sygnałem wizyjnym, nie mogą być zastosowane w terminalach obsługujących interfejsy dedykowane. Aby użytkownicy sieci mogli również komunikować się z wykorzystaniem dźwięku (a w przyszłości obrazu) został stworzony standard G.723.1. Kodek opisany w tym standardzie pracuje z dwoma wartościami przepływności 5,3 [kbit/s] i 6,3 [kbit/s]. Zaimplementowanie obu trybów pracy jest obowiązkowe dla kodera i dekodera. Oczywistym jest, że jakość dźwięku nie będzie taka sama, gdyż lepszą jakość uzyskuje się dla większej przepływności.

Kodek ten został zoptymalizowany tak, aby przedstawić dźwięk z wysoką jakością wykorzystując obie żądane przepływności przy jak najmniejszym skomplikowaniu układu. Wykorzystano tutaj dwie metody kodowania sygnału. Dla przepływności mniejszej koder z liniowym prognozowaniem o pobudzaniu kodowaniem algebraicznym ACELP, a dla wyższej koder o wieloimpulsowej kwantyzacji z maksymalnym prawdopodobieństwem MP-MLQ.

2.2.1.4. Standard G.728

G.728 jest standardem kodowania dźwięku z przepływnością bitową 16 [kbit/s], wykorzystującym metodę kodowania predykcyjnego o małym opóźnieniu LDCELP. Koder ten działa na zasadzie liniowej predykcji i

kwantyzacji wektorowej. Filtr predykcyjny jest dopasowywany do sygnału metodą adaptacji, tzw. „metoda wstecz". Podstawą tego kodera jest kodowanie CELP wykorzystujące tzw. „książeczkę kodową". Subiektywna jakość odtwarzanego komunikatu dźwiękowego zależy od łatwości dopasowania realnego dźwięku do dźwięków „standardowych" zawartych w spisie. Liczba dostępnych pozycji spisu jest ograniczona zarówno ze względu na dopuszczalną objętość książeczki kodowej, jak i czasu niezbędnego na przeszukanie spisu. Całkowite opóźnienie w tym koderze wynosi 0,625 [ms].

Układ książki kodowej wymuszeń generuje wstępną sekwencję, która jest z kolei wzmacniana i przepuszczana przez filtr syntetyzujący wysokość tonu. Różnica między sygnałem zsyntetyzowanym, a widmem sygnału oryginalnego jest następnie ważona w filtrze wagowym, po czym wytwarzany jest sygnał wyjściowy. Współczynniki filtru, wzmocnienie i generator są zmieniane w celu minimalizacji sygnały wyjściowego, który to przebieg jest kodowany za pośrednictwem kwantyzacji wektorowej.

2.2.1.5. Standard G.729

W standardzie G.729 opisany jest kodek sygnału mowy o niskiej przepływności bitowej 8 [kbit/s] wykorzystujący metodę kodowania predykcyjnego o zmiennej strukturze (CSACELP). Jest on rozwinięciem sposobu kodowania przedstawionego w standardzie G.728. Wykorzystuje on kodowanie ACELP, z tą różnicą, że kodek G.729 ma dodatkową adaptacyjną książkę kodową.

2.2.2. Standard MPEG

MPEG jest akronimem od Moving Picture Expert Group - grupy badawczej, która powstała w roku 1988, celem opracowania standardu kodowania obrazów ruchomych wraz z przynależnym do nich dźwiękiem. Część standardu dotycząca kodowania dźwięku nazwana jest MPEG-Audio.

Podobnie jak we wszystkich metodach kompresji pracujących ze stratami, grupa MPEG wykorzystała niedostatki możliwości percepcyjnych człowieka. Punktem wyjścia dla tej metody jest model psychoakustyczny. Z przetwarzanego sygnału usuwane są nie tylko informacje nadmiarowe, lecz także częstotliwości, które nie będą zauważone. Przy użyciu 32 filtrów pasmowych analizuje się spektrum częstotliwości akustycznych z podziałem, na podpasma. Sygnały leżące poza zakresem słyszalności człowieka nie są uwzględniane. Wykorzystuje się przy tym efekt maskowania - sygnały głośne zagłuszają sąsiadujące z nimi sygnały słabsze, tak wiec te słabsze są maskowane i nie muszą być rejestrowane.

Standard MPEG-Audio jest zaprojektowany do kodowania sygnałów z częstotliwościami próbkowania 32, 44.1 oraz 48 [kHz]. Sygnały foniczne mogą być, co najwyżej dwukanałowe. Z tym ograniczeniem wiążą się następujące tryby kodowania sygnału:

- tryb monofoniczny z pojedynczym kanałem dźwiękowym
- tryb niezależnych kanałów monofonicznych (funkcjonuje identycznie do trybu stereo)
- tryb stereo dla kanałów stereo (rozdziela bity, ale nie używa kodowania wspólnego-stereo)
- tryb wspólne-stereo.

Tryb wspólne stereo ma przewagę nad innymi trybami, ponieważ wykorzystuje korelacje pomiędzy kanałami stereo lub brak różnic fazy między kanałami, albo obydwie te zależności naraz. Wyróżnia się tutaj metody łącznego kodowania:

- kodowanie MS (Mid/Side) - wyznaczana jest znormalizowana suma M i różnica S sygnałów z kanału prawego i lewego przed kwantyzacją. Dzięki temu unika się zjawiska zanikania maskowania, występującego w przypadku sygnałów stereofonicznych. Ponadto dla silnie

skorelowanych sygnałów w obu kanałach nie trzeba przesyłać sygnału różnicowego.

- kodowanie głośności (ang. intensity coding) - zachowywana jest tylko informacja o intensywności sygnału w każdym podpasmie, dla którego wyznaczane są współczynniki skalujące. Informacja o kierunkowości sygnału przesyłana jest poprzez zakodowanie niezależnych wartości tych czynników skalujących dla lewego i prawego kanału. W rezultacie, transmitowana jest tylko obwiednia energii dla obu kanałów.

Standard MPEG umożliwia pracę ze stałą wartością przepływności binarnej CBR lub ze zmienną wartością przepływności binarnej VBR. Pierwszy przypadek wykorzystywany był w pierwszych wersjach standardu. Niemniej jednak, mniej złożone składowe sygnału dźwiękowego mogą być zakodowane przy użyciu mniejszej liczby bitów. Najprostszym przykładem jest pasmo ciszy, które można zakodować w trybie najmniejszej przepływności binarnej bez utraty jakości subiektywnej. Dla wielu zastosowań, w których sygnał foniczny jest dostarczany „na żądanie", opcja VBR jest niezwykle użyteczna. Skala możliwych prędkości bitowych sięga (dla obu kanałów) od 64 [kbit/s] do 448 [kbit/s]. Standard dopuszcza również kodowanie z prędkością 32 [kbit/s] dla jednego kanału.

W ramach standardu MPEG istnieje możliwość wyboru jednego z trzech trybów pracy, zwanych warstwami. Numer warstwy decyduje o uzyskiwanej jakości dźwięku: im wyższy numer warstwy, tym wyższa jakość dźwięku (wzrasta jednak złożoność kodera).

Cechą charakterystyczną kodeków MPEG jest to, że modelowanie zjawisk psychoakustycznych dokonywane jest jedynie po stronie kodera. Dzięki temu dekodery są znacznie prostsze, zaś przetwarzanie sygnału odbywa się przy niższym koszcie obliczeniowym.

2.3. Standardy kodowania obrazów

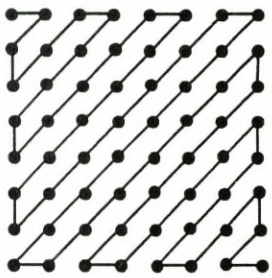
Pojawienie się sieci ISDN, która dostarcza przeciętnie pasmo 64 [kbit/s], umożliwiło powszechna komunikację przy pomocy obrazów ruchomych. Przy obniżeniu wymagań na jakość połączenia, wymaga się stosunkowo małych przepływności. Przekazywany obraz jest praktycznie statyczny, przedstawia zazwyczaj piktogram (lub zbiór piktogramów). Standardy kodowania obrazów wykorzystują to, że kolejno następujące po sobie obrazy są bardzo podobne, a więc kolejne klatki obrazu mogą mieć odniesienie do klatek sąsiednich, co w znacznym stopniu zmniejsza strumień przesyłanych bitów. Jednak, w celu uniknięcia kumulowania się błędów, niektóre klatki obrazu muszą być kodowane jako obrazy pełne nie mające odniesienia do innych. Stąd wynikają dwa podstawowe tryby pracy koderów:

- wewnątrzobrazowy (ang. intraframe);
- międzyobrazowy (ang. interframe).

Kodowanie wewnątrzobrazowe odbywa się niezależnie dla każdego obrazu, podczas gdy kodowanie międzyobrazowe wykorzystuje podobieństwa między kolejnymi obrazami.

Kodowanie wewnątrzobrazowe wykorzystuje najbardziej dziś rozpowszechnioną metodę kodowania obrazów, jaką jest dyskretna transformata kosinusowa DCT. Powodem tej popularności jest generacja niezależnych od siebie współczynników transformaty. W rzeczywistości obraz jest dzielony na niewielkie bloki o rozmiarze 8x8 pikseli, które są przekształcane oddzielnie. W wyniku przekształcenia kosinusowego dla każdego bloku punktów obrazu otrzymuje się blok współczynników transformaty, w którym największą wartość ma współczynnik DC (F(0,0)), który jest proporcjonalny do średniej wartości próbek bloku. Wartości kolejnych współczynników zwanych AC ($F(k_1,k_2)$), maleją, gdyż

odpowiadają większym składowym widma obrazu. Dzięki temu można im przypisać reprezentacje o mniejszej liczbie bitów.



Rys. 1. Ilustracja wzorca kodowania zig-zag

Współczynniki transformaty są kwantowane w celu zmniejszenia ilości informacji. Wykorzystywane są tutaj tablice kwantyzacji, które są definiowane dla każdej składowej obrazu. W tym momencie faktycznie dokonuje się zasadnicza kompresja, a związana z nią utrata informacji jest nieodwracalna. Następnie skwantowane współczynniki transformaty są szeregowane zgodnie z tak zwanym wzorcem zig-zag (rys. 1.

Dzięki takiemu uporządkowaniu uzyskuje się ciąg rozpoczynający się współczynnikiem DC, po którym następują 63 współczynniki AC. W takim ciągu często po kilka sąsiednich współczynników AC ma zerowe wartości. Ciąg można wtedy efektywnie przedstawić za pomocą odmiany kodowania ciągów RLC. Liczby określające kod RLC są następnie kodowane ze zmienną długością słowa VCL, za pomocą kodów Huffman'a.

Kodowanie międzyobrazowe korzysta w zasadzie z tych samych technik, co kodowanie wewnątrzobrazowe. Zasadnicza różnica polega na tym, że przetwarzaniu podlega nie bezpośrednio obraz, lecz błąd predykcji obrazu na podstawie obrazu poprzedniego. W najprostszym przypadku przewiduje się, że obraz jest identyczny z poprzednim. Kodowany błąd

predykcji jest wtedy po prostu różnicą między kolejnymi obrazami. Przesyłanie błędów predykcji pozwala na odtwarzanie w dekoderze kolejnych obrazów na podstawie obrazów wcześniej zdekodowanych. Często wykorzystuje się predykcję z kompensacją ruchu, która polega na wyznaczeniu wektorów ruchu określających położenie poszczególnych makrobloków (16x16 pikseli) w poprzednim obrazie.

Z wykorzystaniem kompensacji ruchu lub bez niej, wyznaczany jest przewidywany obraz, który porównuje się z rzeczywistym obrazem na wejściu kodera. Różnica między tymi dwoma obrazami określa błąd predykcji podlegający kodowaniu podobnemu do kodowania wewnątrzobrazowego. Składający się z bloków o rozmiarze 8x8 pikseli sygnał błędu predykcji jest poddawany dwuwymiarowej transformacji kosinusowej. Współczynniki uzyskanej transformaty są kwantowane, a następnie wewnątrz każdego bloku szeregowane zgodnie z uporządkowaniem zig-zag. Ciągi współczynników poddaje się kodowaniu ciągów RLC, a następnie kodowaniu o zmiennej długości słowa VLC. Tak przygotowana informacja dociera do dekodera. Predykcję międzyobrazową realizuje się za pomocą pętli sprzężenia zwrotnego, do której podawane są skwantowane współczynniki transformaty.

Makrobloki kodowane międzyobrazowo nazywane są makroblokami typu P (ang. predictive) dla odróżnienia od makrobloków kodowanych wewnątrzobrazowo oznaczanych jako I (ang. intraframe). Obraz składający się wyłącznie z makrobloków I nazywa się obrazem typu I. Obrazy typu I są dekodowane niezależnie od poprzednich obrazów. Obrazy składające się z makrobloków P lub I i P są obrazami typu P, które są dekodowane tylko po zdekodowaniu poprzednich obrazów.

Istnieje jeszcze inna odmiana kodowania międzyobrazowego polegająca na zastosowaniu interpolacji dwóch obrazów z kompensacją

ruchu. Obrazy zawierające makrobloki kodowane w ten sposób nazywane są obrazami typu B (ang. bidirectionally-predictive).

W kolejnym podrozdziale zostaną przedstawione standardy kodowania obrazów organizacji międzynarodowej ITU i grupy MPEG. Należy tu podkreślić, że standard ITU H.261 stał się wzorem dla innych standardów. Kodeki obrazów H.263 i MPEG2 stosowane w systemach multimedialnych są bardzo podobne do kodeka H.261.

2.3.1. Standardy organizacji międzynarodowej ITU

2.3.1.1. Standard H.261

Standard H.261 'Video codec for audiovisual services at px64 [kbit/s]' jest podstawowym standardem kompresji sygnału wizyjnego dla sieci ISDN. Głównym zadaniem opisanego tu schematu kodowania obrazu jest zredukowanie strumienia danych wyjściowych do poziomu akceptowalnego w sieci ISDN. Przy projektowaniu kodera zakładano, że będzie go można wykorzystać w kanałach typu H0, jednak w późniejszym okresie okazało się, że zwiększenie współczynników kompresji nie powoduje nie akceptowalnego pogorszenia obrazu i umożliwia przekazywanie wizji w kanale typu B. Parametr p określa liczbę wykorzystywanych kanałów B i może przyjmować wartości od 1 do 30. Generowany strumień danych zawiera się w przedziale od około 40 [kbit/s] do prawie 2 [Mbit/s]. Wykorzystanie kanałów sieci ISDN narzuca charakter ruchu generowanego na wyjściu kodera, jest to ruch o stałej przepływności binarnej.

Na wejście kodera H.261 podawany jest sygnał telewizyjny składający się z sygnału luminancji Y i dwóch sygnałów różnicowych koloru C_R i C_B. Częstotliwość powtarzania obrazu to około 30 razy na sekundę. Każda z próbek sygnału luminancji i sygnałów różnicowych koloru reprezentowana jest w postaci jednego bajtu. W celu zmniejszenia wizyjnego strumienia danych, lecz niestety obniżając jakość obrazu, zastosowano próbkowanie sygnału

analogowego ze stosunkowo niską częstotliwością, co wpłynęło na rozdzielczość obrazu. Standard H.261 dopuszcza cyfrowe obrazy wejściowe o dwóch formatach: CIF oraz QCIF. Format QCIF jest formatem obligatoryjnym dla kodeka H.261, natomiast format CIF jest opcjonalny. Strumień wejściowy kodera wynosi 36.5 [Mbit/s] dla formatu CIF i 9.1 [Mbit/s] dla formatu QCIF (przy 30 obrazach na sekundę). Współczynnik kompresji przyjmuje więc wartości od około 10:1 do ponad 200:1, co najlepiej wskazuje, jakiej jakości sygnału należy się spodziewać po stronie odbiorczej.

Standard ten przewiduje kodowanie obrazu z wykorzystaniem kodowania wewnątrzobrazowego i międzyobrazowego. Zalecenie narzuca, że każdy makroblok ma być kodowany w trybie wewnątrzobrazowym przynajmniej raz na 132 transmisje, co ma stanowić zabezpieczenie przed kumulowaniem się błędów.

Wychodzący z kodera ciąg bitów jest buforowany w celu wyrównania chwilowych niejednostajności wielkości strumienia. Ze względu na wielkość bufora maksymalna liczba bitów potrzebnych do zakodowania pojedynczego obrazu jest przez standard ograniczona do 256 [kbit] dla formatu CIF i 64 [kbit] dla formatu QCIF.

Standard ten jest przewidziany do transmisji kolorowych sekwencji obrazów o jakości od umiarkowanej (przepływność 64 [kbit/s] przy 7,5 obrazów na sekundę) do dobrej (przepływność 1536 [kbit/s] przy 30 obrazach na sekundę).

2.3.1.2. Standard H.263

Standard H.263 'Video codec for low bit rate comunication' jest rozwinięciem zalecenia H.261 i dotyczy kodowania cyfrowych sygnałów wizyjnych z bardzo małą przepływnością. Został on opracowany z myślą o wykorzystaniu w sieciach bezprzewodowych. Standard opracowywano,

biorąc pod uwagę przede wszystkim zastosowanie do systemów wymiany dźwięku i obrazów. Obecnie oczekuje się, że będzie używany w szerokim zakresie przepływności (nawet powyżej 1 [Mbit/s]) i zastąpi H.261 w wielu zastosowaniach.

Algorytm kompresji H.263 jest podobny do H.261, jednakże dokonano w nim pewnych zmian i udoskonaleń w celu zwiększenia efektywności i poprawienia odporności na błędy. Niektóre z wymienionych poniżej funkcji są obligatoryjne, inne stanowią opcje kodeka H.263:

- estymacja ruchu ma rozdzielczość zwiększoną do ½ odstępu między punktami obrazu;
- zakres wartości wektorów ruchu wynosi w obu kierunkach (-16; 15,5);
- wartości wektorów ruchu dla poszczególnych makrobloków są kodowane predykcyjnie, co powoduje zmniejszenie strumienia danych potrzebnego do ich zakodowania;
- możliwość generowania osobnych wektorów dla czterech bloków w ramach makrobloku;
- predykcja wartości punktu z uwzględnieniem wartości wektorów ruchu w sąsiednich blokach;
- możliwość wyznaczania wektorów ruchu wskazujących na punkty poza obrazem;
- wykorzystanie ramek PB, w których przesyłane są dwa kolejne obrazy sekwencji. Pierwszy z nich jest obrazem typu B, a drugi jest obrazem typu P. W trybie tym, przy umiarkowanym ruchu, uzyskuje się dokładniejszą predykcję kosztem dodatkowego opóźnienia w dekodowaniu strumienia danych. Ze względu na zastosowania w wideotelefonii i wideokonferencjach, zalecenie H.263 ogranicza liczbę sąsiadujących ze sobą obrazów typu B do jednego.

- Koder H.263 może pracować z sygnałami wizyjnymi z nowymi formatami: sub-CIF, 4CIF, 16CIF. Charakterystyka wszystkich formatów obrazu dostępnych w tym standardzie została przedstawiona w tabeli 1.3.

H.263+

Nazwa H.263+ oznacza opracowaną modyfikację zalecenia H.263, która uzupełnia poprzedni standard o:

- swobodny wybór formatu wejściowego;
- negocjowany wybór opcjonalnych trybów kodowania;
- zaawansowany tryb kodowania wewnątrzobrazowego – dotyczy zastosowania adaptacyjnej predykcji wartości współczynników DCT;
- filtr usuwający efekty blokowe;
- ulepszone kodowanie ramek PB;
- zmodyfikowane kwantowanie współczynników DCT.

2.3.2. Standardy komitetu standaryzacyjnego MPEG

2.3.2.1. MPEG-2

Grupa badawcza MPEG zajęła się standaryzacją kodowania obrazów ruchomych wraz z przynależnym do nich dźwiękiem do wykorzystania w aplikacjach multimedialnych. Standardy MPEG są oparte na metodzie kodowania zaproponowanej przez organizację standaryzującą ITU w zaleceniu H.261. Jednak grupie MPEG przyświecały nieco odmienne cele niż ITU. Poniżej przedstawione zostaną wymagania, jakie musi spełniać standard MPEG-2, który znalazł zastosowanie w komunikacji audiowizualnej w sieciach szerokopasmowych:

- swobodny dostęp do każdej klatki filmu;

- małe opóźnienia kodowania i dekodowania, aby możliwy był dostęp do obrazu w aplikacjach interaktywnych, całkowity czas dostępu nie może trwać dłużej niż jedną sekundę;
- synchronizm obrazu i dźwięku;
- szybkie wyszukiwanie w przód i wstecz;
- odtwarzanie wstecz, w tym trybie odtwarzania dopuszczalne są pewne kompromisy odnośnie jakości odtwarzanego obrazu, gdyż z reguły nie będzie ona musiała być taka wysoka jak przy odtwarzaniu standardowym;
- elastyczność formatu obrazu. Z uwagi na odtwarzanie obrazu w okienku ekranu, jego wymiary muszą być skalowane w poziomie i w pionie;
- mała wrażliwość na błędy, należy zapewnić odpowiedni mechanizm korekcji;
- kodowanie w czasie rzeczywistym.

Przyglądając się wymaganiom dla kodeka MPEG można powiedzieć, że będzie on bardziej rozbudowany niż kodek H.261 czy H.263.

W odróżnieniu od zaleceń ITU standard MPEG-2 określa stosowanie tablic kwantyzacji dla współczynników transformaty kosinusowej, przy kodowaniu wewnątrzobrazowym. Tablica kwantyzacji może być przesłana w nagłówku sekwencji. W przeciwnym razie stosowana jest tablica typowa, która uwzględnia właściwości percepcyjne ludzkiego systemu widzenia. Standard zezwala na adaptacyjne skalowanie tablicy kwantyzacji za pomocą współczynnika, który może być podawany indywidualnie dla obrazu lub makrobloku W kodowaniu międzyobrazowym wykorzystywane są obrazy typu P i B. Kwantyzator wykorzystuje osobne tablice kwantyzacji dla tego trybu. Kodowany jest błąd predykcji, który charakteryzuje się małą wartością składowej stałej oraz dość równomierną funkcją gęstości widmowej energii.

Regulacja kroku kwantyzatora pozwala na sterowanie kodekiem, który może pracować w trybie stałego strumienia binarnego CBR lub zmiennego

strumienia binarnego VBR. W pierwszym trybie prawie każdy obraz jest kodowany z użyciem prawie tej samej liczby bitów. W celu zapewnienia stałego strumienia najczęściej przydziela się stałą liczbę bitów kolejnym grupom obrazów.

Obrazy typu B mają najlepszy stosunek jakości do kompresji, dlatego obrazy tego typu stanowią zazwyczaj około 2/3 wszystkich obrazów.

Nie bez znaczenia jest kolejność, w jakiej przesyłane są zakodowane obrazy, gdyż, aby zdekodować obraz typu B potrzebna jest znajomość obrazów P lub I i P, między którymi obraz ten występuje. Przykładowa kolejność kodowania, przesyłania i dekodowania przedstawiona jest poniżej:

- kolejność obrazów na wejściu kodera

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

- kolejność obrazów na wyjściu kodera, w strumieniu danych i na wejściu dekodera

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

- kolejność obrazów na wyjściu dekodera

1	2	3	4	5	6	7	8	9	10	11	12	13

Na wyjściu kodera ostateczne dane są kodowane jak poprzednio z wykorzystaniem algorytmu Huffman'a. Wykorzystuje się tu trzy warianty kodowania:

- statyczny – tabela częstotliwości znaków ustalana jest z góry, niezależnie od tego, jakie dane są kodowane;

- dynamiczny – w celu wypełnienia tabeli częstotliwościami, dokładnie dla tych danych, które są kodowane, przed rozpoczęciem kodowania dokonywana jest analiza częstotliwości znaków w kodowanych danych;
- adaptacyjny – wstępne założenia o częstotliwości występowania znaków zostają w trakcie kodowania dostosowane do częstotliwości rzeczywiście występujących.

2.3.2.2. MPEG-4

Standard MPEG-4 jest opracowywany od 1994 roku głównie ze względu na zastosowania kompresji w trzech obszarach:

- telekomunikacja audiowizualna: np. wideotelefonia, systemy wideokonferencyjne, teleedukacja, telekonsultacje;
- interaktywny dostęp do audiowizualnych baz danych: np. instrukcje obsługi;
- zdalne dozorowanie i sterowanie: np. kontrola bezpieczeństwa danego terminala.

Podczas opracowywania standardu uwzględniono między innymi następujące aspekty:

- dostępność danych systemu kompresji w środowisku sprzyjającym generowaniu błędów;
- interaktywny dostęp do danych wizualnych oraz efektywne metody modyfikacji takich danych;
- kodowanie danych mieszanych – reprezentujących zarówno obrazy scen naturalnych, jak i obrazy sztucznie wytworzone metodami grafiki komputerowej;
- znaczną efektywność kompresji – krytyczna dla przepływności poniżej 64 [kbit/s];

- ulepszony swobodny dostęp do całej sekwencji wizualnej, na przykład dostęp do obrazów.

W roku 1998 zakończono przygotowywanie pierwszej wersji standardu. Nadal trwają prace mające na celu opracowanie drugiej wersji, która ma między innymi obejmować kodowanie animowanych obiektów trójwymiarowych.

W standardzie MPEG-4 koduje się obiekty wizyjne VO, które są odwzorowaniami rzeczywistych obiektów sceny. Automatyczna segmentacja sceny na obiekty wyróżnione ze względu na ich znaczenie w rzeczywistym świecie nadal jest otwartym problemem. W szczególnym przypadku cała sekwencja może być jednym obrazem wizyjnym. Wtedy kodowanie sekwencji odbywa się podobnie jak w poprzednio omówionych standardach. Obiekty wizyjne składają się z płaszczyzn obiektów wizyjnych VOP, które są reprezentacjami danego obiektu wizyjnego w kolejnych obrazach sekwencji. Płaszczyzny obiektów wizyjnych mogą mieć dowolny kształt i są kodowane zarówno w trybie wewnątrzobrazowym I-VOP, jak i w trybie międzyobrazowym P-VOP, B-VOP. Kompresja obrazów wizyjnych wymaga kodowania kształtu oraz tekstury.

Kodowanie kształtu obiektu

Kodowanie kształtu odbywa się w obrębie poszczególnych makrobloków przeciętych brzegiem obiektu wizyjnego. Koder przyjęty w standardzie MPEG-4 wykorzystuje obraz binarny, w którym punkty należące do obiektu mają wartość 1, a punktom tła przypisano wartość 0. W celu poprawy efektywności kompresji odpowiadające makroblokom obrazy binarne o rozmiarze 16x16 pikseli przed właściwym kodowaniem mogą być decymowane do rozmiarów 8x8 lub 4x4 piksele.

Kodowanie odbywa się w dwóch trybach:

- intra – makroblok każdej płaszczyzny jest kodowany niezależnie;
- inter – dla kolejnych płaszczyzn obiektu wizyjnego koduje się błąd predykcji z kompensacją ruchu.

Kodowanie tekstur obiektów.

W najprostszym przypadku obiekty wizyjne są prostokątne i nawet cała sekwencja może być jednym obiektem wizyjnym. W zasadzie wykorzystuje się algorytm kompresji znany z zalecenia ITU-T H.263, w którym wprowadzono jednak wiele ulepszeń zwiększających efektywność kompresji obrazów:

- dopuszczona jest analiza między liniowa (przeplot linii) kodowanego sygnału wizyjnego;
- kodowane są całe obrazy lub pojedyncze pola;
- możliwość estymacji i kompensacji ruchu w blokach o rozmiarze 8x8 pikseli;
- predykcja uwzględniająca wartości wektorów ruchu wyznaczone dla sąsiednich bloków;
- współczynniki transformaty kosinusowej są kwantowane według jednej z dwóch technik: metodą ujętą w zaleceniu ITU-T H.263 lub metodą znaną ze standardu MPEG-2;
- współczynniki DC bloków I są kodowane predykcyjnie;
- pozostałe skwantowane współczynniki również można kodować predykcyjnie (włączanie i wyłączanie trybu kodowania predykcyjnego współczynników może być dokonywane tylko dla całych makrobloków);
- wartości błędów predykcji współczynników DC w blokach typu I są kodowane z różnymi tablicami kodu dla luminancji i chrominancji.

Część płaszczyzn obiektów wizyjnych jest kodowana jako płaszczyzny interpolowane B-VOP. Makrobloki w takich płaszczyznach mogą być

kodowane na dwa sposoby, które zostały zdefiniowane dla kodowania makrobloków typu B w poprzednio omawianych standardach:

- kodowanie makrobloków typu B zgodnie z zaleceniem ITU-T H.263;
- kodowanie obrazów typu B zgodnie ze standardem MPEG-2.

Jeżeli obiekt ma nieregularne kształty, to bloki mogą być podzielone na trzy grupy:

- mieszczące się całkowicie wewnątrz obiektu wizyjnego;
- mieszczące się częściowo wewnątrz obiektu wizyjnego;
- pozostające poza obiektem wizyjnym.

Bloki pozostające wewnątrz obiektu są kodowane w sposób opisany wyżej. Bloki znajdujące się poza obiektem oczywiście nie są wcale kodowane. Natomiast bloki przecięte brzegiem obiektu są kodowane tylko częściowo z użyciem tak zwanej transformaty kosinusowej dopasowanej do kształtu.

Wersja 2 standardu MPEG-4 ma zostać uzupełniona o następujące elementy:

- poprawione kodowanie skalowalne;
- ulepszone kodowanie w celu transmisji w kanałach o bardzo małej przepływności;
- poprawiona odporność na błędy transmisji;
- animacja obiektów trójwymiarowych;
- kodowanie trójwymiarowych siatek.

3. Metody oceny jakości połączeń do przesyłania komunikatów multimedialnych

Potrzeba pomiarów jakości połączeń między poszczególnymi aplikacjami symulatora stała się koniecznością ze względu na:

- relatywnie duży ruch w sieci;

- przesyłanie bez opóźnień dużej ilości danych.

W połączeniach między aplikacjami, gdzie mamy do czynienia z przekazem kodów sterujących w celu wywołania określonego dźwięku lub obrazu, bardzo ważnym zagadnieniem jest jakość odbieranych informacji. O tej jakości mogą świadczyć następujące subiektywne wrażenia:

- synchronizacja sygnału dźwiękowego z obrazem;

- naturalność brzmienia głosu;

- czytelność odtworzonego efektu dźwiękowego;

- łatwość interpretacji wywołanego efektu;

- zachowanie wyrazistości podkładów mapowych;

- zrozumiałość przekazywanej informacji;

- natężenie uwagi w czasie trwania połączenia;

- uwzględnianie hałasu otoczenia i nakładających się dźwięków z kilku interfejsów dedykowanych jednocześnie.

Metody oceny jakości połączeń w konstruowanym symulatorze podzielono na dwie podstawowe kategorie: oceny subiektywne (z udziałem użytkowników) i oceny obiektywne (realizowane przy użyciu urządzeń). Uważa się, że oceny jakości dźwięku i obrazu najlepiej może dokonać człowiek, bowiem to on jest odbiorcą danego przekazu.

Jednak warto by w przyszłości zrealizować testy jakości połączeń metodami obiektywnymi, które byłyby powtarzalne, możliwe do przeprowadzenia łatwo, szybko i najlepiej sprzętem przenośnym. Dawałoby to administratorowi symulatora (obsłudze technicznej) możliwość optymalnego rozmieszczeni poszczególnych stanowisk dedykowanych.

Można by w ten sposób eliminować efekty zakłócania się nawzajem terminali, na których poprzez interfejsy dedykowane uruchamiane by były efekty dźwiękowe.

W dalszej części tego rozdziału zostaną przedstawione metody subiektywne oceny jakości dźwięku i wizualizacji sytuacji operacyjno-taktycznej na ekranach monitorów oraz metody obiektywne oparte o parametry odpowiadające odczuciom człowieka. Testy przeprowadzono w Dowództwie 2 Brygady Lotnictwa Taktycznego.

3.1. Badania subiektywne

Testy subiektywne zostały przeprowadzone w określonych, kontrolowanych warunkach. Nie wynajęto odpowiedniego laboratorium badawczego, gdzie była by możliwość wyciszenia do max wszystkich dźwięków zewnętrznych. Jednak pomieszczenia do testów wybrano celowo, gdzie wyciszenie pomieszczenia uzyskano poprzez zamknięcie wszystkich okien, na ścianach są płytki dźwiękochłonne, pomieszczenie jest usytuowane w miejscu, gdzie jest minimalne natężenie ruchu i nie dochodzą z zewnątrz żadne hałasy.

Celowo nie chciano zapewniać warunków izolacji dźwiękowej, gdyż istota tego pomiaru było zapewnienia warunków testów, z jakimi w rzeczywistości spotkają się użytkownicy. Warunki laboratoryjne zbyt wyidealizowane byłyby sztucznie stworzone i pomiar jakości byłby obarczony błędem. Zatem przyjęto, iż tylko pomiar w warunkach jak najbardziej zbliżonych do rzeczywistości będzie źródłem uzyskania wiarygodnych wyników.

W badaniach subiektywnych napotyka się na kilka podstawowych trudności: opinie o jakości połączeń, a następnie efektów czasami znacznie się różnią w zależności od osób oceniających, zależą od czasu, okoliczności, a nawet od kondycji psychofizycznej oceniających. Najbardziej korzystną

sytuacją z punktu widzenia wiarygodności wyników byłoby ankietyzowanie grupy użytkowników symulatora tuż po przeprowadzonym pokazie. Być może do tej koncepcji należy wrócić podczas realizacji zadania 11 i 13 z harmonogramu.

Zdecydowanie bardziej profesjonalna metodą są testy subiektywne przeprowadzone w laboratorium. Wymagają one najczęściej mniejszej liczby uczestników i są mniej czasochłonne. Badania laboratoryjne mają jeszcze dodatkową zaletę, że rezultaty testów subiektywnych mogą być porównywane z uzyskanymi w innych laboratoriach badawczych. Jednak ta opcja nie będzie realizowana. Podstawową przyczyną rezygnacji z tych testów jest aspekt finansowy. Wynajęcie odpowiedniego laboratorium do tego typu badań generowałaby dodatkowe koszty.

3.1.1. Metody subiektywne oceny jakości dźwięku

3.1.1.1. Metoda uśrednionej opinii oceniających

Miarą jakości transmisji efektów dźwiękowych, komunikatów dźwiękowych w całym łańcuchu transmisji jest uśredniona opinia oceniających MOS. Procedury przeprowadzenia badań subiektywnych oraz kryteria oceny są opisane w zaleceniach:

- Methods for subjective determination of transmission quality P.800 ITU-T;
- Method for evaluation of service from the standpoint of speech transmission quality P.82 ITU-T;
- Subjective performance assessment of telephone-band and wideband digital codecs P.830 ITU-T.

Literatura rozróżnia się dwa rodzaje testów subiektywnych, których rezultatem jest uśredniona opinia oceniających jakość realizowanego połączenia, przy przesyłaniu danych multimedialnych między komputerami. Są to:

- testy konwersacyjne (połączenie w obie strony);
- testy odsłuchowe (połączenie w jedną stronę).

Subiektywne testy konwersacyjne badanego połączenia polegają na przekazie komunikatu multimedialnego między uczestnikami testu, po czym oceniają dane łącze w skali od 1 do 5.

Uczestnicy testu powinni znajdować się przy stanowisku komputerowym, na którym jest zainstalowany dany interfejs dedykowany. Poziom hałasu wewnątrz nie powinien przekraczać 35 dB.
Uczestnicy eksperymentu reprezentują zwykłą grupę użytkowników symulatora.

Zakres warunków testu powinien być wystarczająco szeroki, aby użytkownicy zapoznali z transmisją dźwięku, który od bardzo źle przygotowanych próbek dźwiękowych do perfekcyjnie czytelnych. W czasie sesji treningowej użytkownicy symulatora powinni być bardzo dokładnie zapoznani z przebiegiem eksperymentu i ze sposobem określania swojej opinii o jakości odbieranych komunikatów dźwiękowych.

Subiektywne testy odsłuchowe polegają na słuchaniu komunikatu za pośrednictwem badanego połączenia. Po wysłuchaniu materiału testowego uczestnicy dokonują oceny jakości komunikatów w skali ocen od 1 do 5.

W testach dla zapewnienia wystarczającej dokładności pomiaru powinno brać udział 6 oceniających. ITU-T P.800 zaleca, aby w trakcie sesji treningowej liczba warunków połączenia wynosiła nie więcej niż 10. Warunki akustyczne pomieszczenia gdzie zainstalowano komputery z interfejsami dedykowanymi powinny być zbliżone do tych, jakie są wymagane w testach konwersacyjnych. Dopuszcza się udział większej liczby osób równocześnie słuchających i oceniających. Wówczas wymiary pomieszczenia muszą być odpowiednio większe z zachowaniem wymaganych parametrów akustycznych. Odsłuchujący nie powinni między sobą dzielić się opiniami o teście.

Materiał testowy stanowi pięć krótkich komunikatów oznajmiających w danym języku (czas trwania około 20 s). Nagrywane są głosy męskie (z dobrą dykcją) poprzez dobrej kasy mikrofon, kartę Sond Blaster Live. Wydaje się, że tej klasy karta muzyczna stanowi wystarczająco wysokiej klasy cyfrowe urządzeniach z opcją do rejestrowania. Plik z nagraniem źródłowym, materiałem testowym jest następnie odtwarzana zgodnie z procedurą eksperymentu.

Uczestnicy testu powinni być bardzo dokładnie zapoznani z przebiegiem eksperymentu i ze sposobem określania swojej opinii o jakości odbieranego komunikatu dźwiękowego.

Wadą testów odsłuchowych jest to, że ze swej natury nie nadają się do oceny opóźnień, czy zakłóceń, które oddziałują na komunikat dźwiękowy w trakcie połączenia.

Zarówno w testach konwersacyjnych jak i odsłuchowych uczestnicy mają do dyspozycji te same kryteria oceny. Rozróżnia się następujące kryteria oceny jakości komunikatów dźwiękowych:

- bezwzględne kategorie oceny

A. Doskonała

B. Dobra

C. Dość słaba

D. Słaba

E. Zła

- Kategoria natężenia słuchu

A. Całkowity relaks

B. Niewielka uwaga

C. Umiarkowana uwaga

D. Znaczny wysiłek słuchowy

E. Brak zrozumienia, duża uwaga

- Kategoria głośności

A. Znacznie głośniej niż potrzeba

B. Głośniej niż potrzeba

C. Głośność taka jak potrzeba

D. Ciszej niż trzeba

E. Znacznie ciszej niż trzeba

W ramach każdej kategorii ocenom A, B, C, D, E są przyporządkowane wagi liczbowe odpowiednio 5, 4, 3, 2, 1. W każdym eksperymencie, w danej kategorii, suma ocen wszystkich uczestników testu podzielona przez liczbę osób stanowi wartość MOS.

3.1.1.2. Metoda badania wyrazistości logatomowej

Badanie wyrazistości logatomowej wykonuje się według procedury i w warunkach środowiskowych podanych w projekcie normy PN-V-90002 (dla łączy cyfrowych). Polega ono na określeniu procentu odebranych prawidłowo bezsensownych sylab (jedno- lub wielosegmentowych) w stosunku do ogólnej liczby nadawanych logatomów.

Badania wyrazistości logatomowej przeprowadza się w warunkach rzeczywistych lub w warunkach sztucznie stworzonych w laboratorium.
Pomiary wykonywane są w dwóch odizolowanych akustycznie pomieszczeniach, w których poziom tła nie przekracza 40 [dB]. W teście powinna wziąć udział grupa przeszkolonych użytkowników (minimum 5 osób) o prawidłowym słuchu.

Odebrane logatomy są notowane w specjalnym arkuszu pomiarowym i sprawdzane przez eksperta. Następnie obliczana jest średnia wartość wyrazistości logatomowej W_L według wzoru:

$$W_L = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} W_{n,k} \qquad (3.1)$$

gdzie:

N – liczba oceniających,

K – liczba list logatomów,

$W_{n,k}$ – wyrazistość logatomowa dla k-tej listy i n-tego oceniającego, tj. wyrażony w procentach stosunek liczby prawidłowo odebranych logatomów do liczby logatomów odczytanych[2].

3.1.2. Metody subiektywne

Metody subiektywne oceny jakości komunikatów wizualnych są szczegółowo opisane w zaleceniach organizacji międzynarodowej ITU i są to:

- Methodology for subjective assessnemt of the quality of television pictures BT.500-7 ITU-R;
- Subjective video quality assessment methods for multimedia application P.910 ITU-T.

W dokumentach tych zawarte są wymagania i wskazówki dotyczące przeprowadzenia testów, wyposażenia stanowiska badawczego, przebiegu eksperymentu, doboru sekwencji testowych, obróbki statystycznej wyników i stosowanych skal ocen (numerycznych, ciągłych, opisowych). Normalizacja metod subiektywnych pozwala na porównanie wyników badań przeprowadzonych w różnych miejscach, gdzie realizowano testy.

Przygotowanie i przeprowadzenie testów subiektywnych jest bardzo pracochłonne, między innymi ze względu na konieczność dokładnego przygotowania stanowiska laboratoryjnego. Musi ono spełniać szereg wymagań na np.:

- jasność i kontrast monitora,
- stosunek luminancji nieaktywnego monitora do wartości szczytowej luminancji,

---

1. Trzaskowska J.: Subiektywna ocena jakości mowy w sieciach telekomunikacyjnych, Krajowe Sympozjum Telekomunikacji 2000.

- stosunek luminancji monitora przy wyświetlaniu obrazu czarnego do wartości szczytowej luminancji, w kompletnie ciemnym pomieszczeniu,
- stosunek luminancji tła za monitorem do wartości szczytowej luminancji,
- chrominancja tła za monitorem,
- oświetlenie pomieszczenia,
- maksymalny kąt obserwacji, mierzony do prostopadłej do monitora.

Wyniki oceny subiektywnej jakości komunikatów wizualnych często zależą od zakresu jakości materiału testowanego, doświadczenia i oczekiwań oceniających. Aby kontrolować te efekty wizualne, kiedy jest to tylko możliwe, sesje testowe powinny zawierać odpowiednie sekwencje „kotwiczące". Powinny być one prezentowane w sposób losowy w czasie sesji. Oceniający nie wiedząc o występowaniu tych sekwencji będą oceniać je razem z sekwencjami testowanymi. Sekwencje „kotwiczące" spełniają dwa cele: po pierwsze rozszerzają i/lub uzupełniają zakres jakości prezentowanego materiału, po drugie pozwalają na ocenę zachowania oceniających pomiędzy testami[3].

3.1.2.1. Metoda dwubodźcowa z ciągłą skalą jakości (DSCQS)

DSCQS jest metodą, w której każda próba testowa składa się z pary bodźców: jeden bodziec jest odniesieniem, a drugi testowanym. Bodziec testowany jest to obraz uzyskany w procesie dekompresji, lub poddany innym przekształceniom. Dwa bodźce są prezentowane dwukrotnie w czasie próby, w sposób zmienny, wybierany losowo w każdej próbie. Do pomocy oceniającym można wykorzystać sygnały dźwiękowe, sygnalizujące, kiedy

---

2. Klink J.: *Systemy komunikacji multimedialnych*, Krajowe Sympozjum Telekomunikacji 1999

zaczyna się próba, kiedy bodziec, kiedy oceniać i jaki jest numer aktualnie przebiegającej próby w sekwencji składającej się na sesję testową.

Oceniający nie są informowani o porządku testu, oceniają każdy prezentowany bodziec przez zaznaczenie oceny na ciągłej skali oceny jakości. Tak, więc przy każdej próbie testowej, w metodzie DSCQS wystawiane są dwie oceny: jedna dla sekwencji odniesienia i druga dla sekwencji testowanej. Czasami w testach dwa prezentowane bodźce są bodźcami odniesienia. Próby takie są stosowane, aby zbadać niekonsekwencję zachowania oceniających.

Metoda DSCQS jest zwykle stosowana do oceny jakości wizualizaji, w których różnice jakościowe między sekwencjami odniesienia i testowanymi nie są zbyt duże. Dlatego, też metoda ta wykorzystywana jest do oceny jakości obrazów przesyłanych z dużymi przepływnościami, co najmniej 512 [kbit/s][4].

3.1.2.2. Metoda dwubodźcowa ze skalą pogorszeń jakości (DSIS)

Sposób wykonania badań metodą oceny jakości obrazu DSIS, jest bardzo podobny do badań metodą DSCQS. Tutaj również każda próba składa się z pary bodźców: odniesienia i testowanego. Jednak w metodzie DSIS bodźce te są zawsze prezentowane w tej samej kolejności, tzn. bodziec odniesienia jest zawsze pierwszy, po nim następuje bodziec testowany.

W metodzie DSIS oceniający porównują dwa bodźce i oceniają stopień zniekształceń w bodźcu testowanym w porównaniu do bodźca odniesienia, używając pięciostopniowej skali ocen. Tak, więc wystawiana jest tylko jedna ocena dla każdej próby. Skala ocen nie jest ciągła, jak to miało miejsce w metodzie DSCQS.

---

3. Wolf S.: Measuring the End-to-End Performance of Digital Video Systems, Institute for Telecommunication Sciencies

Metoda DSIS jest zwykle używana do oceny wpływu zniekształceń obrazu na niezadowolenie użytkownika, dlatego też stosowana jest głównie do oceny funkcjonowania systemów, w których występują wyraźne zniekształcenia, wynikające np. ze zbyt małej przepływności. Metodę tą zwykle wykorzystuje się do badania systemów o średniej przepływności, od 100 do 350 [kbit/s].

Ze względu na to, że ocena jest wystawiana na podstawie porównania bodźca testowanego z bodźcem odniesienia, ważne jest, aby różnice poziomów jakości obu bodźców nie były zbyt duże. W wypadku, gdy bodźce różnią się znacząco zalecane jest obniżenie jakości bodźca odniesienia[5].

3.1.2.3. Metoda jednobodźcowa (SS)

Metoda oceny jakości obrazu SS jest metodą, w której w trakcie prezentacji sekwencji testowej oceniającym nie pokazuje się obrazów odniesienia, a jedynie obrazy testowane. Ze względu na tą podstawową cechę metody jednobodźcowej, jest ona chętnie wykorzystywana w skromniej wyposażonych ośrodkach badawczych, gdzie są trudności z uzyskaniem bodźca odniesienia. Testujący ocenia każdy bodziec, używając pięciostopniowej skali ocen. Metoda ta wykorzystywana zazwyczaj dla małych przepływności, może stwarzać pewne kłopoty związane z możliwością koncentracji ocen w dolnej części skali ocen, co utrudnia analizę wyników. Dlatego też dopuszcza się rozszerzenie skali ocen do 11 stopni Dodatkowo przeprowadza się szkolenie wstępne oceniających, podczas którego mają możliwość obejrzenia próbek testowych o jakości z pełnego zakresu spodziewanego w teście. Tego typu działania zmniejszają ryzyko koncentracji ocen w dolnej części skali ocen. Dodatkowo w metodzie tej ze względu na to, że nie są prezentowane bodźce odniesienia zauważalny jest

---

4. Wolf S.: Measuring the End-to-End Performance of Digital Video Systems, Institute for Telecommunication Sciencies

większy, niż w innych metodach, wpływ poprzednio oglądanych bodźców testowych na ocenę aktualnie oglądanego. Aby skompensować ten efekt zaleca się dwukrotne przeprowadzenie tego samego testu ze zmienioną kolejnością prezentowanych bodźców[6].

3.1.2.4. Metoda dwubodźcowa ze skalą dwuwartościową (DSBV)

Metoda DSBV jest nową, niestandaryzowaną metodą testów subiektywnych, opracowaną przez ekspertów testujących wydajność kodeków MPEG. Wykorzystywano tą metodę do oceny efektywności pracy kodeków w środowisku, w którym występują paczki błędów na tyle długie, że mogłyby prowadzić do utraty synchronizacji. W tych warunkach oczekiwano bardzo słabej jakości odtwarzanego obrazu. Dlatego pytanie oceniających o ocenę jakości lub zniekształceń uważano za nieefektywne. W zamian oceniający określali czy dany kodek zdolny był do odzyskania sekwencji obrazu ze strumienia bitów z paczkami błędów.

Sekwencja prezentowana w tej metodzie jest taka sama jak w metodzie DSIS, z tą różnicą, że sekwencja dekodowana w warunkach nie występowania błędów była prezentowana jako bodziec odniesienia dla bodźca testowanego dekodowanego ze strumienia bitów zawierającego błędy. Oceniający porównując obie sekwencje i oceniając „tak" lub „nie" określają czy testowany bodziec został odtworzony poprawnie[7].

3.2. Badania obiektywne

Powyżej przedstawione zostały subiektywne metody oceny jakości połączeń dla transmisji danych multimedialnych, jak się okazuje wiarygodne przeprowadzenie testów subiektywnych uwarunkowane jest przez szereg

---

5. Burakowski W.: *Sieci IP z jakością obsługi*, Krajowe Sympozjum Telekomunikacji 2000
6. Sochan A.: *Laboratoryjne metody badania sieci ATM*, Zeszyty naukowe Politechniki Śląskiej 1999.

czynników np. odpowiednio dobrany materiał testowy czy oczekiwania oceniających. Trudno jest tu o zapewnienie powtarzalności wyników. Dlatego też rozpoczęto prace nad obiektywnymi metodami oceny jakości dźwięku i obrazu. Stworzono szereg parametrów, które odpowiadają odczuciom człowieka.

3.2.1. Metody obiektywne oceny jakości mowy

Głośność, zrozumiałość i wyrazistość są podstawą metod obiektywnych oceny jakości dźwięku. Parametry te nie są od siebie niezależne. Jakość komunikacji jest niedostateczna zarówno wówczas, gdy dźwięki są zbyt ciche lub zbyt głośne, chociaż przesyłane w pełnym paśmie, jak i wówczas, gdy głośność jest właściwa, lecz pasmo przesyłowe jest zbyt wąskie[8].

Poniżej przedstawione zostaną: metoda wskaźnika wyrazistości zalecana przez ANSI, metoda głośnościowa zalecana przez ITU-T i Polską Normę oraz metoda PSQM zalecana, przez ITU-T.

3.2.1.1. Metoda wskaźnika wyrazistości

Idea wprowadzenia wskaźnika wyrazistości AI opiera się na procentowo różnym udziale obszarów natężeń i częstotliwości dźwięku. Wskaźnik wyrazistości jest addytywną, nieujemną miarą, określoną na pełnym paśmie częstotliwości słyszalnych. Wskaźnik wyrazistości danego pasma zależy od wyrazistości tego pasma, a nie od jego położenia. Miara AI traktowana jako wskaźnik oceny jakości transmisji mowy posiada przewagę nad wyrazistością W, gdyż jest addytywną funkcją przesyłanego pasma częstotliwości.

W ogólnym przypadku, jeżeli widmo mowy zostanie podzielone na 20 podzakresów, tj. pasm elementarnych o jednakowej wyrazistości cząstkowej,

---

7. Brachmański S.: *Obiektywne metody oceny jakości transmisji mowy*, Krajowe Sympozjum Telekomunikacji 2000

czynników np. odpowiednio dobrany materiał testowy czy oczekiwania oceniających. Trudno jest tu o zapewnienie powtarzalności wyników. Dlatego też rozpoczęto prace nad obiektywnymi metodami oceny jakości dźwięku i obrazu. Stworzono szereg parametrów, które odpowiadają odczuciom człowieka.

3.2.1. Metody obiektywne oceny jakości mowy

Głośność, zrozumiałość i wyrazistość są podstawą metod obiektywnych oceny jakości dźwięku. Parametry te nie są od siebie niezależne. Jakość komunikacji jest niedostateczna zarówno wówczas, gdy dźwięki są zbyt ciche lub zbyt głośne, chociaż przesyłane w pełnym paśmie, jak i wówczas, gdy głośność jest właściwa, lecz pasmo przesyłowe jest zbyt wąskie[8].

Poniżej przedstawione zostaną: metoda wskaźnika wyrazistości zalecana przez ANSI, metoda głośnościowa zalecana przez ITU-T i Polską Normę oraz metoda PSQM zalecana, przez ITU-T.

3.2.1.1. Metoda wskaźnika wyrazistości

Idea wprowadzenia wskaźnika wyrazistości AI opiera się na procentowo różnym udziale obszarów natężeń i częstotliwości dźwięku. Wskaźnik wyrazistości jest addytywną, nieujemną miarą, określoną na pełnym paśmie częstotliwości słyszalnych. Wskaźnik wyrazistości danego pasma zależy od wyrazistości tego pasma, a nie od jego położenia. Miara AI traktowana jako wskaźnik oceny jakości transmisji mowy posiada przewagę nad wyrazistością W, gdyż jest addytywną funkcją przesyłanego pasma częstotliwości.

W ogólnym przypadku, jeżeli widmo mowy zostanie podzielone na 20 podzakresów, tj. pasm elementarnych o jednakowej wyrazistości cząstkowej,

---

7. Brachmański S.: *Obiektywne metody oceny jakości transmisji mowy*, Krajowe Sympozjum Telekomunikacji 2000

wskaźnik wyrazistości AI pełnego pasma częstotliwości wyraża się zależnością:

$$AI = 0.05\sum_{i=1}^{20}\gamma_i \qquad (3.2)$$

gdzie γ_i jest wskaźnikiem wyrazistości w i-tym paśmie elementarnym[9].

3.2.1.2. Metoda głośnościowa

Punktem wyjścia w metodzie oceny jakości mowy według kryterium głośności były subiektywne pomiary, w których porównuje się głośność wzorca i badanego sygnału. Otrzymywana ocena nie jest wielkością bezwzględną, lecz otrzymaną przez porównanie ze wzorcem nie znanym użytkownikowi systemu i tym samym nic mu nie mówiąca. Kierując się tą przesłanką oraz mając na uwadze wady metod subiektywnych opracowano metodę obiektywną nie opartą na wzorcu. W metodzie tej właściwości transmisyjne są określane wskaźnikami głośnościowymi LR według zależności:

$$LR = -\left(\frac{10}{m}\right)\log_{10}\sum_{i=1}^{N}10^{-0.1m(W_i+L_i)} \qquad (3.3)$$

gdzie:

m – współczynnik skalujący,

N – liczba pasm częstotliwości,

W_i – współczynnik ważkości i-tego pasma częstotliwości,

L_i – tłumienność drogi od punktu odniesienia ust do punktu odniesienia ucha badanego połączenia w i-tym pasmie częstotliwości.

Współczynniki W_i wynika normatywnych testów psychoakustycznych i ich wartości są podane w zaleceniu P.79 ITU-T. Sygnał testowy generowany

8. Op. cit.

jest sygnałem akustycznym o płynnie zmieniającej się częstotliwości od 200 [Hz] do 4 [kHz][10].

3.2.1.3. Metoda PSQM

Metoda percepcyjnego pomiaru jakości mowy PSQM zalecana jest przez ITU-T, jako obiektywna metoda oceny kodeków mowy, mająca symulować subiektywną percepcję dźwięków w warunkach rzeczywistych. W metodzie tej symulowane są takie cechy percepcyjnego przetwarzania dźwięków przez ucho ludzkie jak:

- wypaczenie skali częstotliwości;
- zmiana czułości ucha wraz ze zmianą częstotliwości;
- różnica między poziomem głośności, a skalą subiektywnego odczucia głośności.

Zasadnicza nielinearność modelu słuchowego zawiera w sobie to, że konieczne jest oddzielne przetwarzanie każdego sygnału mowy, który ucho w danym momencie słyszy. System słuchowy człowieka jest znany ze słabszej dyskryminacji w zakresie wysokich częstotliwości niż w zakresie częstotliwości niskich. To, wraz ze zjawiskiem maskowania przez szum, doprowadziło do zamodelowania analizy dokonywanej przez ucho ludzkie za pomocą filtrów barkowych. Model ten wymaga przetwarzania sygnałów przez grupę takich filtrów, z odległościami częstotliwości środkowych i szerokościami pasm rosnącymi wraz z częstotliwością. Filtry te mogą być traktowane jako krzywe przestrajania nerwów słuchowych. Ciągłe widmo uzyskane w wyniku tej operacji zostało nazwane rozkładem pobudzeń, gdyż odpowiada ono rozkładowi bodźców w nerwach słuchowych.

Modyfikacje widmowe przedstawione dotychczas wynikały z charakterystyki częstotliwościowej ucha i nieliniowego efektu wygładzania

---

9. Brachmański S.: *Obiektywne metody oceny jakości transmisji mowy*, Krajowe Sympozjum Telekomunikacji 2000

wprowadzanego filtracją w pasmach krytycznych przez ślimak błędnika ucha. Teraz należy uwzględnić fakt, że ucho nie jest jednakowo czułe na bodźce o różnych częstotliwościach. W tym celu należy przeprowadzić zmianę poziomów natężenia wyrażonych w decybelach na poziomy głośności wyrażone w fonach. W ten sposób otrzymuje się widmo skorygowane względem głośności. Ostatnim krokiem jest uwzględnienie faktu, że wzrost głośności w fonach potrzebny do podwojenia subiektywnego odczucia głośności nie jest stały, lecz zmienia się z poziomem głośności. Należy, więc przejść ze skali fonowej na skalę sonową.

Zarówno oryginalny sygnał mowy jak i sygnał badany są osobno przetwarzane w identyczny sposób, prowadząc do uzyskania tzw. widm barkowych odpowiednio sygnału oryginalnego i badanego. Miarą jakości jest wówczas odpowiednio zdefiniowana odległość między tymi widmami zwana wskaźnikiem PSQM[11].

3.2.2. Metody obiektywne oceny jakości obrazu

Badanie jakości obrazu tymi metodami wymaga modelowania złożonych i nie w pełni poznanych procesów widzenia, jakie zachodzą w ludzkim systemie widzenia.

Metody obiektywnej oceny jakości obrazu zazwyczaj obejmują między innymi:

- wstępne przetwarzanie – statystyczne transformacje nieliniowe, transformacje barw itd.;
- analizę przestrzenną lub czasowo-przestrzenną;
- modelowanie procesów percepcji kontrastu i maskowania.

---

[11] Klink J.: *Systemy komunikacji multimedialnych*, Krajowe Sympozjum Telekomunikacji 1999

Wartość wskaźnika jakości może być wyznaczona w różny sposób, na przykład przez:

- sumowanie zakłóceń z poszczególnych pasm czasowo-przestrzennych;
- sumowanie zniekształceń wyznaczonych dla obiektów uprzednio uzyskanych w drodze segmentacji obrazu;
- sumowanie zakłóceń różnych typów;
- modelowanie postrzegania różnic między obrazami[12].

Poniżej przedstawiona zostanie przykładowa metoda obiektywnej oceny jakości obrazu, opracowana w Institute for Telecommunication Sciences (ITS). Jest to metoda oparta o parametry, które obliczane są na podstawie dwóch cech: jedna pochodzi z obrazu po przejściu przez filtr Sobel, a druga z obrazu różnicy ruchu. Filtracja Sobel jest metodą, która umożliwia uwydatnienie krawędzi w obrazie. Po filtracji otrzymuje się czarno-biały obraz, który zawiera czarne obszary tam gdzie obraz badany był jednolity, a białe gdzie w obrazie badanym znajdowały się gwałtowne zmiany jasności pikseli. Parametry oparte o filtrację Sobel okazują się być użyteczne do obliczeń zniekształceń przestrzennych, jak np. zamazanie obrazu, natomiast parametry oparte o obraz różnicy ruchu są użyteczne do obliczeń zniekształceń czasowych, jak np. klatkowanie obrazu.

3.2.2.1. Pomiar cech źródła i przeznaczenia

Pomiar informacji przestrzennej źródła wideo (SI_S) i informacji przestrzennej przeznaczenia wideo (SI_D). SI_S i SI_D są cechami, których sposób pomiaru dla źródła i przeznaczenia jest taki sam. Cechy te charakteryzują szczegóły przestrzeni sceny na ekranie monitora, dlatego też

---

10. Domański M.: *Zaawansowane techniki kompresji obrazów i sekwencji wizyjnych*, Wydawnictwo Politechniki Poznańskiej, Poznań 2000

parametry obiektywne jakości obrazu mogą być obliczone przy użyciu tych cech.

Metoda pomiaru:

1. Sygnały źródła i przeznaczenia są próbkowane i wartości $Y_S(t_n)$ i $Y_D(t_m)$ reprezentują odpowiednio luminancję źródła i przeznaczenia w czasie t_n i t_m.
2. Wartości $Y_S(t_n)$ i $Y_D(t_m)$ są następnie filtrowane dwoma maskami 3x3, operacja filtracji detekuje krawędzie horyzontalne $H*Y_S(t_n)$ i $H*Y_D(t_m)$ oraz krawędzie wertykalne $V*Y_S(t_n)$ i $V*Y_D(t_m)$. Następnie tworzone są obrazy luminancji po filtracji Sobel, jako kombinacje sygnałów wyjściowych dwóch operacji filtracji:

$$Sobel(Y_S(t_n)) = \sqrt{(H * Y_S(t_n))^2 + (V * Y_S(t_n))^2} \tag{3.4}$$

$$Sobel(Y_D(t_m)) = \sqrt{(H * Y_D(t_m))^2 + (V * Y_D(t_m))^2} \tag{3.5}$$

   gdzie wartości bezwzględne i dodawanie są obliczane piksel po pikselu.

3. SI_S dla t_n i SI_D dla t_m oblicza się jako odchylenie standardowe (std) po filtracji Sobel:

$$SI_S(t_n) = std[Sobel(Y_S(t_n))] \tag{3.6}$$

$$SI_D(t_m) = \frac{std[Sobel(Y_S(t_m))]}{G} \tag{3.7}$$

   gdzie odchylenie standardowe obliczane jest dla wszystkich próbek, G oznacza wzmocnienie kanału transmisyjnego lub HRC. G można łatwo obliczyć na podstawie porównania poziomów sygnałów czerni i bieli:

$$G = \frac{przeznaczenie_{biel} - przeznaczenie_{czerń}}{źródo_{biel} - źródo_{czerń}} \tag{3.8}$$

4. Powyższych obliczeń dokonuje się dla wszystkich ramek testowanej sekwencji na ekranie monitora.

Pomiar informacji czasowej źródła wideo (TI_S) i informacji czasowej przeznaczenia wideo (TI_D). Rozróżnia się trzy typy informacji czasowej TI_{mean} (wartość średnia), TI_{std} (odchylenie standardowe), TI_{rms} (wartość średniokwadratowa). Pomiarów dokonuje się tak samo dla źródła i przeznaczenia wideo. TI_S i TI_D charakteryzują płynność ruchu w scenach wideo. Cechy te służą do obliczenia obiektywnych parametrów jakości wideo będących odzwierciedleniem pogorszenia czasowego obrazu.

Metoda pomiaru:

1. Sygnały źródła i przeznaczenia są próbkowane i wartości $Y_S(t_n)$ i $Y_D(t_m)$ reprezentują odpowiednio luminancję źródła i przeznaczenia w czasie t_n i t_m.
2. Komponenty informacji czasowej są obliczane z zależności:

$$TI_{meanS}(t_n) = mean|Y_S(t_n) - Y_S(t_{n-1})| \tag{3.9}$$

$$TI_{stdS}(t_n) = std|Y_S(t_n) - Y_S(t_{n-1})| \tag{3.10}$$

$$TI_{rmsS}(t_n) = \sqrt{[TI_{meanS}(t_n)]^2 + [TI_{stdS}(t_n)]^2} \tag{3.11}$$

$$TI_{meanD}(t_m) = \frac{mean|Y_D(t_m) - Y_D(t_{m-1})|}{G} \tag{3.12}$$

$$TI_{stdD}(t_m) = \frac{std|Y_D(t_m) - Y_D(t_{m-1})|}{G} \tag{3.13}$$

$$TI_{rmsD}(t_m) = \sqrt{[TI_{meanD}(t_m)]^2 + [TI_{stdD}(t_m)]^2} \tag{3.14}$$

Działania odejmowania i wyznaczania wartości bezwzględnej w równaniach (3.9), (3.10), (3.12) i (3.13) są wykonywane piksel po pikselu, a wartości średnie i odchylenia standardowe są obliczane przy użyciu pikseli wynikowych. Wartość G można wyznaczyć jak wyżej. W dalszych obliczeniach $TI_S(t_n)$ i $TI_D(t_m)$ są rozumiane jako wartości średniokwadratowe.

3. Powyższych obliczeń dokonuje się dla wszystkich ramek testowanego sceny wideo.

Czasowe dopasowanie źródła i przeznaczenia obrazu. Przed rozpoczęciem obliczeń obiektywnych parametrów jakości obrazu zmierzone cechy muszą zostać zgrane w czasie. Może być do tego użyta energia ruchu zawarta w obrazie źródła i przeznaczenia, wykorzystuje się wtedy automatyczny algorytm do określania opóźnienia transmisji w kanale. Algorytm polega na odfiltrowaniu maksymalnych wartości TI_S i TI_D, a następnie wyliczeniu dopasowania czasowego na podstawie minimum odchylenia standardowego różnicy przebiegów (dla uproszczenia przyjmiemy $d_v = t_m - t_n$)[13].

3.2.2.2. Metoda obliczania obiektywnych parametrów jakości obrazu

Wszystkie parametry przedstawione poniżej zostały zaprojektowane tak, aby odzwierciedlać pogorszenie jakości obrazu wywołane przez dostrzegalne zniekształcenia.

* P_1: maksymalny stosunek TI

  Najlepszą funkcją porównująca TI okazał się log10 ze stosunku TI_D do TI_S

$$TIratio(t_n) = \log 10\left(\frac{TI_D(t_m)}{TI_s(t_n)}\right) \qquad (3.15)$$

$$P_1 = \max\left[\max_{time}(TIratio(t_n)),0\right]$$

  Jeśli wartość Tiratio(t_n) ma wartość dodatnią, oznacza to, że energia ruchu została dodana do sekwencji przeznaczenia w porównaniu do sekwencji źródła.

* P_2: wartość średniokwadratowa stosunku TI

---

11. Rec. ITU-T G.711: *Pulse code modulation PCM of voice ferquencies*, Geneva 1972

Parametr ten jest miarą średniej różnicy ruchu między obrazami źródła i przeznaczenia

$$P_2 = rms_{time}[TIratio(t_n)] \tag{3.16}$$

* P_3: różnica maksymalnego i minimalnego stosunku TI

Parametr ten jest podobny do parametru P_1, z tą różnicą, że uwzględnia straconą energię ruchu. Energia ruchu jest tracona w wypadku, gdy HRC przeprowadza powtarzanie ramek.

$$P_3 = \max[\max_{time}(TIratio(t_n)),0] - \min[\min_{time}(TIratio(t_n)),0] \tag{3.17}$$

* P_4: różnica dodatniej i ujemnej wartości średniej stosunku TI

Parametr ten jest formą parametru P_3, jednak nie jest tak wrażliwy na wartości szczytowe TIratio(t_n)

$$P_4 = posmean_{time}[TIratio(t_n)] - negmean_{time}[TIratio(t_n)] \tag{3.18}$$

Dodatnią wartość średnią (posmean) wyznacza się jako sumę dodatnich TIratio(t_n) podzieloną przez liczbę dodatnich wartości średnich. Analogicznie wyznacza się ujemną wartość średnią (negmean). Jeżeli nie ma dodatnich wartości średnich to posmean=0 lub, gdy nie będzie ujemnych wartości to negmean=0.

* P_5: wartość średniokwadratowa stosunku błędów TI

Tak jak TIratio(t_n), TIerrorratio(t_n) jest bardzo przydatną funkcją. Jest to stosunek ilości błędów w TI między źródłem i przeznaczeniem obrazu, unormowanym do TI źródła.

$$TIerrorratio(t_n) = \frac{TI_S(t_n) - TI_D(t_m)}{TI_S(t_n)} \tag{3.19}$$

$$P_5 = rms_{time}[TIerrorratio(t_n)] \tag{3.20}$$

* P_6: wartość średniokwadratowa dodatniej części stosunku błędów TI (stracona energia ruchu)

  TIerrorratio(t_n) przyjmuje wartości dodatnią, gdy sekwencja przeznaczenia zawiera mniej informacji czasowej niż sekwencja źródła, informacja ta może zostać utracona np. przy powtórzeniu ramki przez HRC. A zatem parametr ten jest miarą ilości powtórzonych ramek przez HRC.

$$P_6 = rms_{time}[\max(TIerrorratio(t_n),0)] \tag{3.21}$$

* P_7: maksymalna wartość bezwzględna stosunku błędów SI

  SIerrorratio(t_n) jest to stosunek ilości błędów w SI między źródłem i przeznaczeniem wideo, unormowanym do SI źródła.

$$SIerrorratio(t_n) = \frac{SI_S(t_n) - SI_D(t_m)}{SI_S(t_n)} \tag{3.22}$$

  SIerrorratio(t_n) przyjmuje wartości dodatnią, gdy sekwencja przeznaczenia zawiera mniej informacji przestrzennej niż sekwencja źródła, jest to wynikiem zamazywania obrazu.

  Parametr P_7 mierzy maksymalną wartość bezwzględną stosunku błędów SI.

$$P_7 = \max_{time}\left(\left|SIerrorratio(t_n)\right|\right) \tag{3.23}$$

* P_8: wartość średniokwadratowa stosunku błędów SI

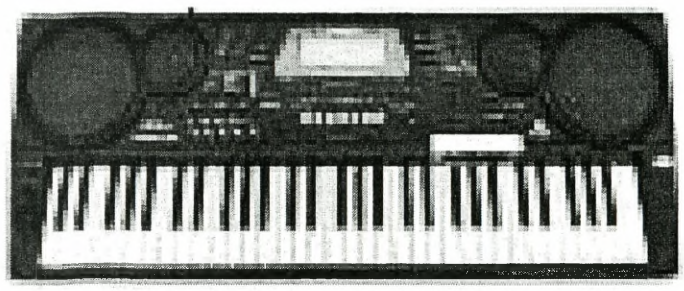
  Parametr ten wprowadza miarę różnicy w informacji przestrzennej między obrazami źródła i przeznaczenia.

$$P_8 = rms_{time}[SIerrorratio(t_n)]. \tag{3.24}$$

4. Generowanie dźwięków

Zadanie to polegało na wygenerowaniu oryginalnych dźwięków, które będzie można zaszyć w programach. By nie narazić się na plagiat lub wykorzystanie bez zgody innych autorów ich próbek dźwiękowych należało skomponować dźwięki oraz nagrać odpowiednie komunikaty głosowe.

Użyto syntezatora CASIO model CTK-611, który ma możliwość współpracy z komputerem poprzez złącze MIDI.



Rys. 2. Ilustracja syntezatora CTK-611

Przy jego wykorzystaniu wygenerowano szereg próbek dźwiękowych, a następnie za pomocą procedur programowych zapewniono dowiązanie ich do odpowiednich zdarzeń w programie.

Użyto do tego celu części komponentu DirectX o nazwie DirectSound. Komponent ten obsługuje karty dźwiękowe, ale przede wszystkim pozwala tworzyć i wydajnie obsługiwać bufory dźwiękowe.

Poniżej zostanie przedstawiona istota implementacji mechanizmu obsługującego efekty dźwiękowe. Mechanizm ten został opracowany dla języka wysokopoziomowego w notacji czytelnej dla C++.

4.1. Obsługiwanie zdarzeń dźwiękowych

```
struct musmsg {
 long msg_type;
char msg_text[12];
};
```

```
extern int msg_id;
#endif
// lista dołączanych bibliotek
// Biblioteka 1
// Biblioteka 2
:
// Biblioteka N-1
// Biblioteka N
// Efekt 3D Sound
#ifdef HW3SOUND
#endif

// komendy sterujące do serwera
#ifdef MUSSERV
consvar_t musserver_cmd = {"musserver_cmd","musserver",CV_SAVE};
consvar_t musserver_arg = {"musserver_arg","-t 20 -f -u 0",CV_SAVE};
#endif
#ifdef SNDSERV
consvar_t sndserver_cmd = {"sndserver_cmd","llsndserv",CV_SAVE};
consvar_t sndserver_arg = {"sndserver_arg","-quiet",CV_SAVE};
#endif

#if defined (__WIN32__) && !defined (SURROUND)
#define SURROUND
#endif

#ifdef __MACOS__
consvar_t  play_mode = {"play_mode","0",CV_SAVE,CV_Unsigned};
#endif
```

```
// stereo odwrócenie: 1=true, 0=false
consvar_t stereoreverse = {"stereoreverse","0",CV_SAVE ,CV_OnOff};

consvar_t precachesound = {"precachesound","0",CV_SAVE ,CV_OnOff};

CV_PossibleValue_t
soundvolume_cons_t[]={{0,"MIN"},{31,"MAX"},{0,NULL}};
cv_soundvolume = {"soundvolume","15",CV_SAVE,soundvolume_cons_t};
consvar_t                    cv_musicvolume                    =
{"musicvolume","15",CV_SAVE,soundvolume_cons_t};

// numery ścieżek do dyspozycji
void SetChannelsNum(void);
consvar_t cv_numChannels = {"snd_channels","16",CV_SAVE | CV_CALL,
CV_Unsigned,SetChannelsNum};

#ifdef SURROUND
consvar_t surround = {"surround", "0", CV_SAVE, CV_OnOff};
#endif

#define S_MAX_VOLUME        127

// w przypadku gdy clip dźwiękowy na zewnątrz
#define S_CLIPPING_DIST     (1200*0x10000)

#define S_CLOSE_DIST        (160*0x10000)

// regulacja głośności
```

```
#define    S_ATTENUATOR                                ((S_CLIPPING_DIST-
S_CLOSE_DIST)>>(FRACBITS+4))

// wyrónywanie menu
#define NORM_VOLUME          snd_MaxVolume

#define NORM_PITCH           128
#define NORM_PRIORITY        64
#define NORM_SEP             128

#define S_PITCH_PERTURB      1
#define S_STEREO_SWING       (96*0x10000)

#ifdef SURROUND
#define SURROUND_SEP         -128
#endif

// procent tłumienności
#define S_IFRACVOL           30

typedef struct
{
   // informacja o dźwięku (jeśli pusty, użytkuj kanał)
   sfxinfo_t*  sfxinfo;

   // muza
   void*       origin;

   // posługiwanie się dźwiękiem uprzednio nagranym
```

```
    int         handle;

} channel_t;


// komplet ścieżek będących w dyspozycji
static channel_t*       channels;


// pouzowanie dźwieków
static boolean          mus_paused;

// aktualnie wydawane dźwięki
static musicinfo_t*     mus_playing=0;

static int              nextcleanup;


//
// wewnętrzne mechanizmy
//
int
S_getChannel
( void*         origin,
  sfxinfo_t*    sfxinfo );


int
S_AdjustSoundParams
```

```
 ( mobj_t*       listener,
   mobj_t*       source,
   int*          vol,
   int*          sep,
   int*          pitch );

 static void S_StopChannel(int cnum);


 void S_RegisterSoundStuff (void)
 {
        //added:11-04-98: stereoreverse
    CV_RegisterVar (&stereoreverse);
    CV_RegisterVar (&precachesound);

#ifdef SNDSERV
    CV_RegisterVar (&sndserver_cmd);
    CV_RegisterVar (&sndserver_arg);
#endif
#ifdef MUSSERV
    CV_RegisterVar (&musserver_cmd);
    CV_RegisterVar (&musserver_arg);
#endif
#ifdef SURROUND
    CV_RegisterVar (&surround);
#endif

#ifdef __MACOS__        //mp3 playlist stuff
    {
```

```
        int i;
        for (i=0;i<PLAYLIST_LENGTH;i++)
        {
           user_songs[i].name = malloc(7);
           sprintf(user_songs[i].name, "song%i%i",i/10,i%10);
           user_songs[i].defaultvalue = malloc(1);
           *user_songs[i].defaultvalue = 0;
           user_songs[i].flags = CV_SAVE;
           user_songs[i].PossibleValue = NULL;
           CV_RegisterVar (&user_songs[i]);
        }
        CV_RegisterVar (&play_mode);
    }
#endif
}

void SetChannelsNum(void)
{
   int i;
// ustawianie
   // ścieżki do zagrania
   // wartości maksymalnej
   // ustalenie strefy
   if(channels)
     Z_Free(channels);

#ifdef HW3SOUND
   if (hws_mode != HWS_DEFAULT_MODE)
   {
```

```
        HW3S_SetSourcesNum();
        return;
    }
#endif
    channels =
     (channel_t        *)        Z_Malloc(cv_numChannels.value*sizeof(channel_t),
PU_STATIC, 0);

    // używanie wolnych kanałów
    for (i=0 ; i<cv_numChannels.value ; i++)
      channels[i].sfxinfo = 0;

    }

void S_InitRuntimeMusic()
{
  int i;

  for(i = mus_firstfreeslot; i < mus_lastfreeslot; i++)
    S_music[i].name = NULL;
}

//
// inicjacja procedury
// ustawianie kanału i głośności
//  wydziela bufor ścieżki
//
void S_Init ( int          sfxVolume,
              int          musicVolume )
```

```
{
    int         i;

     S_SetSfxVolume (sfxVolume);
    S_SetMusicVolume (musicVolume);

    SetChannelsNum();

// w przypadku gdy żadne dźwięki nie są odtwarzane ani nie jest włączona
// pouza
    mus_paused = 0;

    // żadne dźwięki nie zostały zadeklarowane
    for (i=1 ; i<NUMSFX ; i++)
        S_sfx[i].lumpnum = S_sfx[i].usefulness = -1;      // for I_GetSfx()

    //
    if(!nosound && (M_CheckParm("-precachesound") || precachesound.value) )
    {
        // inicjacja zewnętrznych danych dźwiękowych
        CONS_Printf("Loading sounds... ");

        for (i=1 ; i<NUMSFX ; i++)
        {
            // łączenie w czasie gdy następuje start  (nie wcześniej !!!)
            if (S_sfx[i].name && !S_sfx[i].link)
                S_sfx[i].data = I_GetSfx (&S_sfx[i]);

        }
```

```
        CONS_Printf(" pre-cached all sound data\n");
    }

    S_InitRuntimeMusic();
}

// odzyskiwanie zwolnionego numeru
//
int S_GetSfxLumpNum (sfxinfo_t* sfx)
{
    char namebuf[9];
    int  sfxlump;

    if( gamemode == heretic )
        strncpy(namebuf, sfx->name, 9);
    else
        sprintf(namebuf, "ds%s", sfx->name);

    sfxlump=W_CheckNumForName(namebuf);
    if (sfxlump>0)
        return sfxlump;

    if( gamemode != heretic )
        strncpy(namebuf, sfx->name, 9);
    else
        sprintf(namebuf, "ds%s", sfx->name);

    sfxlump=W_CheckNumForName(namebuf);
```

```
    if (sfxlump>0)
        return sfxlump;

    if( gamemode == heretic)
        return W_GetNumForName ("keyup");
    else
        return W_GetNumForName ("dspistol");
}


//
// uruchomienie kodu
// zatrzymanie warunkowe
void S_StopSounds()
{
  int cnum;

#ifdef HW3SOUND
  if (hws_mode != HWS_DEFAULT_MODE)
  {
     HW3S_StopSounds();
     return;
  }
#endif

  // stop dla wszystkich dźwieków
  for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
    if (channels[cnum].sfxinfo)
      S_StopChannel(cnum);
```

```
}


void S_Start(void)
{
 int mnum;

 // start nowego dźwięku
 mus_paused = 0;

 if (gamemode == commercial)
  mnum = mus_runnin + gamemap - 1;
 else if( gamemode == heretic )
    mnum = mus_he1m1 + (gameepisode-1)*9 + gamemap - 1;
 else
 {
  int spmus[]=
  {
   // gdzie ma grać?

   mus_e3m4, // American    e4m1
   mus_e3m2, // Romero      e4m2
   mus_e3m3, // Shawn       e4m3
   mus_e1m5, // American    e4m4
   mus_e2m7, // Tim  e4m5
   mus_e2m4, // Romero      e4m6
   mus_e2m6, // J.Anderson   e4m7 CHIRON.WAD
   mus_e2m5, // Shawn       e4m8
   mus_e1m9  // Tim         e4m9
```

```
    };

    if (gameepisode < 4)
      mnum = mus_e1m1 + (gameepisode-1)*9 + gamemap-1;
    else
      mnum = spmus[gamemap-1];
    }

  if(info_music && *info_music)
    S_ChangeMusicName(info_music, true);
  else
    S_ChangeMusic(mnum, true);

  nextcleanup = 15;
}

void S_StartSoundAtVolume( void*       origin_p,
                           int         sfx_id,
                           int         volume )
{

  int         sep;
  int         pitch;
  int         priority;
  sfxinfo_t*  sfx;
  int         cnum;

  mobj_t*     origin = (mobj_t *) origin_p;
```

```
    if(nosound || (origin && origin->type == MT_SPIRIT))
        return;

#ifdef HW3SOUND
    if (hws_mode != HWS_DEFAULT_MODE)
    {
        HW3S_StartSound(origin, sfx_id);
        return;
    };
#endif

    // Śledzenie
    /*fprintf( stderr,
            "S_StartSoundAtVolume: playing sound %d (%s)\n",
            sfx_id, S_sfx[sfx_id].name );*/

#ifdef PARANOIA
    // check for bogus sound #
    if (sfx_id < 1 || sfx_id > NUMSFX)
        I_Error("Bad sfx #: %d\n", sfx_id);
#endif

    sfx = &S_sfx[sfx_id];

    if (sfx->skinsound!=-1 && origin && origin->skin)
    {
        // przeadresowanie kanału
        sfx_id = ((skin_t *)origin->skin)->soundsid[sfx->skinsound];
        sfx    = &S_sfx[sfx_id];
```

```
    }

    // Inicjacja parametrów dźwięku
    if (sfx->link)
    {
      pitch = sfx->pitch;
      priority = sfx->priority;
      volume += sfx->volume;

      if (volume < 1)
        return;

    //    ustawienie głośności
    }
    else
    {
      pitch = NORM_PITCH;
      priority = NORM_PRIORITY;
    }


    // sprawdzenie i powtórne ustawienie paramnetrów
    if (origin && origin != players[displayplayer].mo && !(cv_splitscreen.value
&& origin == players[secondarydisplayplayer].mo))
    {
       int          rc,rc2;
       int volume2=volume,sep2/*=sep*/,pitch2=pitch;
       rc=S_AdjustSoundParams(players[displayplayer].mo,
                              origin,
```

```
                &volume,
                &sep,
                &pitch);
if(cv_splitscreen.value)
{
    rc2=S_AdjustSoundParams(players[secondarydisplayplayer].mo,
                origin,
                &volume2,
                &sep2,
                &pitch2);
    if(!rc2)
    {
       if( !rc )
          return;
    }
    else
    if(!rc || (rc && volume2>volume))
    {
       volume=volume2;
       sep=sep2;
       pitch=pitch2;
       if ( origin->x == players[secondarydisplayplayer].mo->x &&
            origin->y == players[secondarydisplayplayer].mo->y )
       {
          sep      = NORM_SEP;
       }
    }
}
else
```

```
        if(!rc) return;

      if ( origin->x == players[displayplayer].mo->x &&
           origin->y == players[displayplayer].mo->y )
     {
      sep      = NORM_SEP;
     }
    }
    else
    {
     sep = NORM_SEP;
    }

    // zmiana przedziału tonów

// Wygaszenie starego dźwięku przed rozpoczęciem odtwarzania nowego
    S_StopSound(origin);

    // try to find a channel
    cnum = S_getChannel(origin, sfx);

    if (cnum<0)
     return;

    //
// buforowanie do pamięci notanikowej
    if (sfx->link)
       sfx->data = sfx->link->data;
```

```
    if (!sfx->data)
    {
        if (!sfx->link)
            sfx->data = I_GetSfx (sfx);
        else
        {
            sfx->data = I_GetSfx (sfx->link);
            sfx->link->data = sfx->data;
        }
    }

    // zwiększanie mocy efektu
    if (sfx->usefulness++ < 0)
        sfx->usefulness = -1;


#ifdef SURROUND
    // wykonanie
    if (stereoreverse.value && sep != SURROUND_SEP)
        sep = (~sep) & 255;
#else
    if (stereoreverse.value)
        sep = (~sep) & 255;
#endif

    channels[cnum].handle = I_StartSound(sfx_id,
                                /*sfx->data,*/
                                volume,
                                sep,
```

```
                    pitch,
                    priority);
}

void S_StartSound( void*        origin,
                   int          sfx_id )
{
  // zwiększenei głośności
#ifdef HW3SOUND
  if (hws_mode != HWS_DEFAULT_MODE)
    HW3S_StartSound(origin, sfx_id);
  else
#endif
    S_StartSoundAtVolume(origin, sfx_id, 255);
}


void S_StopSound(void *origin)
{
  int cnum;

// blokowanie uruchomienia nowych ścieżek dźwiękowych
  if(!origin)
   return;

#ifdef HW3SOUND
  if (hws_mode != HWS_DEFAULT_MODE)
  {
    HW3S_StopSound(origin);
```

```
        return;
    }
    else
    {
#endif
        for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
        {
            if (channels[cnum].sfxinfo && channels[cnum].origin == origin)
            {
                S_StopChannel(cnum);
                break;
            }
        }
#ifdef HW3SOUND
    }
#endif
}



//
// stop i pauza w czasie odtwarzania
//
void S_PauseSound(void)
{
    if (mus_playing && !mus_paused)
    {
        I_PauseSong(mus_playing->handle);
        mus_paused = true;
    }
```

```
    // pauza
    I_PauseCD ();
#else
    I_StopCD ();
#endif
}

void S_ResumeSound(void)
{
    if (mus_playing && mus_paused)
    {
        I_ResumeSong(mus_playing->handle);
        mus_paused = false;
    }

    I_ResumeCD ();
}

//
// uaktulanienie danych w buforze
//

void S_UpdateSounds(void)
{
    int         audible;
    int         cnum;
    int         volume;
    int         sep;
```

```
    int         pitch;
    sfxinfo_t*  sfx;
    channel_t*  c;

    mobj_t*     listener = players[displayplayer].mo;

// możliwość włączenia ręcznego sterowania
    if (actualsfxvolume != cv_soundvolume.value)
        S_SetSfxVolume (cv_soundvolume.value);
    if (actualmusicvolume != cv_musicvolume.value)
        S_SetMusicVolume (cv_musicvolume.value);

#ifdef HW3SOUND
    if (hws_mode != HWS_DEFAULT_MODE)
    {
        HW3S_UpdateSources();
        return;
    }
#endif

    /* czyszczenie bufora z danych
    if (gametic > nextcleanup)
    {
        for (i=1 ; i<NUMSFX ; i++)
        {
            if (S_sfx[i].usefulness==0)
            {
                //S_sfx[i].usefulness--;
```

```
                CONS_Printf ("\2flushed sfx %.6s\n", S_sfx[i].name);
        }
    }
    nextcleanup = gametic + 15;
}*/

for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
{
    c = &channels[cnum];
    sfx = c->sfxinfo;

    if (c->sfxinfo)
    {
        if (I_SoundIsPlaying(c->handle))
        {

            volume = 255;
            pitch = NORM_PITCH;
            sep = NORM_SEP;

            if (sfx->link) // strange (BP)
            {
                pitch = sfx->pitch;
                volume += sfx->volume;
                if (volume < 1)
                {
                    S_StopChannel(cnum);
                    continue;
                }
```

```
        }

// sprawdzenie czy operacje czekające na obsłużenie nie wejdą w kolizję
        if (c->origin && listener != c->origin && !(cv_splitscreen.value &&
c->origin==players[secondarydisplayplayer].mo))
        {
          int audible2;
          int volume2=volume,sep2=sep,pitch2=pitch;
          audible = S_AdjustSoundParams(listener,
                                        c->origin,
                                        &volume,
                                        &sep,
                                        &pitch);

          if(cv_splitscreen.value)
          {

audible2=S_AdjustSoundParams(players[secondarydisplayplayer].mo,
                                        c->origin,
                                        &volume2,
                                        &sep2,
                                        &pitch2);
            if(audible2 && (!audible || (audible && volume2>volume)))
            {
              audible=true;
              volume=volume2;
              sep=sep2;
              pitch=pitch2;
            }
```

```
                }

                if (!audible)
                {
                   S_StopChannel(cnum);
                }
                else
                   I_UpdateSoundParams(c->handle, volume, sep, pitch);
             }
          }
          else
          {
// jeśli nie odtwarzaj
             S_StopChannel(cnum);
          }
       }
    }

}


void S_SetMusicVolume(int volume)
{
   if (volume < 0 || volume > 31)
      CONS_Printf("musicvolume should be between 0-31\n");

   CV_SetValue (&cv_musicvolume, volume&31);
   actualmusicvolume = cv_musicvolume.value;   //sprawdzenie zmiennych
```

```
#ifndef __WIN32__
   I_SetMusicVolume(31);
#endif
   I_SetMusicVolume(volume&31);
}

void S_SetSfxVolume(int volume)
{
   if (volume < 0 || volume > 31)
      CONS_Printf("sfxvolume should be between 0-31\n");

   CV_SetValue (&cv_soundvolume, volume&31);
   actualsfxvolume = cv_soundvolume.value;      //sprawdzenie zmiennych

#ifdef HW3SOUND
   hws_mode == HWS_DEFAULT_MODE
      ? I_SetSfxVolume (volume&31)
      : HW3S_SetSfxVolume(volume & 31);
#else
   // now hardware volume
   I_SetSfxVolume(volume&31);
#endif

}

//
// start odtwarzania próbki
//
void S_StartMusic(int m_id)
```

```
{
   S_ChangeMusic(m_id, false);
}

//
void S_ChangeMusicName(char *name, int looping)
{
 int  music;

 if(!strncmp(name, "-", 6))
 {
  S_StopMusic();
  return;
 }

 music = S_FindMusic(name);

 if(music > mus_None && music < NUMMUSIC)
  S_ChangeMusic(music, looping);
 else
  {
   CONS_Printf("music not found: %s\n", name);
   S_StopMusic(); // blokada odtwarzania
  }
}

void S_ChangeMusic( int                 music_num,
                    int                 looping )
{
```

```
musicinfo_t*  music;

if( nomusic )
   return;

if ( (music_num <= mus_None) ||
    (music_num >= NUMMUSIC) )
{
   CONS_Printf ("ERROR: Bad music number %d\n", music_num);
   return;
}
else
   music = &S_music[music_num];


if (mus_playing == music)
   return;

// shutdown old music
S_StopMusic();

// indeksowanie
if (!music->lumpnum)
{
   if( gamemode == heretic )
      music->lumpnum = W_GetNumForName( music->name );
   else
      music->lumpnum = W_GetNumForName( va("d_%s", music->name));
}
```

```
    // rezerwacja rejestru
    music->data = (void *) W_CacheLumpNum(music->lumpnum, PU_MUSIC);
#ifdef __MACOS__
    music->handle = I_RegisterSong(music_num);
#else
    music->handle = I_RegisterSong(music->data,W_LumpLength(music-
>lumpnum));
#endif

#ifdef MUSSERV

    if (msg_id != -1) {
      struct musmsg msg_buffer;

      msg_buffer.msg_type=6;
      memset(msg_buffer.msg_text,0,sizeof(msg_buffer.msg_text));
      sprintf(msg_buffer.msg_text,"d_%s", music->name);
      msgsnd(msg_id,(struct                                    msgbuf
*)&msg_buffer,sizeof(msg_buffer.msg_text),IPC_NOWAIT);
    }

#endif /* #ifdef MUSSERV */

    // play it
    I_PlaySong(music->handle, looping);

    mus_playing = music;
}
void S_StopMusic(void)
```

```
{
    if (mus_playing)
    {
        if (mus_paused)
            I_ResumeSong(mus_playing->handle);
        I_StopSong(mus_playing->handle);
        I_UnRegisterSong(mus_playing->handle);
        Z_ChangeTag(mus_playing->data, PU_CACHE);

        mus_playing->data = 0;
        mus_playing = 0;
    }
}




static void S_StopChannel(int cnum)
{

    int        i;
    channel_t*  c = &channels[cnum];

    if (c->sfxinfo)
    {
        // stop odtwarzania
        if (I_SoundIsPlaying(c->handle))
        {
            I_StopSound(c->handle);
```

```
        }

        for (i=0 ; i<cv_numChannels.value ; i++)
        {
            if (cnum != i
                && c->sfxinfo == channels[i].sfxinfo)
            {
                break;
            }
        }

        c->sfxinfo->usefulness--;
        c->sfxinfo = 0;
    }
}


//
// jeśli dźwięk nie jest słyszalny test zwróci 0
//
int S_AdjustSoundParams ( mobj_t*       listener,
                          mobj_t*       source,
                          int*          vol,
                          int*          sep,
                          int*          pitch )
{
    fixed_t     approx_dist;
    fixed_t     adx;
    fixed_t     ady;
```

```
    angle_t     angle;

    adx = abs(listener->x - source->x);
    ady = abs(listener->y - source->y);

     approx_dist = adx + ady - ((adx < ady ? adx : ady)>>1);

    if (gamemap != 8
       && approx_dist > S_CLIPPING_DIST)
    {
       return 0;
    }

    // odtwarzanie ze źródła
    angle = R_PointToAngle2(listener->x,
                            listener->y,
                            source->x,
                            source->y);

    if (angle > listener->angle)
       angle = angle - listener->angle;
    else
       angle = angle + (0xffffffff - listener->angle);

#ifdef SURROUND

    if (surround.value == 1 && (angle > (ANG90 + (ANG45/3)) && angle <
(ANG270 - (ANG45/3))))
       *sep = SURROUND_SEP;
```

```
    else
    {
#endif

    angle >>= ANGLETOFINESHIFT;

    *sep                =                128                -
(FixedMul(S_STEREO_SWING,finesine[angle])>>FRACBITS);

#ifdef SURROUND
    }
#endif

    // głośność programowo ustawiana
    if (approx_dist < S_CLOSE_DIST)
    {
        *vol = 255; //snd_SfxVolume;
    }
    else
    {
        *vol = (15
                * ((S_CLIPPING_DIST - approx_dist)>>FRACBITS))
            / S_ATTENUATOR;
    }

    return (*vol > 0);
}
int S_getChannel( void*         origin,
                  sfxinfo_t*    sfxinfo )
```

```
{
  // channel number to use
  int        cnum;

  channel_t*  c;

  // znajduj i odtwarzaj poprzez wolny kanał
  for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
  {
    if (!channels[cnum].sfxinfo)
      break;
    else if (origin &&  channels[cnum].origin ==  origin)
    {
      S_StopChannel(cnum);
      break;
    }
  }

  if (cnum == cv_numChannels.value)
  {
    for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
      if (channels[cnum].sfxinfo->priority >= sfxinfo->priority) break;

    if (cnum == cv_numChannels.value)
    {
      return -1;
    }
    else
    {
```

```
            S_StopChannel(cnum);
        }
    }

    c = &channels[cnum];

    c->sfxinfo = sfxinfo;
    c->origin = origin;

    return cnum;
}


int S_SoundPlaying(void *origin, int id)
{
    int        cnum;

#ifdef HW3SOUND
    if (hws_mode != HWS_DEFAULT_MODE)
    {
        return HW3S_SoundPlaying(origin, id);
    }
#endif

    for (cnum=0 ; cnum<cv_numChannels.value ; cnum++)
    {
        if (origin &&  channels[cnum].origin ==  origin)
            return 1;
        if (id != -1 && channels[cnum].sfxinfo - S_sfx == id)
```

```
        return 1;
    }
    return 0;
}

#define MAXNEWSOUNDS 10
int     newsounds[MAXNEWSOUNDS] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

void S_StartSoundName(void *mo, char *soundname)
{
 int  i;
 int  soundnum = 0;
 for(i = sfx_None + 1; i < NUMSFX; i++)
 {
  if(!S_sfx[i].name)
   continue;
  if(!stricmp(S_sfx[i].name, soundname))
  {
   soundnum = i;
   break;
  }
 }

 if(!soundnum)
 {
  for(i = 0; i < MAXNEWSOUNDS; i++)
  {
   if(newsounds[i] == 0)
    break;
```

```
    if(!S_SoundPlaying(NULL, newsounds[i]))
      {S_RemoveSoundFx(newsounds[i]); break;}
   }

   if(i == MAXNEWSOUNDS)
   {
    CONS_Printf("Cannot load another extra sound!\n");
    return;
   }

   soundnum = S_AddSoundFx(soundname, false);
   newsounds[i] = soundnum;
  }
 S_StartSound(mo, soundnum);}
```

4.2. Składanie komunikatów dźwiękowych z głosek

Alternatywą dla dźwięków komponowanych jest nagrywanie poprzez mikrofon odpowiednich komunikatów, a następnie dowiązywanie tych komunikatów do odpowiednich zdarzeń.

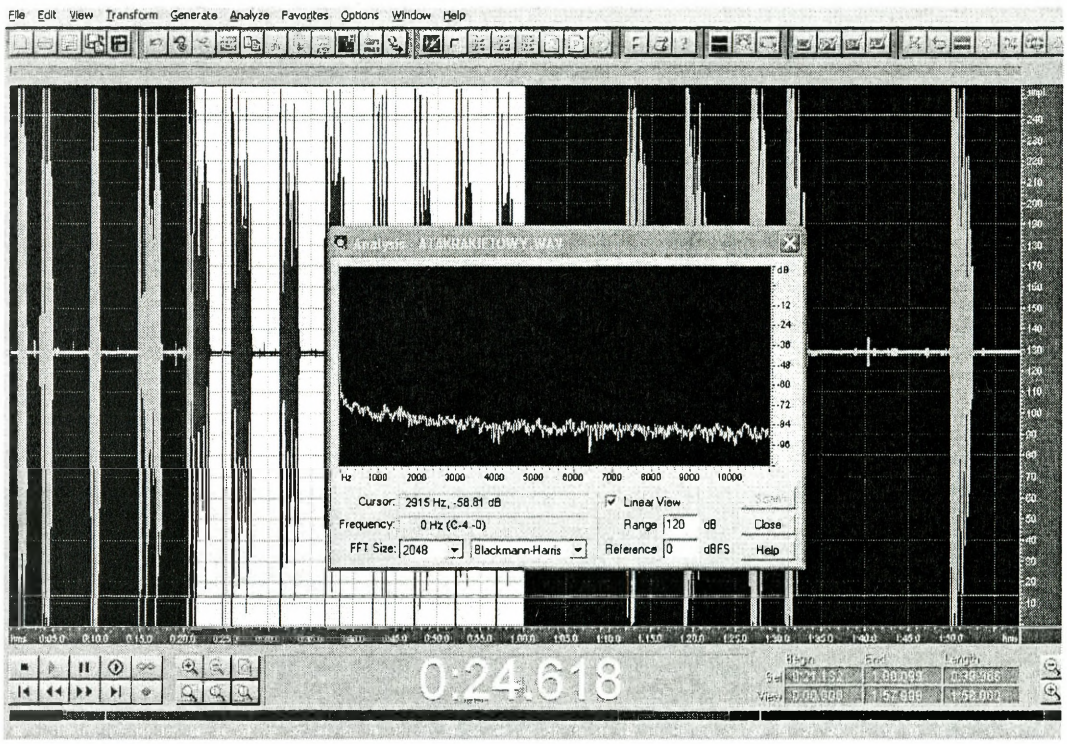
Do celów implementacji przygotowano następujące komunikaty, np.:

- „samolot został zaatakowany przez wyrzutnię rakietową”
- „samolot zaatakował obiekt”
- „samolot został wykryty przez stację radiolokacyjną”.

Trzeba jednak stwierdzić, iż bardzo żmudne i czasochłonne próby nie przyniosły spodziewanych rezultatów. Mimo to opanowano technologie dowiązywania nagranych komunikatów do odpowiednich zdarzeń w programach symulatora. Komunikaty zostały osadzone na maszynach, na których miałoby nastąpić ich odtworzenie.

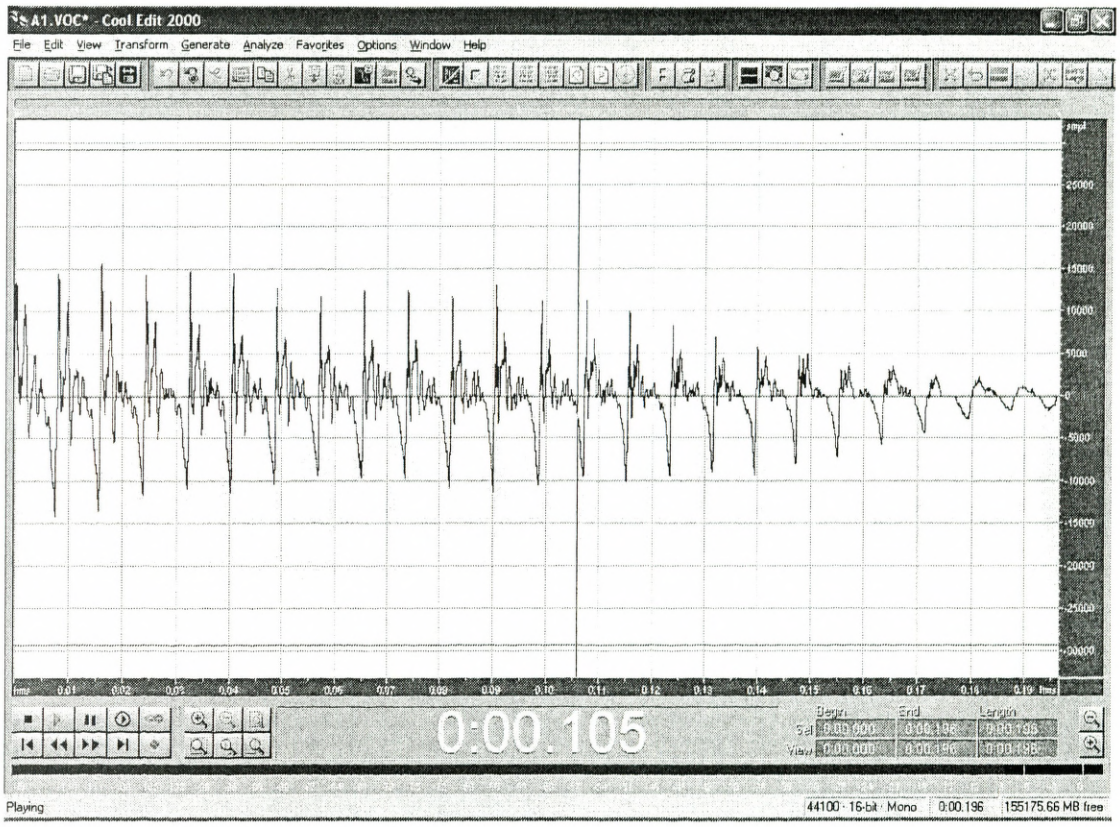
Jednak w dalszym ciągu podczas odtwarzania nawet krótkiego (5 sekundowego) komunikatu zwalniane były wszystkie symulowane procesy.

Komunikaty (pliki wav) zostały dołączone do opracowania.

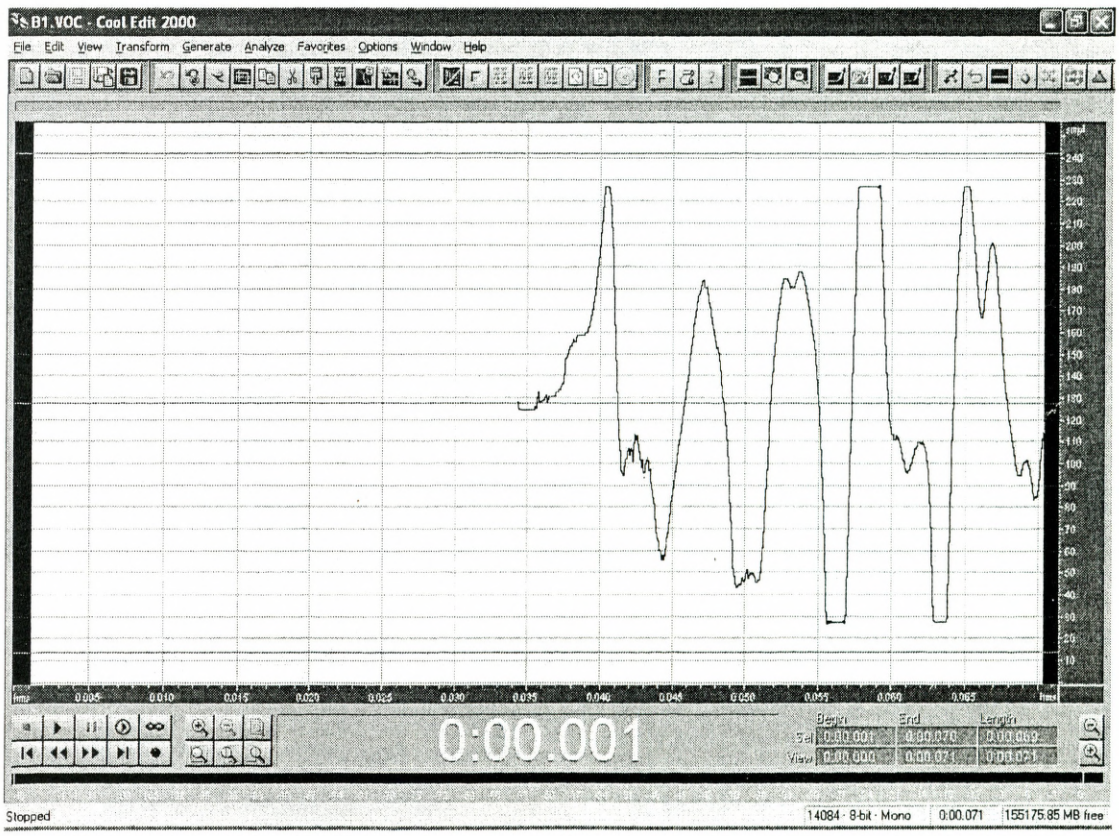


Rys. 3. Interfejs edytora do obróbki dźwięku (odtwarzanie i wycinanie odpowiedniego fragmentu)

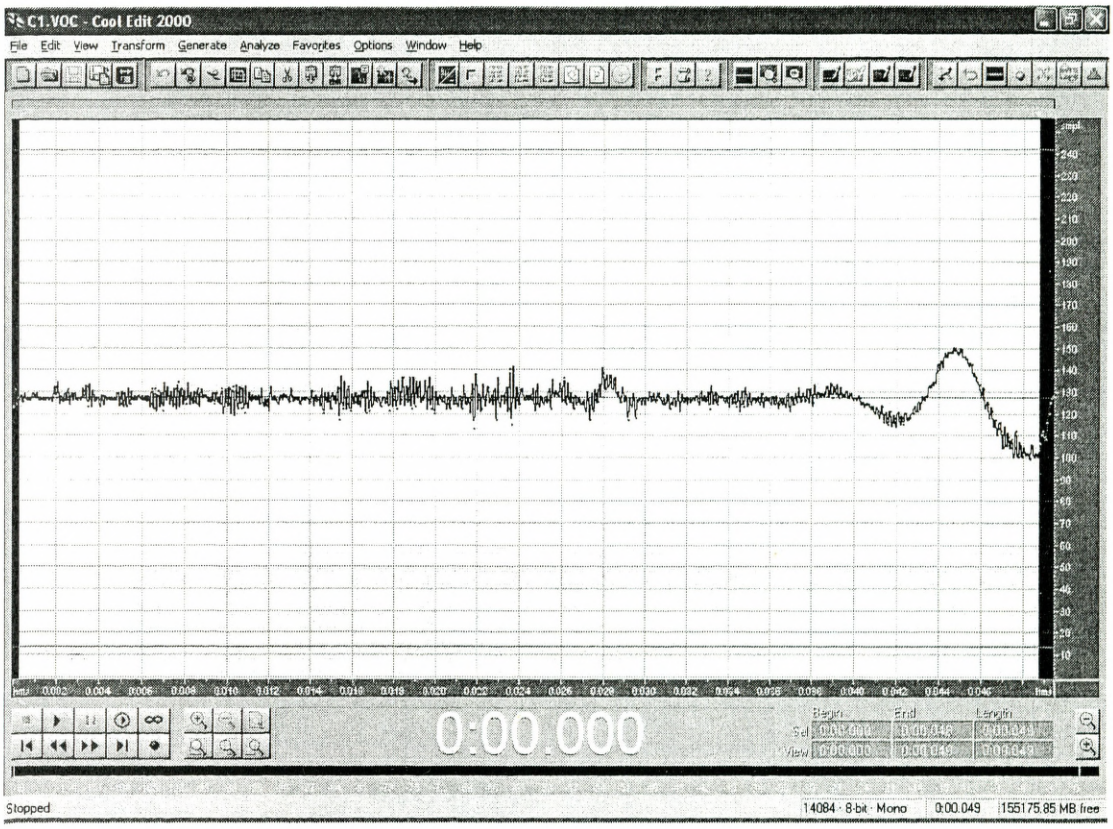
W ramach opracowania nagrano również zbiór głosek występujących w języku polskim, z którego podjęto próbę składania komunikatów.



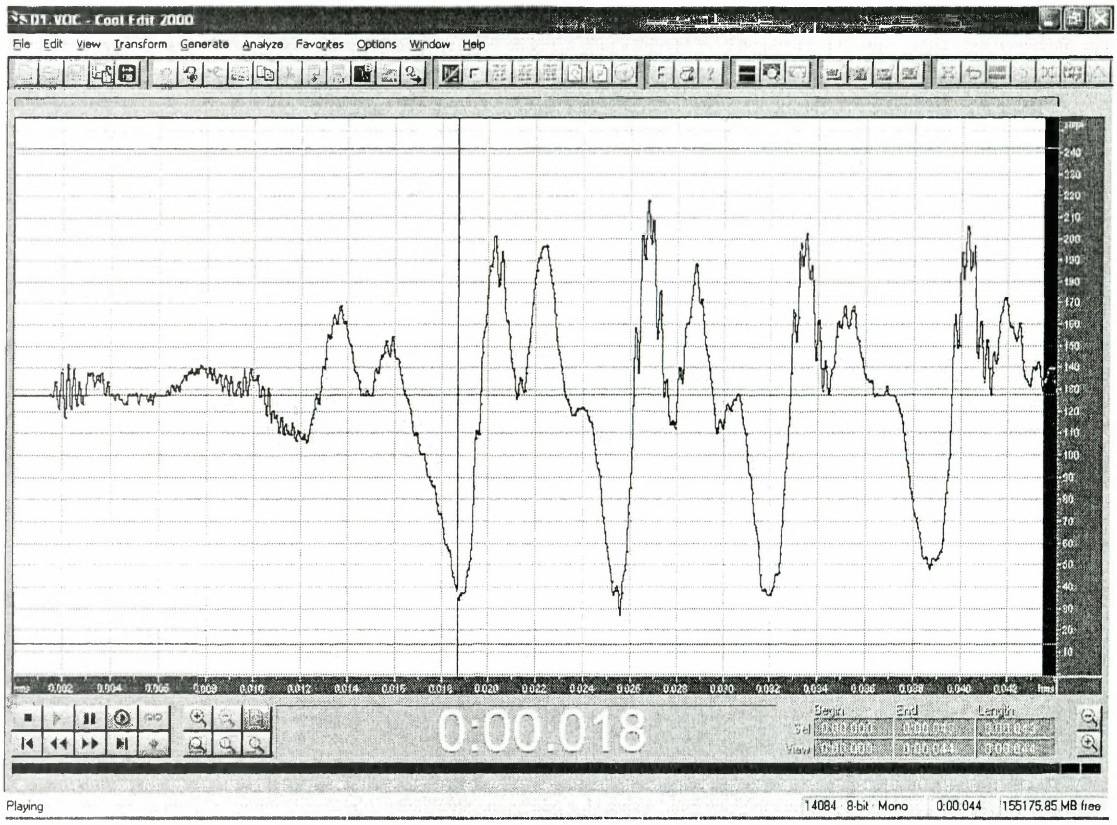
Rys. 4. Ilustracja głoski A



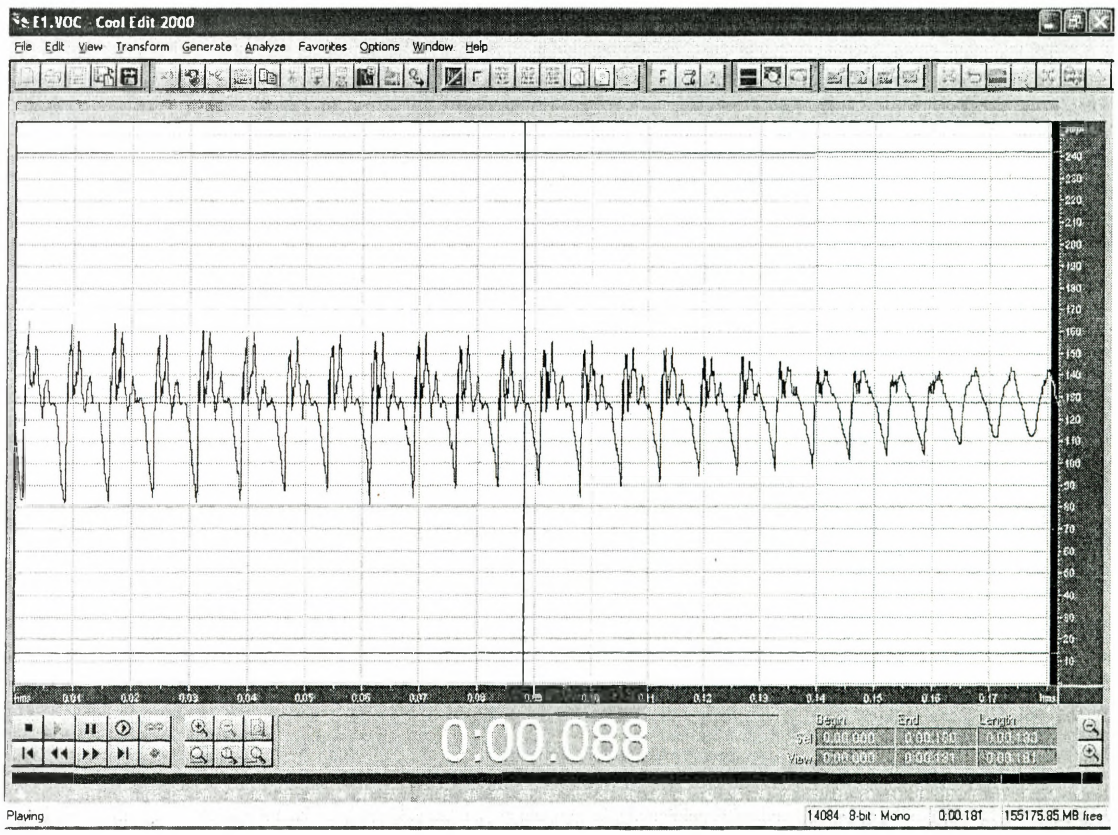
Rys. 5. Ilustracja głoski B



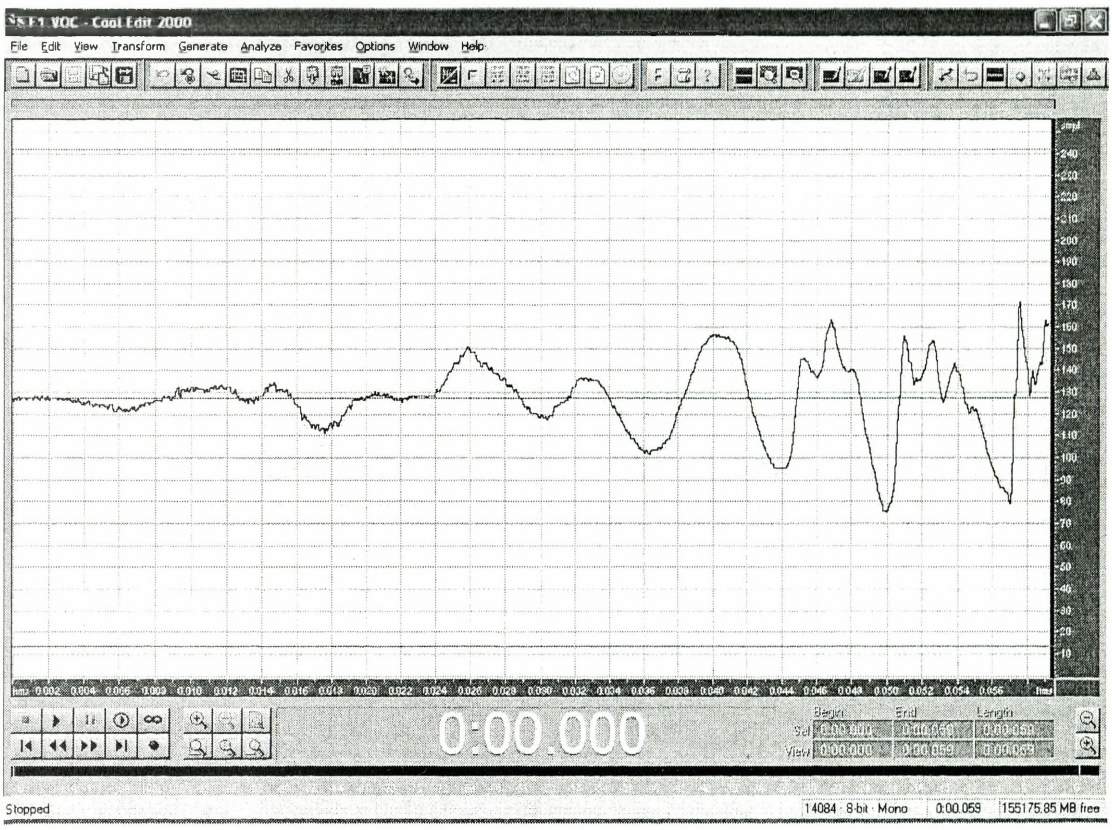
Rys. 6. Ilustracja głoski C



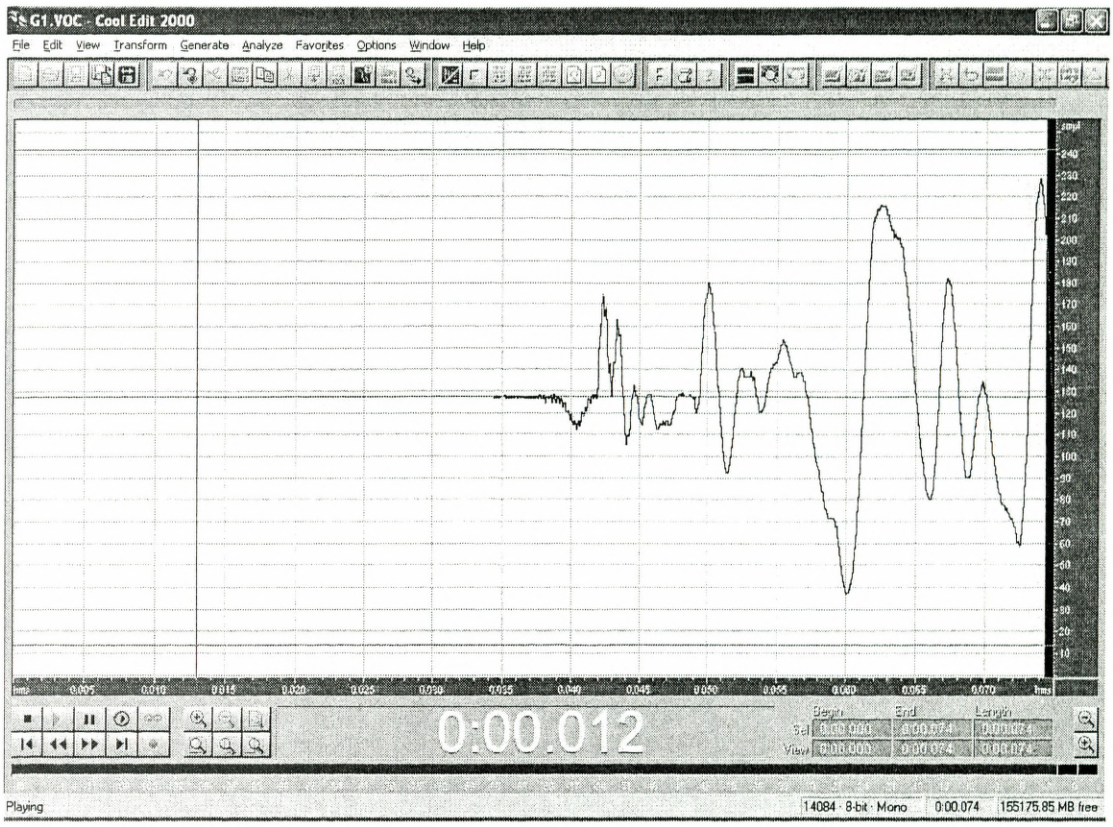
Rys. 7. Ilustracja głoski D



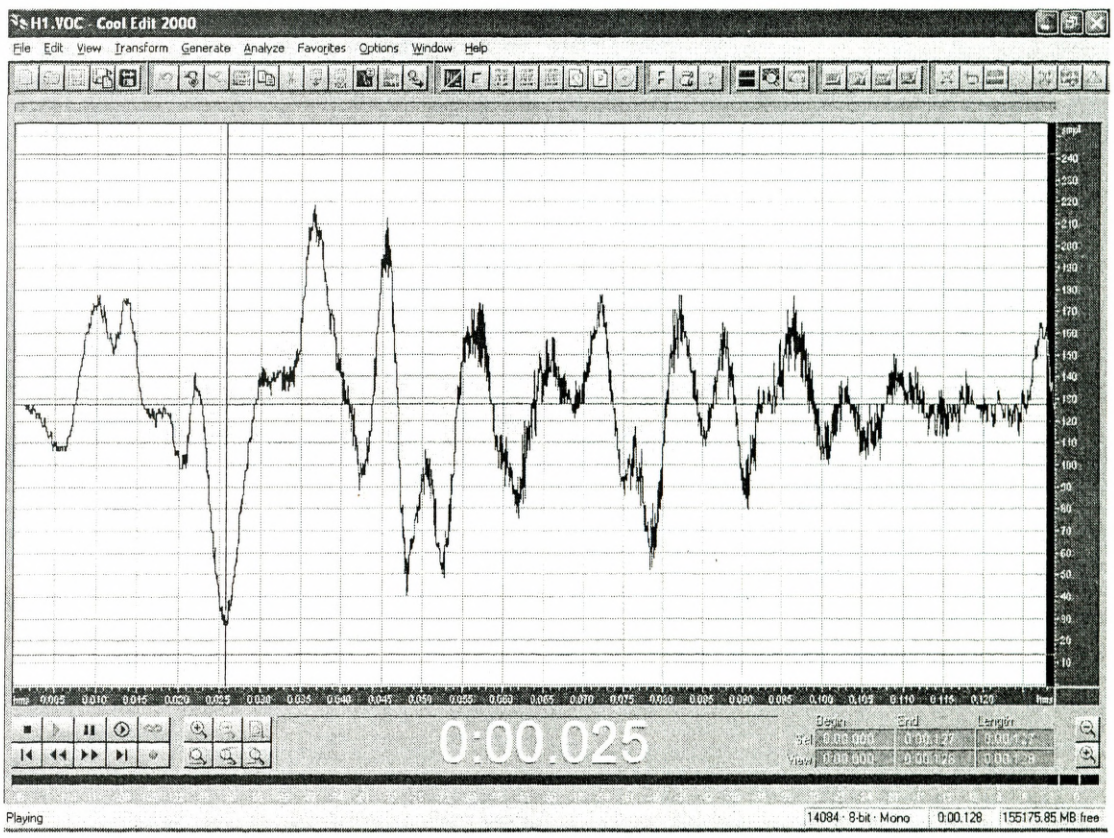
Rys. 8. Ilustracja głoski E



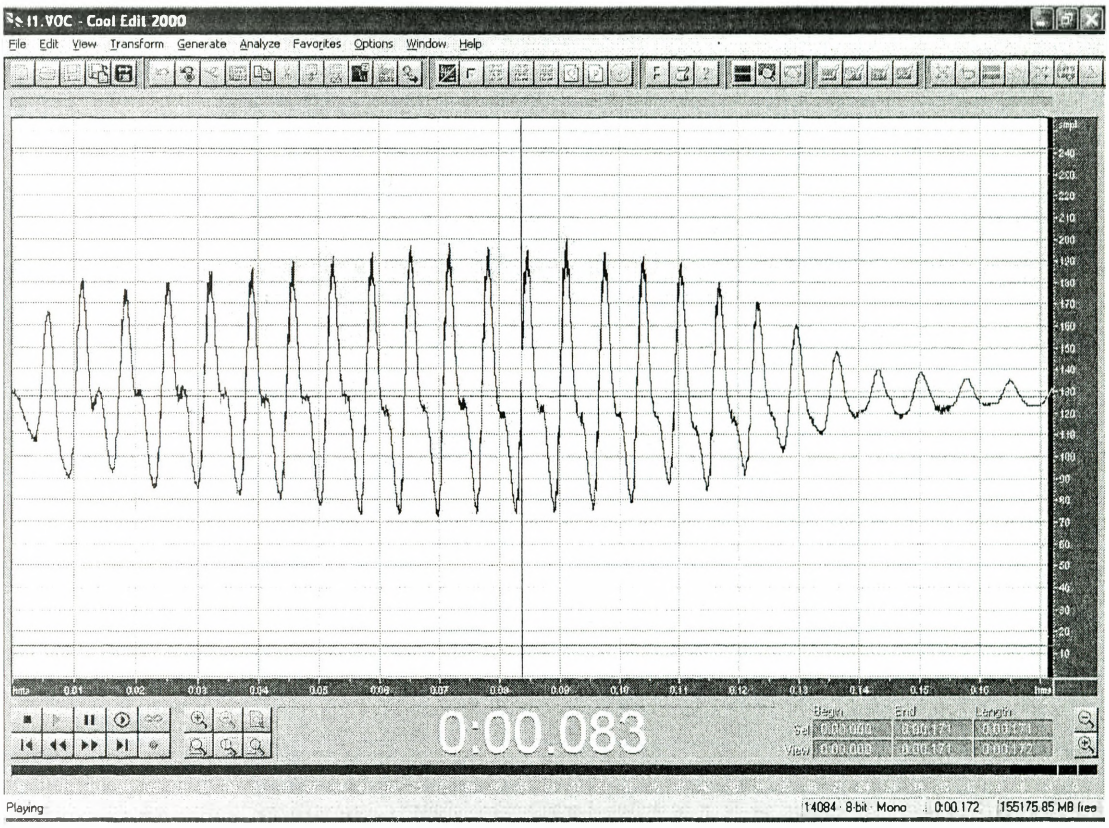
Rys. 9. Ilustracja głoski F



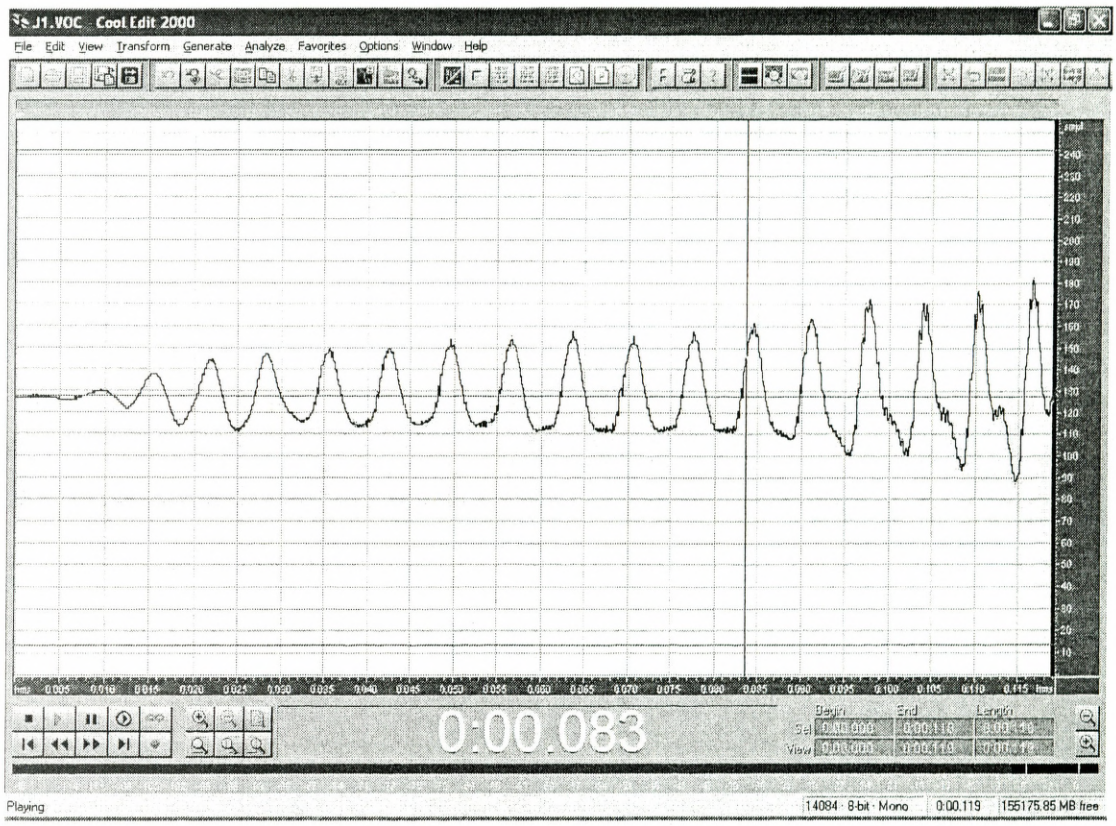
Rys. 10. Ilustracja głoski G



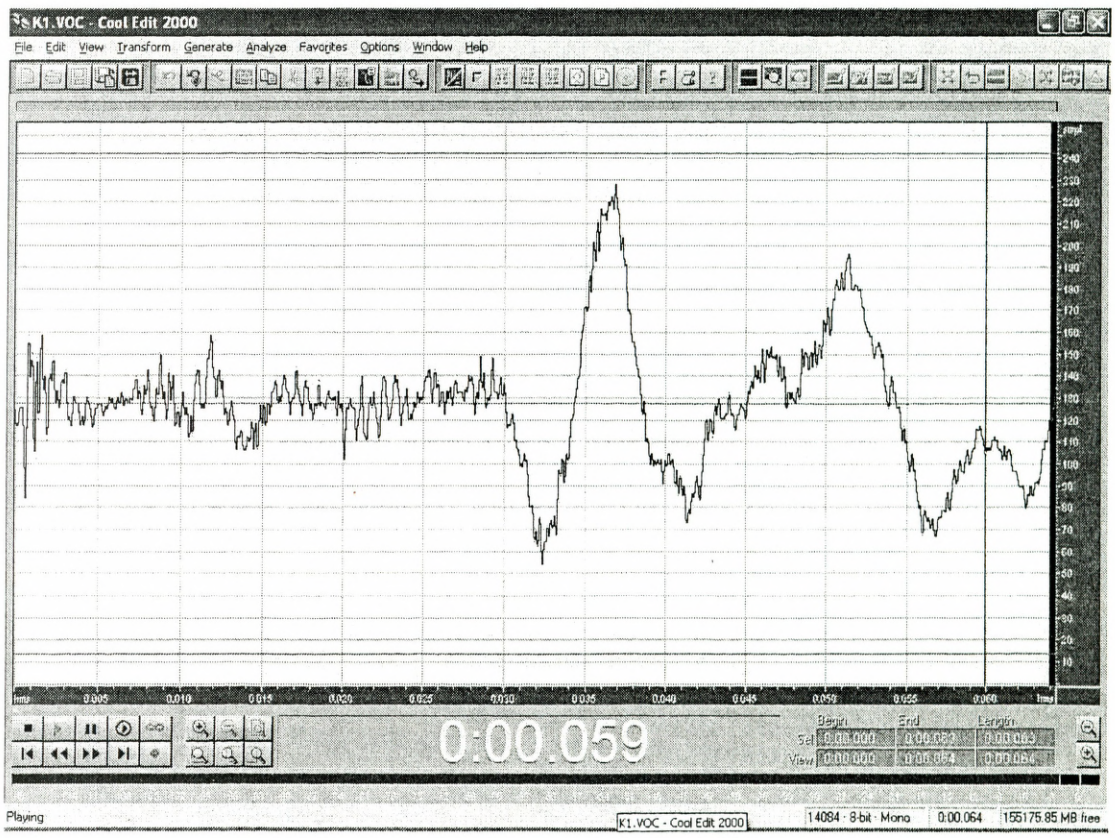
Rys. 11. Ilustracja głoski H



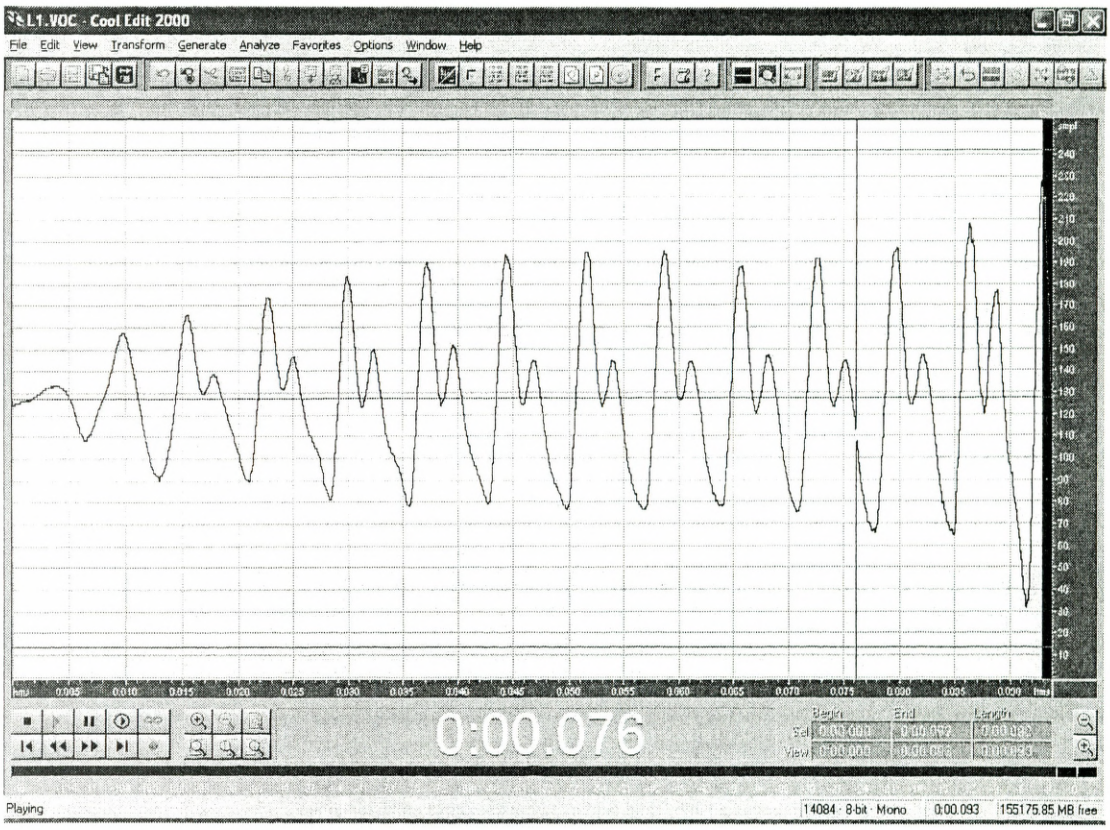
Rys. 12. Ilustracja głoski I



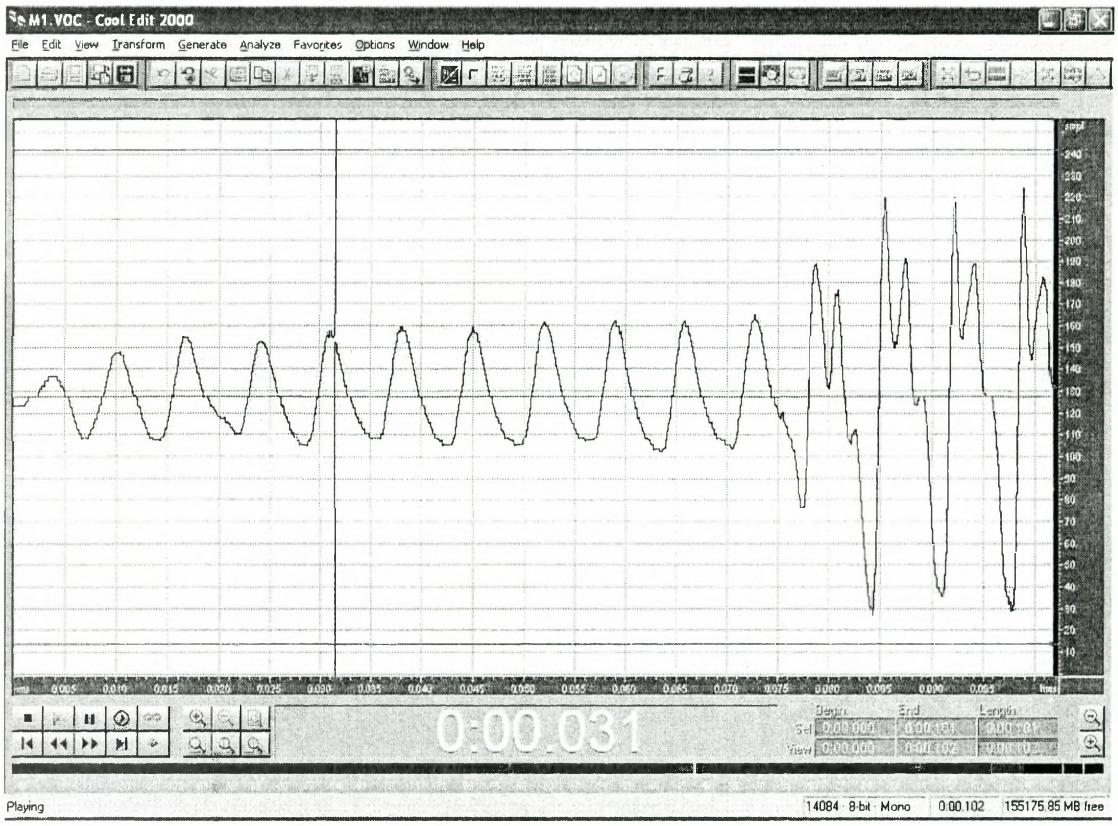
Rys. 13. Ilustracja głoski J



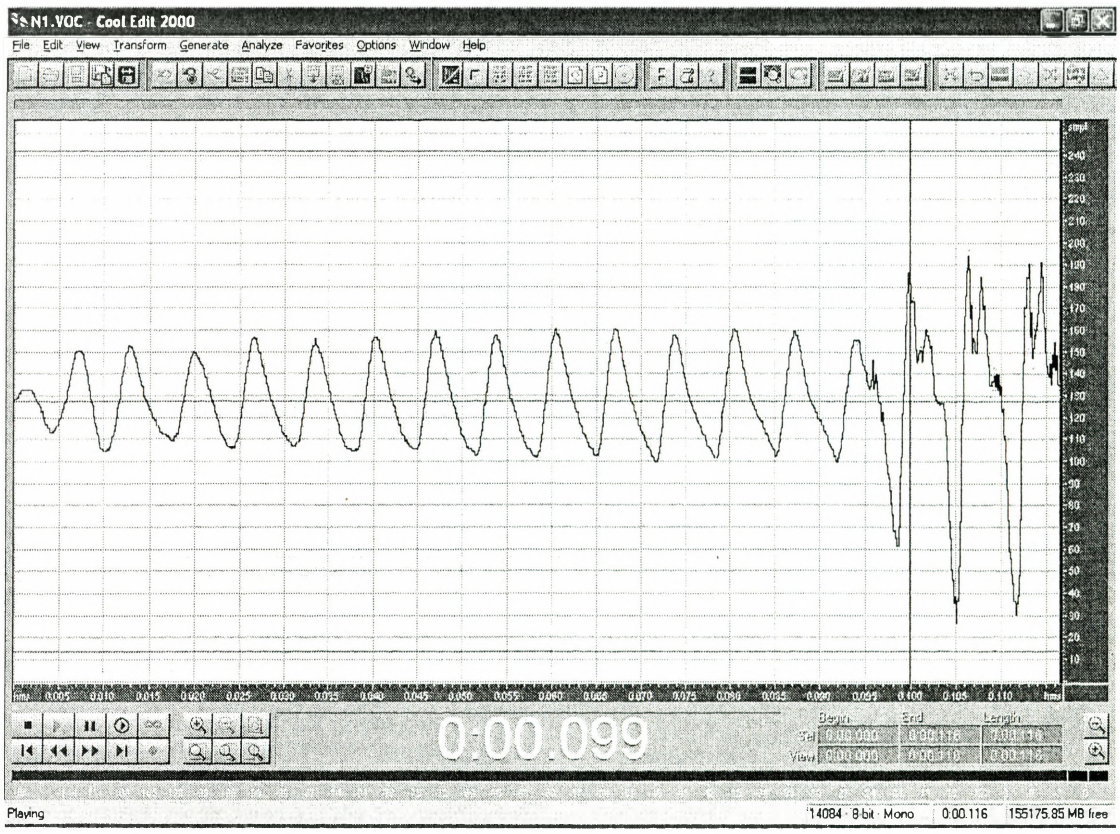
Rys. 14. Ilustracja głoski K



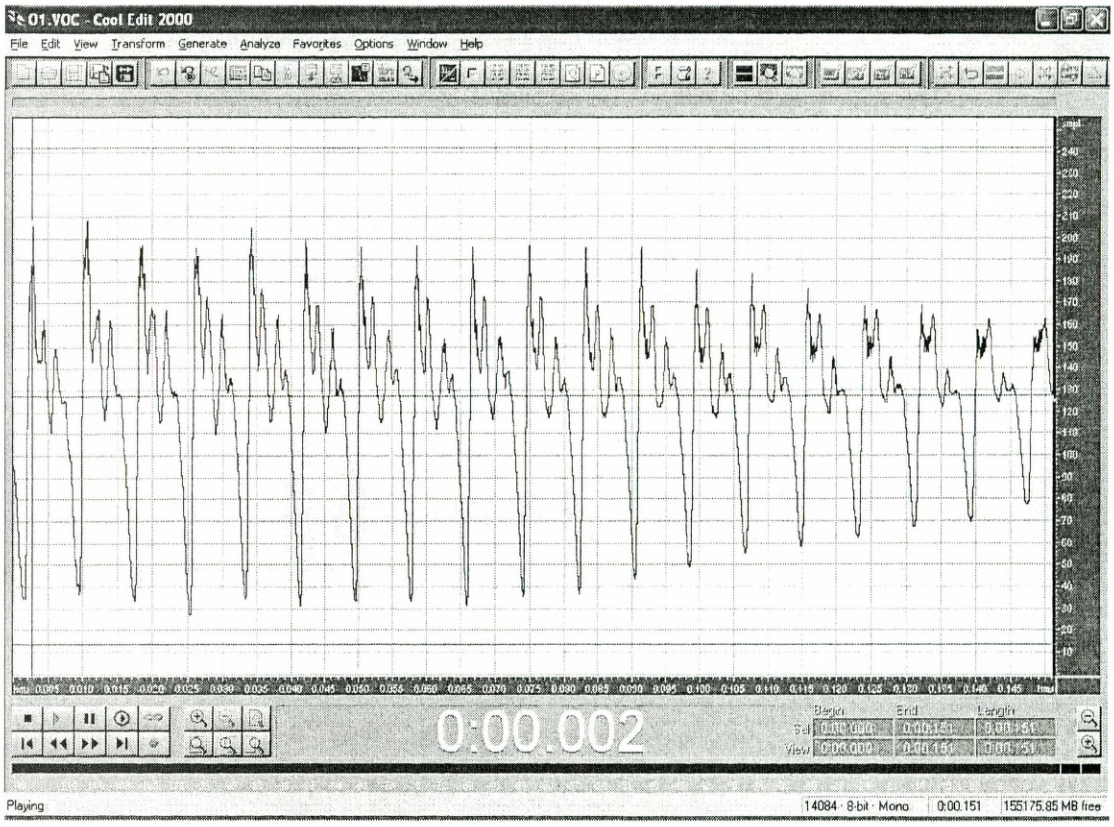
Rys. 15. Ilustracja głoski L



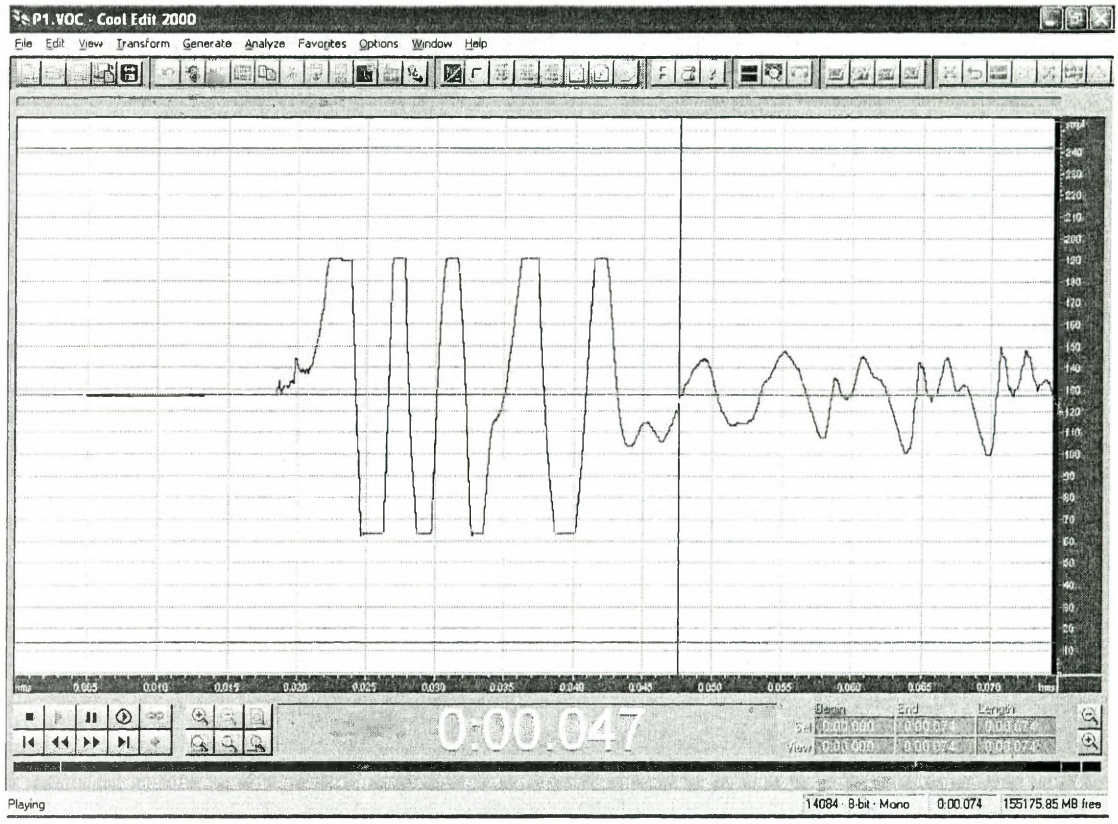
Rys. 16. Ilustracja głoski M



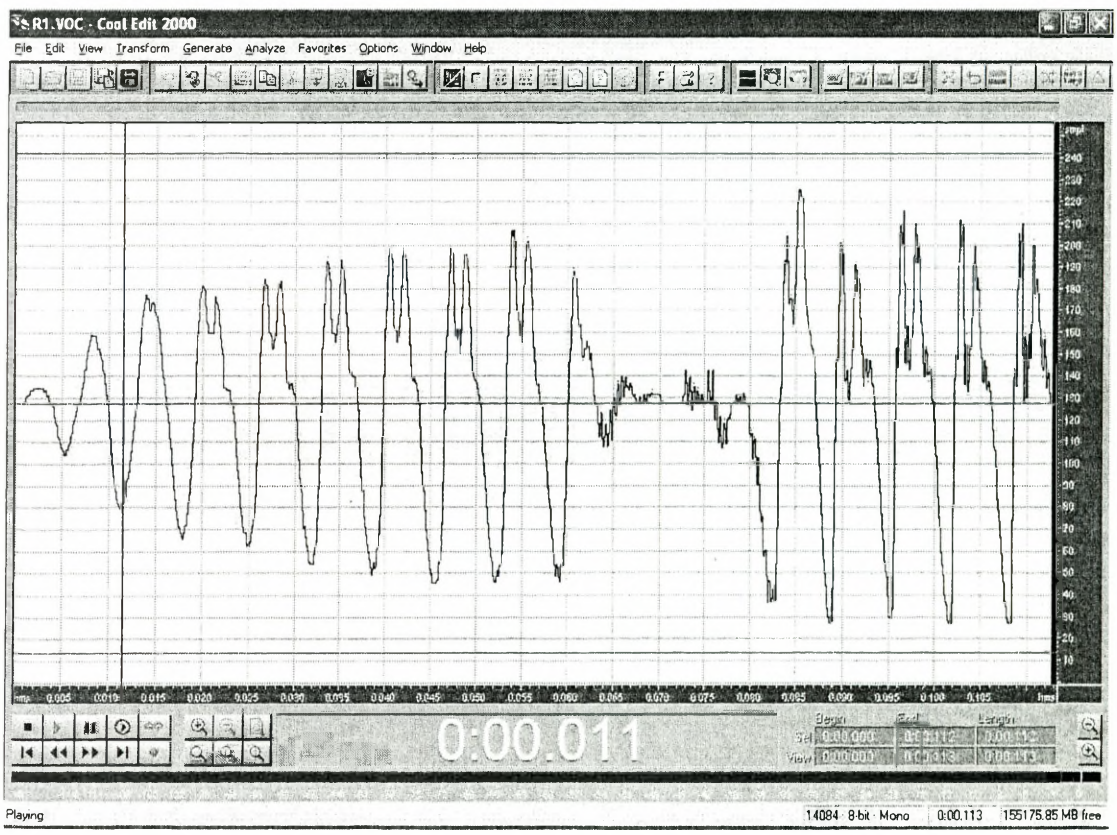
Rys. 17. Ilustracja głoski N



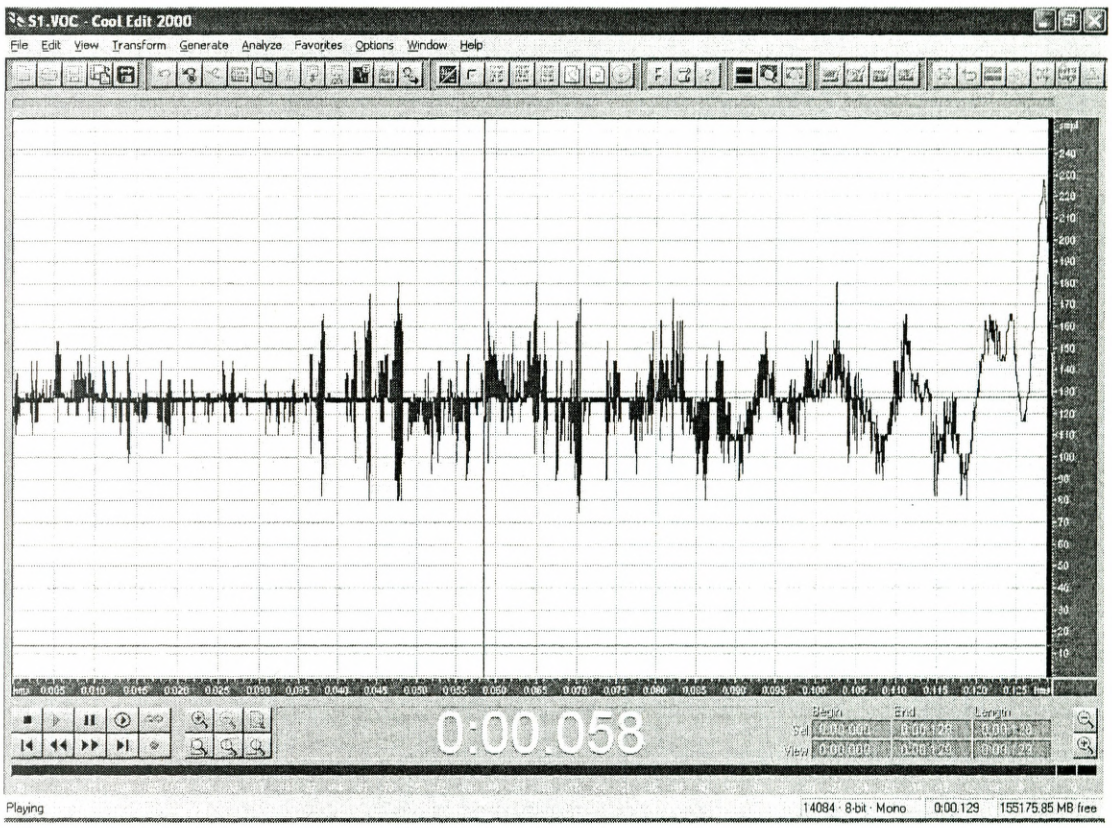
Rys. 6. Ilustracja głoski O



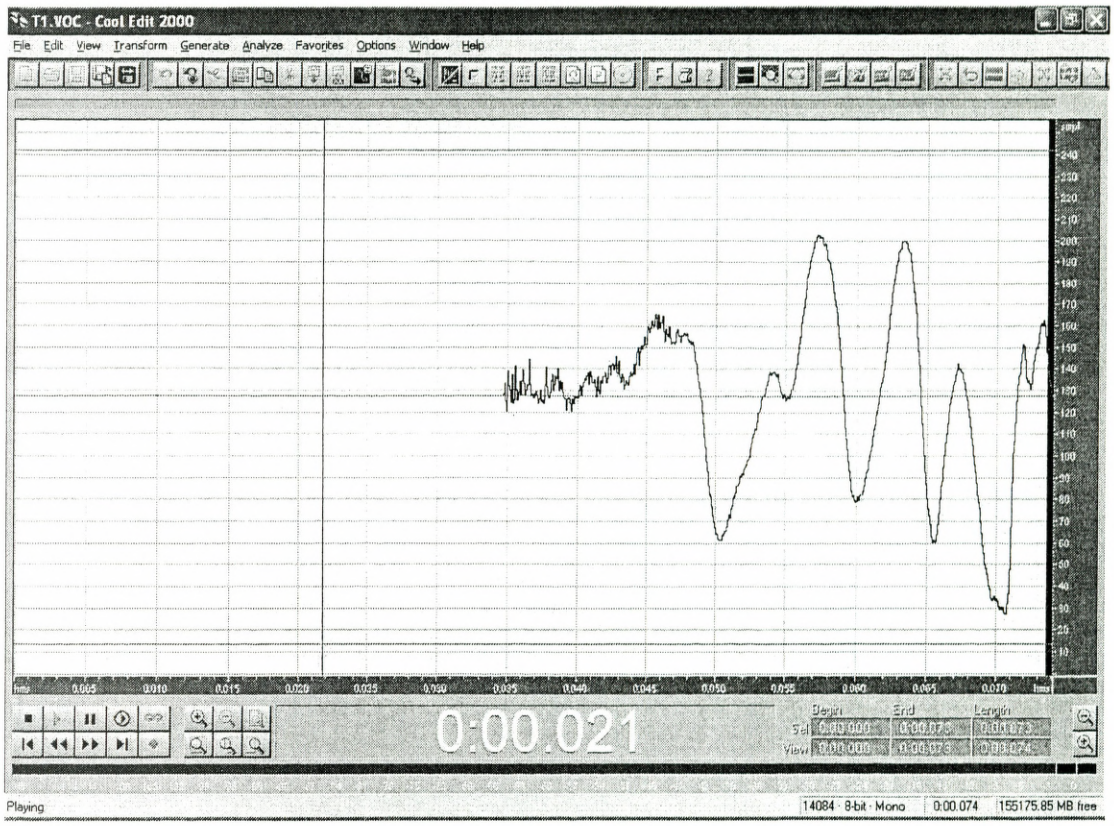
Rys. 18. Ilustracja głoski P



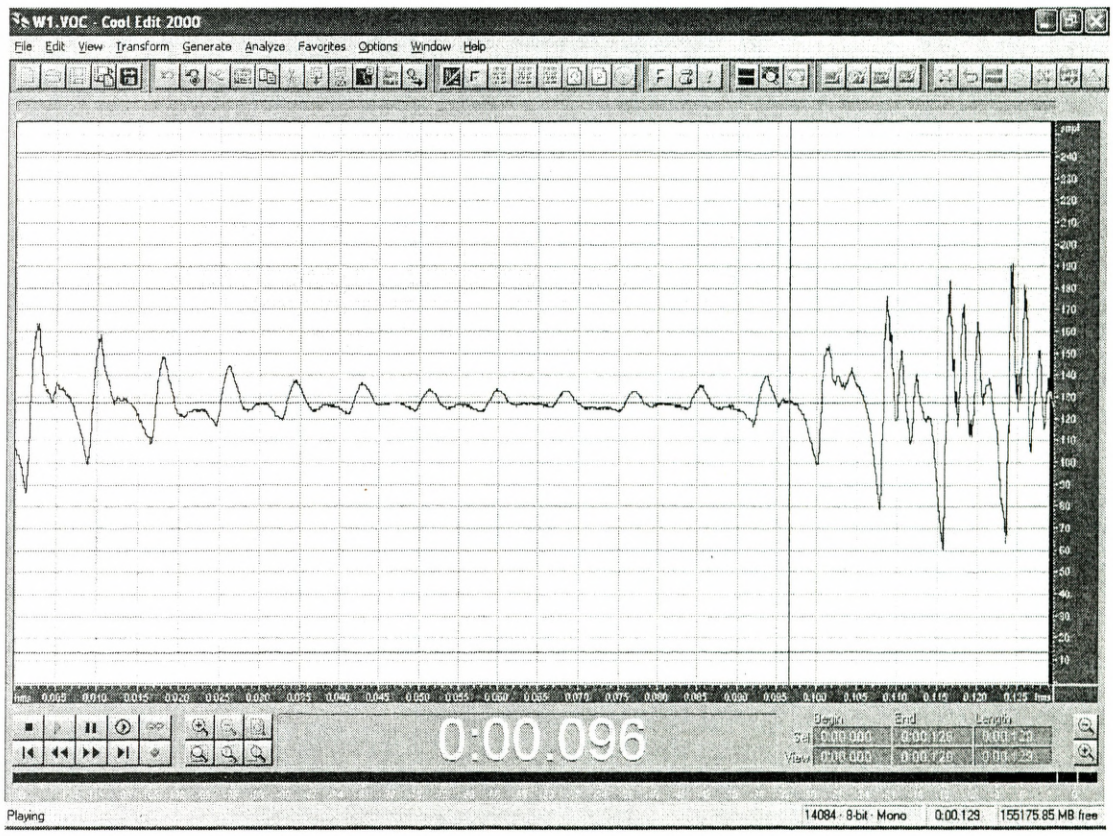
Rys. 19. Ilustracja głoski R



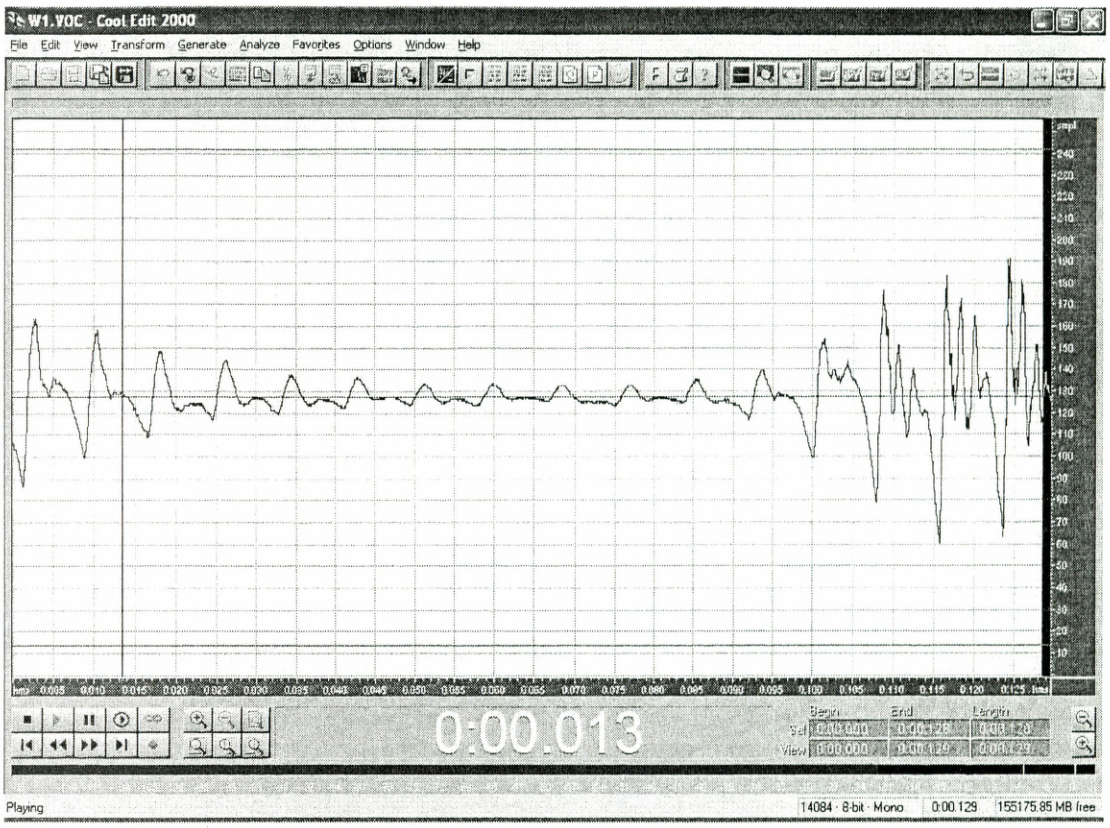
Rys. 20. Ilustracja głoski S



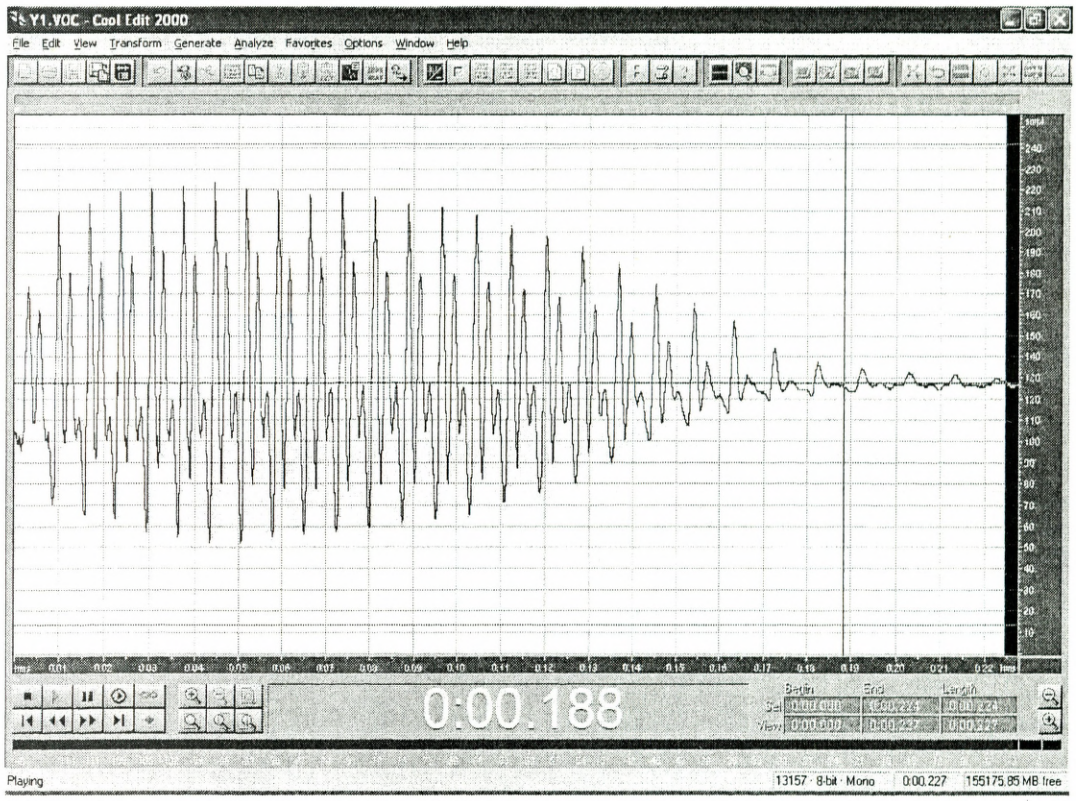
Rys. 21. Ilustracja głoski T



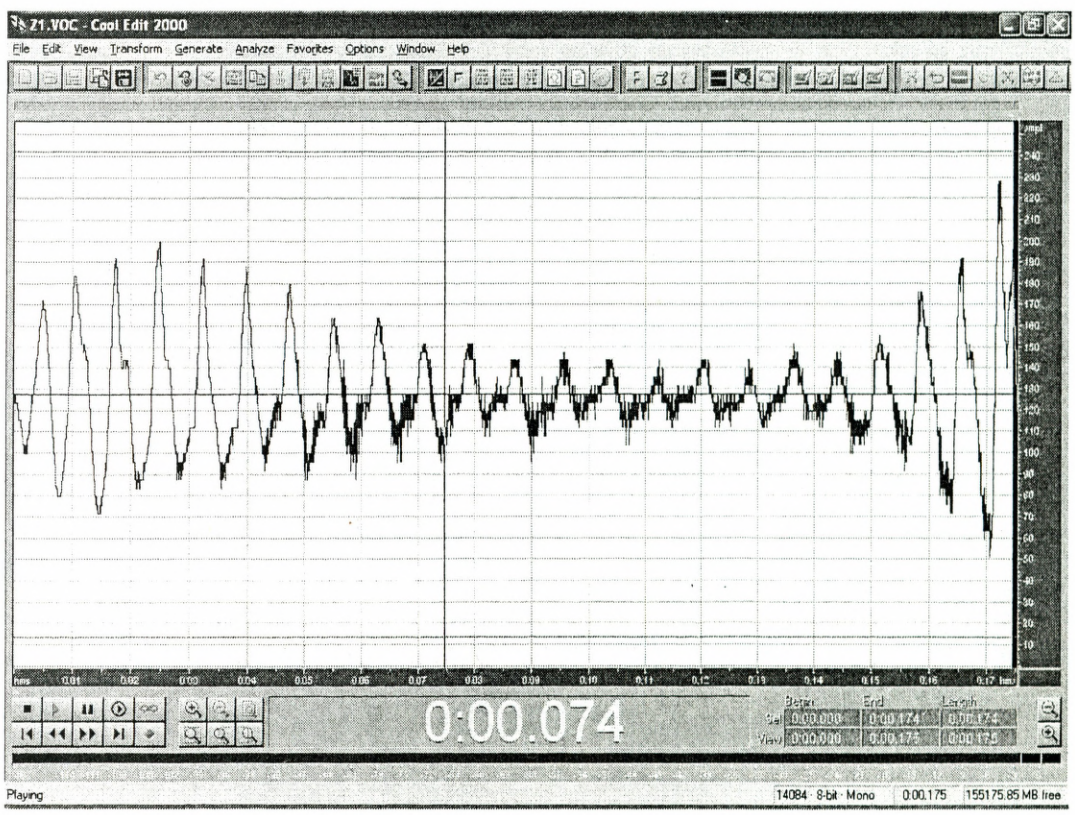
Rys. 22. Ilustracja głoski U



Rys. 23. Ilustracja głoski W



Rys. 24. Ilustracja głoski Y



Rys. 25. Ilustracja głoski Z

4.2. Budowa komunikatu syntezowanego

W podrozdziale tym zaprezentowano najistotniejsze elementy kodu źródłowego, gdzie podjęto próby6 składania komunikatów dźwiękowych z głosek (tzw. komunikaty syntezowane).

```
/* budowa syntezowanego komunikatu */

        if (nr_przycisku==1 && flaga_ekran_pocz==1 &&
         x>10 && x<210 && y>420 && y<470)
        {
                HideMouse();

klawisz_wcisn(10,420,210,470,kolor,"synteza",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora();} while (nr_przycisku!=0);
                HideMouse();
                klawisz(10,420,210,470,kolor,"Syntezer",tekst_przyc);
                flaga_ekran_pocz=0;
                ekran();
                ShowMouse();
                flaga_synt=1;
        }

        /* obsluga Konfiguracja */

        if (nr_przycisku==1 && flaga_ekran_pocz==1 &&
                x>220 && x<420 && y>420 && y<470)
        {
                HideMouse();
```

```
;
        ShowMouse();
        do { pozycja_kursora(); } while (nr_przycisku!=0);
        HideMouse();
        klawisz(220,420,420,470,kolor,"Konfiguracja",tekst_przyc);
        ShowMouse();
        konfiguracja();
    }

    /* obsluga Koniec */

    if (nr_przycisku==1 && flaga_ekran_pocz==1 &&
        x>430 && x<630 && y>420 && y<470)
    {
        HideMouse();

klawisz_wcisn(430,420,630,470,kolor,"Koniec",tekst_przyc_wcisn);
        ShowMouse();
        do { pozycja_kursora(); } while (nr_przycisku!=0);
        HideMouse();
        klawisz(430,420,630,470,kolor,"Koniec",tekst_przyc);
        ShowMouse();
        closegraph();
        clrscr();
        exit(0);
    }

    /* obsluga Powiedz */
```

```
        if (nr_przycisku==1 && flaga_play==1 && flaga_ekran_pocz==0
&&
                (flaga_tekst==1 || flaga_edit==1) &&
                x>10 && x<70 && y>450 && y<470)
        {
                HideMouse();

    klawisz_wcisn(10,450,70,470,kolor,"Powiedz",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                powiedz();
                HideMouse();
                klawisz(10,450,70,470,kolor,"Powiedz",tekst_przyc);
                ShowMouse();
        }

        /* obsluga Przerwa literowa << */

        if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                x>10 && x<30 && y>330 && y<350)
        {
                HideMouse();

    klawisz_wcisn(10,330,30,350,kolor,"<<",tekst_przyc_wcisn);
                ShowMouse();
                do
                {
                        pozycja_kursora();
```

```
                    if (nr_przycisku==1) delay(300);
                    if (letter_delay>0) letter_delay--;
                    HideMouse();
                    setcolor(kolor);
                    bar(40,332,180,348);
                    setcolor(kolor_opisu);
                    outtextxy(110-
strlen(itoa(letter_delay,string,10))*4,337,itoa(letter_delay,string,10));
                    ShowMouse();
                } while (nr_przycisku!=0);
                HideMouse();
                klawisz(10,330,30,350,kolor,"<<",tekst_przyc);
                ShowMouse();

        }

        /* obsługa Przerwa literowa */

        if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                x>190 && x<210 && y>330 && y<350)
        {
                HideMouse();

    klawisz_wcisn(190,330,210,350,kolor,">>",tekst_przyc_wcisn);
    ShowMouse();
                do
                {
                        pozycja_kursora();
```

```
                    if (nr_przycisku==1) delay(300);
                    if (letter_delay<1000) letter_delay++;
                    HideMouse();
                    setcolor(kolor);
                    bar(40,332,180,348);
                    setcolor(kolor_opisu);
                    outtextxy(110-
strlen(itoa(letter_delay,string,10))*4,337,itoa(letter_delay,string,10));
                    ShowMouse();
                } while (nr_przycisku!=0);
                HideMouse();
                klawisz(190,330,210,350,kolor,">>",tekst_przyc);
                ShowMouse();

        }

        /* obsluga Przerwa sylabowa << */

        if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                x>10 && x<30 && y>375 && y<395)
        {
                HideMouse();

    klawisz_wcisn(10,375,30,395,kolor,"<<",tekst_przyc_wcisn);
                ShowMouse();
                do
                {
```

```
                    pozycja_kursora();
                    if (nr_przycisku==1) delay(300);
                    if (przerw_sylab>0) przerw_sylab--;
                    HideMouse();
                    setcolor(kolor);
                    bar(40,377,180,393);
                    setcolor(kolor_opisu);
                    outtextxy(110-
strlen(itoa(przerw_sylab,string,10))*4,382,itoa(przerw_sylab,string,10));
                    ShowMouse();
                } while (nr_przycisku!=0);
                HideMouse();
                klawisz(10,375,30,395,kolor,"<<",tekst_przyc);
                ShowMouse();

            }

            /* obsługa Przerwa sylabowa */

            if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                x>190 && x<210 && y>375 && y<395)
            {
                HideMouse();

        klawisz_wcisn(190,375,210,395,kolor,">>",tekst_przyc_wcisn);
                ShowMouse();
                do
                {
```

```
                pozycja_kursora();
                if (nr_przycisku==1) delay(300);
                if (przerw_sylab<1000) przerw_sylab++;
                HideMouse();
                setcolor(kolor);
                bar(40,377,180,393);
                setcolor(kolor_opisu);
                outtextxy(110-
strlen(itoa(przerw_sylab,string,10))*4,382,itoa(przerw_sylab,string,10));
                ShowMouse();
            } while (nr_przycisku!=0);
            HideMouse();
            klawisz(190,375,210,395,kolor,">>",tekst_przyc);
            ShowMouse();
        }

        /* obsługa Przerwa wyrazowa << */

        if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
            x>10 && x<30 && y>420 && y<440)
        {
            HideMouse();

    klawisz_wcisn(10,420,30,440,kolor,"<<",tekst_przyc_wcisn);
            ShowMouse();
            do
            {
                pozycja_kursora();
```

```
                    if (nr_przycisku==1)
                    {
                        delay(300);
                        word_delay--;
                    }
                    if (nr_przycisku==2) word_delay=word_delay-10;
                    if (word_delay<=0) word_delay=0;
                    HideMouse();
                    setcolor(kolor);
                    bar(40,422,180,438);
                    setcolor(kolor_opisu);
                    outtextxy(110-
strlen(itoa(word_delay,string,10))*4,427,itoa(word_delay,string,10));
                    ShowMouse();
                } while (nr_przycisku!=0);
                HideMouse();
                klawisz(10,420,30,440,kolor,"<<",tekst_przyc);
                ShowMouse();

            }

            /* obsługa Przerwa wyrazowa >> */

            if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                x>190 && x<210 && y>420 && y<440)
            {
                HideMouse();
```

```
klawisz_wcisn(190,420,210,440,kolor,">>",tekst_przyc_wcisn);
                ShowMouse();
                do
                {
                        pozycja_kursora();
                        if (nr_przycisku==1)
                        {
                                delay(300);
                                word_delay++;
                        }
                        if (nr_przycisku==2) word_delay=word_delay+10;
                        if (word_delay>=1000) word_delay=1000;
                        HideMouse();
                        setcolor(kolor);
                        bar(40,422,180,438);
                        setcolor(kolor_opisu);
                        outtextxy(110-
strlen(itoa(word_delay,string,10))*4,427,itoa(word_delay,string,10));
                        ShowMouse();
                } while (nr_przycisku!=0);
                HideMouse();
                klawisz(190,420,210,440,kolor,">>",tekst_przyc);
                ShowMouse();
        }

        /* obsługa Próbkowanie << */
```

```
            if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
                  x>220 && x<240 && y>330 && y<350)
            {
                  HideMouse();

      klawisz_wcisn(220,330,240,350,kolor,"<<",tekst_przyc_wcisn);
                  ShowMouse();
                  do
                  {
                        pozycja_kursora();
                        if (nr_przycisku==1)
                        {
                              delay(300);
                              sample_rate--;
                        }
                        if (nr_przycisku==2) sample_rate=sample_rate-10;
                        if (sample_rate<=1) sample_rate=1;
                        HideMouse();
                        setcolor(kolor);
                        bar(260,332,380,348);
                        setcolor(kolor_opisu);
                        outtextxy(320-
strlen(itoa(sample_rate,string,10))*4,337,itoa(sample_rate,string,10));
                        ShowMouse();
                  } while (nr_przycisku!=0);
                  HideMouse();
                  klawisz(220,330,240,350,kolor,"<<",tekst_przyc);
                  ShowMouse();
```

```
        }

        /* obsługa Próbkowanie >> */

        if ((nr_przycisku==1 || nr_przycisku==2) && flaga_ekran_pocz==0
&&
            x>400 && x<420 && y>330 && y<350)
        {
            HideMouse();

    klawisz_wcisn(400,330,420,350,kolor,">>",tekst_przyc_wcisn);
            ShowMouse();
            do
            {
                pozycja_kursora();
                if (nr_przycisku==1)
                {
                    delay(300);
                    sample_rate++;
                }
                if (nr_przycisku==2) sample_rate=sample_rate+10;
                if (sample_rate>=22000) sample_rate=22000;
                HideMouse();
                setcolor(kolor);
                bar(260,332,380,348);
                setcolor(kolor_opisu);
                outtextxy(320-
strlen(itoa(sample_rate,string,10))*4,337,itoa(sample_rate,string,10));
```

```
                ShowMouse();
            } while (nr_przycisku!=0);
            HideMouse();
            klawisz(400,330,420,350,kolor,">>",tekst_przyc);
            ShowMouse();
        }

        /* obsługa Tekst */

        if (nr_przycisku==1 && flaga_tekst==0 && flaga_ekran_pocz==0
&&
            x>430 && x<490 && y>450 && y<470)
        {
            HideMouse();

    klawisz_wcisn(430,450,490,470,kolor,"Tekst",tekst_przyc_wcisn);
            klawisz(500,450,560,470,kolor,"Edycja",tekst_przyc);
            ShowMouse();
            do { pozycja_kursora(); } while (nr_przycisku!=0);
            flaga_tekst=1;
            flaga_edit=0;
        }

        if (nr_przycisku==1 && flaga_tekst==1 && flaga_ekran_pocz==0
&&
            x>430 && x<490 && y>450 && y<470)
        {
            HideMouse();
            klawisz(430,450,490,470,kolor,"Tekst",tekst_przyc);
```

```
                flaga_tekst=0;
                pocz_pliku=1;
                koniec_pliku=0;
                poz_litery_x=0;
                poz_litery_y=0;
                fseek(temp_in,0,SEEK_SET);
                ile_stron=0;
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
        }


        /* obsluga Tekst << */

        if (nr_przycisku==1 && pocz_pliku==0 &&
           flaga_auto==0  &&  flaga_play==1  &&  flaga_ekran_pocz==0
&&
           x>10 && x<110 && y>210 && y<230)
        {
                HideMouse();

    klawisz_wcisn(10,210,110,230,kolor,"<<",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                HideMouse();
                klawisz(10,210,110,230,kolor,"<<",tekst_przyc);
                setfillstyle(SOLID_FILL,BLACK);
                bar(12,62,628,198);
                poz_litery_y=0;
```

```
                poz_litery_x=0;
                ile_stron--;
                if (ile_stron==0) pocz_pliku=1;
                fseek(temp_in,poz_w_pliku[ile_stron],SEEK_SET);
                do
                {
                        wypisz_tekst(20,70);

                } while (poz_litery_y<=11 && litery[0]!=EOF);
                ShowMouse();
                koniec_pliku=0;
        }


        /* obsluga Tekst >> */

        if (nr_przycisku==1 && koniec_pliku==0 &&
           flaga_play==1  &&  flaga_auto==0  &&  flaga_ekran_pocz==0
&&
           x>530 && x<630 && y>210 && y<230)
        {
                HideMouse();

    klawisz_wcisn(530,210,630,230,kolor,">>",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                HideMouse();
                klawisz(530,210,630,230,kolor,">>",tekst_przyc);
                setfillstyle(SOLID_FILL,BLACK);
```

```
                bar(12,62,628,198);
                poz_litery_y=0;
                poz_litery_x=0;
                ile_stron++;
                fseek(temp_in,poz_w_pliku[ile_stron],SEEK_SET);
                do
                {
                        wypisz_tekst(20,70);

                } while (poz_litery_y<=11 && litery[0]!=EOF);
                ShowMouse();
                pocz_pliku=0;
        }


        /* obsluga Automatyczne przewijanie */

        if (nr_przycisku==1 && flaga_auto==0 &&
                        flaga_ekran_pocz==0 &&
                        x>115 && x<525 && y>210 && y<230)
        {
                HideMouse();
                klawisz_wcisn(115,210,525,230,kolor,"Automatyczne
przewijanie",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                flaga_auto=1;
        }
```

```
        if (nr_przycisku==1 && flaga_auto==1 &&
                    x>115 && x<525 && y>210 && y<230)
        {
            HideMouse();
            klawisz(115,210,525,230,kolor,"Automatyczne
przewijanie",tekst_przyc);
            ShowMouse();
            do { pozycja_kursora(); } while (nr_przycisku!=0);
            flaga_auto=0;
        }

        /* obsluga Edycja */

        if (nr_przycisku==1 && flaga_ekran_pocz==0 && flaga_edit==0
&&
                    x>500 && x<560 && y>450 && y<470)
        {
            HideMouse();

    klawisz_wcisn(500,450,560,470,kolor,"Edycja",tekst_przyc_wcisn);
            klawisz(430,450,490,470,kolor,"Tekst",tekst_przyc);
            do { pozycja_kursora(); } while (nr_przycisku!=0);
            setfillstyle(SOLID_FILL,BLACK);
            bar(22,272,618,288);
            edycja();
            ShowMouse();
            flaga_edit=1;
            flaga_tekst=0;
            flaga_play=1;
```

```
        }

        if (nr_przycisku==1 && flaga_ekran_pocz==0 && flaga_edit==1
&&
                x>500 && x<560 && y>450 && y<470)
        {
            do { pozycja_kursora(); } while (nr_przycisku!=0);
            HideMouse();
            setfillstyle(SOLID_FILL,BLACK);
            bar(22,272,618,288);
            klawisz(500,450,560,470,kolor,"Edycja",tekst_przyc);
            ShowMouse();
            flaga_edit=0;
            flaga_play=0;
        }

        /* obsluga wskaznikow zbiorow */

        for (i=0;i<8;i++)
        {
            if (nr_przycisku==1 && flaga_ekran_pocz==0 &&
                x>440 && x<550 && y>(339+12*i) &&
y<(347+12*i))
            {
                ktory_zbior=0;
                ktory_zbior=i+wsk_zbiorow;
                HideMouse();
                for (j=0;j<8;j++)
klawisz(440,339+12*j,448,347+12*j,kolor," ",tekst_przyc);
```

```
                    if                          (ktory_zbior<ile_zbiorow)
klawisz_wcisn(440,339+12*i,448,347+12*i,LIGHTGREEN,"
",tekst_przyc_wcisn);
                    if (flaga_tekst==1)
                    {
                        setfillstyle(SOLID_FILL,BLACK);
                        bar(12,62,628,198);
                    }
                    ShowMouse();
                    do { pozycja_kursora(); } while (nr_przycisku!=0);
                    flaga_cancel=1;
                    pocz_pliku=1;
                    koniec_pliku=0;
                    flaga_wybrany=1;
                    flaga_play=0;
                }
            }

            /* obsluga Pg Up */

            if (nr_przycisku==1 && wsk_zbiorow>0 &&
flaga_ekran_pocz==0 &&
                    x>560 && x<620 && y>388 && y<408)
            {
                HideMouse();

    klawisz_wcisn(560,388,620,408,kolor,"PgUp",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
```

```
                HideMouse();
                klawisz(560,388,620,408,kolor,"PgUp",tekst_przyc);
                wsk_zbiorow=wsk_zbiorow-8;
                for                                                    (j=0;j<8;j++)
klawisz(440,339+12*j,448,347+12*j,kolor," ",tekst_przyc);
                setfillstyle(SOLID_FILL,kolor);
                bar(450,338,549,436);
                setcolor(BLACK);
                for (i=0;i<8;i++)
                {

      outtextxy(455,339+12*i,tab_zbiorow[wsk_zbiorow+i]);
                }
                ShowMouse();
                flaga_cancel=0;
                flaga_wybrany=0;
          }

          /* obsluga Pg Dn */

          if (nr_przycisku==1 && wsk_zbiorow<(ile_zbiorow-10) &&
                    flaga_ekran_pocz==0 &&
                    x>560 && x<620 && y>413 && y<433)
          {
                HideMouse();

     klawisz_wcisn(560,413,620,433,kolor,"PgDn",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
```

```
                HideMouse();
                klawisz(560,413,620,433,kolor,"PgDn",tekst_przyc);
                wsk_zbiorow=wsk_zbiorow+8;
                for                                          (j=0;j<8;j++)
klawisz(440,339+12*j,448,347+12*j,kolor," ",tekst_przyc);
                setfillstyle(SOLID_FILL,kolor);
                bar(450,338,549,436);
                setcolor(BLACK);
                for (i=0;i<8;i++)
                {

      outtextxy(455,339+12*i,tab_zbiorow[wsk_zbiorow+i]);
                }
                ShowMouse();
                flaga_cancel=0;
                flaga_wybrany=0;
          }

          /* obsluga Ok */

          if (nr_przycisku==1 && flaga_cancel==1 && flaga_wybrany==1
&&
                    flaga_ekran_pocz==0 &&
                    x>560 && x<620 && y>338 && y<358)
          {
                HideMouse();

     klawisz_wcisn(560,338,620,358,kolor,"Ok",tekst_przyc_wcisn);
                ShowMouse();
```

```
        do { pozycja_kursora(); } while (nr_przycisku!=0);
        HideMouse();
        klawisz(560,338,620,358,kolor,"Ok",tekst_przyc);
        strcpy(nazwa_in,tab_zbiorow[ktory_zbior]);
        fclose(in);
        fclose(temp_in);
        kopiuj_tekst();
        fseek(temp_in,0,SEEK_SET);
        flaga_play=1;
        pocz_pliku=1;
        koniec_pliku=0;
        setfillstyle(SOLID_FILL,BLACK);
        bar(12,62,628,198);
        do
        {
                wypisz_tekst(20,70);
                if (litery[0]==EOF) koniec_pliku=1;
        } while (poz_litery_y<=11 && litery[0]!=EOF);
        ile_stron=0;
        poz_litery_y=0;
        poz_litery_x=0;
        ShowMouse();
}

/* obsluga Anuluj */

if (nr_przycisku==1 && flaga_wybrany==1 &&
                flaga_ekran_pocz==0 &&
                x>560 && x<620 && y>363 && y<383)
```

```
        {
                HideMouse();

    klawisz_wcisn(560,363,620,383,kolor,"Anuluj",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                HideMouse();
                klawisz(560,363,620,383,kolor,"Anuluj",tekst_przyc);
                for                                          (j=0;j<8;j++)
klawisz(440,339+12*j,448,347+12*j,kolor," ",tekst_przyc);
                if (flaga_tekst==1)
                {
                        setfillstyle(SOLID_FILL,BLACK);
                        bar(12,62,628,198);
                }
                ShowMouse();
                flaga_play=0;
                flaga_cancel=0;
                flaga_wybrany=0;
                fclose(temp_in);
                fclose(in);
        }

        /* obsluga Koniec */

        if (nr_przycisku==1 && flaga_ekran_pocz==0 &&
                        x>570 && x<630 && y>450 && y<470)
        {
                HideMouse();
```

```
        klawisz_wcisn(570,450,630,470,kolor,"Koniec",tekst_przyc_wcisn);
                ShowMouse();
                do { pozycja_kursora(); } while (nr_przycisku!=0);
                HideMouse();
                klawisz(570,450,630,470,kolor,"Koniec",tekst_przyc);
                cleardevice();
                /*restore_screen();*/
                klawisz(430,420,630,470,kolor,"Koniec",tekst_przyc);
                ShowMouse();
                flaga_ekran_pocz=1;

            }

        }  while (flaga_exit==0);

        ctvm_speaker(0);
        closegraph();
        fclose(in);
        fclose(temp_in);
        clrscr();
        return 0;
}

void powiedz(void)
{
int     probkowanie;

        probkowanie=ceil(256-1000000/sample_rate);
```

```
for (i=0;i<60;i++)
{

poke(FP_SEG(lpVoiceBuf[i]),FP_OFF(lpVoiceBuf[i])+30,probkowanie);
}

if (flaga_tekst==1)
{
        fseek(in,poz_w_pliku[ile_stron],SEEK_SET);

        z4=z3=' ';
        z2=male_lit(fgetc(in));
        z1=male_lit(fgetc(in));
        poz_litery_y=0;
        do
        {
                if (wsk==60)
                {
                        licznik_akc=0;
                        spr_licznik_sylab=0;
                        akcent();
                }

                z4=z3;
                z3=z2;
                z2=z1;
                z1=male_lit(fgetc(in));
                licznik_akc++;
```

```
konw_alof();

if (podwojny==1)
{
        z4=z3;
        z3=z2;
        z2=z1;
        z1=male_lit(fgetc(in));
        licznik_akc++;
        podwojny=0;
}

akcent_sylaby();

if (z3=='\n') poz_litery_y++;

if (poz_litery_y==12 && flaga_auto==1)
{
        setfillstyle(SOLID_FILL,BLACK);
        bar(12,62,628,198);
        pocz_pliku=0;
        poz_litery_y=0;
        poz_litery_x=0;
        ile_stron++;
        fseek(temp_in,poz_w_pliku[ile_stron],SEEK_SET);
        do
        {
                wypisz_tekst(20,70);
        } while (poz_litery_y<=11 && litery[0]!=EOF);
```

```
                poz_litery_y=0;
            }

        } while (z2 != EOF && flaga_stop!=1);
        flaga_stop=0;
}
if (flaga_edit==1)
{
        z4=z3=' ';
        z2=male_lit(wyraz[0]);
        z1=male_lit(wyraz[1]);
        litera_wyrazu=2;

        do
        {
            if (wsk==60)
            {
                licznik_akc=0;
                spr_licznik_sylab=0;
                akcent();
            }

            z4=z3;
            z3=z2;
            z2=z1;
            z1=male_lit(wyraz[litera_wyrazu]);
            licznik_akc++;

            konw_alof();
```

```
                if (podwojny==1)
                {
                        litera_wyrazu++;
                        z4=z3;
                        z3=z2;
                        z2=z1;
                        z1=male_lit(wyraz[litera_wyrazu]);
                        licznik_akc++;
                        podwojny=0;
                }
                litera_wyrazu++;

                akcent_sylaby();

        } while (flaga_stop!=1 && litera_wyrazu<dlug_edycji+2);
        flaga_stop=0;
    }
}

void akcent_sylaby(void)
{
    if (licznik_akc==wsk_akcentu)
    {
        switch (wsk)
        {
                case 0  : wsk=50 ;break;
                case 1  : wsk=51 ;break;
                case 2  : wsk=52 ;break;
```

```
                case 13 : wsk=53 ;break;
                case 14 : wsk=54 ;break;
                case 15 : wsk=55 ;break;
                case 22 : wsk=56 ;break;
                case 33 : wsk=57 ;break;
                case 44 : wsk=58 ;break;
                case 47 : wsk=59 ;break;
                default : break;
        }
}
if (wsk!=60)
{
        OutputVoice(lpVoiceBuf[wsk]) ;
        delay(letter_delay);

                {
                wsk_przerw_sylab=1;
                spr_licznik_sylab++;
                if (z3=='i')
                        {
                                wsk_przerw_sylab=0;
                                spr_licznik_sylab--;
                        }
                }
        }
        else wsk_przerw_sylab=0;

        if (wsk_przerw_sylab==1 && spr_licznik_sylab<licznik_sylab)
                delay(przerw_sylab);
```

```
        }

        if (wsk==60 && wsk_temp!=60) delay(word_delay);
                wsk_temp=wsk;
}

void ekran_pokaz(void)
{
        HideMouse();

        cleardevice();
        setfillstyle(SOLID_FILL,kolor);
        bar(0,0,639,479);
                ShowMouse();
}

void info(void)
{
int     zezw_odtw=0;

        flaga_exit=0;
        flaga_play=0;
        flaga_stop=0;
        flaga_pause=0;
        flaga_tekst=0;
        flaga_cancel=0;
        flaga_wybrany=0;
        flaga_synt=0;
        flaga_auto=0;
```

```
        flaga_edit=0;

        ekran_pokaz();
        do
        {
                pozycja_kursora();

                /* obsluga Przebiegi czasowe */

                if (nr_przycisku==1 &&
                        x>10 && x<210 && y>10 && y<30)
                {
                        HideMouse();
                        klawisz_wcisn(10,10,210,30,kolor,"Przebiegi
czasowe",tekst_przyc_wcisn);
                        ShowMouse();
                        do {pozycja_kursora();} while (nr_przycisku!=0);
                        HideMouse();
                        klawisz(10,10,210,30,kolor,"Przebiegi
czasowe",tekst_przyc);
                        ShowMouse();
                        flaga_przeb_czasowe=1;
                        flaga_opis_synt=0;
                        flaga_synt_alof=0;
                        przeb_czasowe();
                }

                /* obsluga Powiedz w Przeb Czas */
```

```
        if (nr_przycisku==1 && flaga_przeb_czasowe==1 &&
            zezw_odtw==1 && x>10 && x<110 && y>450 && y<470)
        {
            HideMouse();

    klawisz_wcisn(10,450,110,470,kolor,"Powiedz",tekst_przyc_wcisn);
            ShowMouse();
            do {pozycja_kursora();} while (nr_przycisku!=0);
            OutputVoice(lpVoiceBuf[kolumna+rzad*25]);
            HideMouse();
            klawisz(10,450,110,470,kolor,"Powiedz",tekst_przyc);
            ShowMouse();
        }

        for (j=0;j<2;j++)
        {
            for (i=0;i<25;i++)
            {
                if (nr_przycisku==1 && flaga_przeb_czasowe==1 &&
                x>10+i*25 && x<25*(i+1)+5 && y>395+j*20 &&
y<390+(j+1)*20)
                {
                    rzad=j;
                    kolumna=i;
                    HideMouse();

    klawisz_wcisn(10+i*25,395+j*20,(i+1)*25+5,390+(j+1)*20,kolor,nazw_
przeb[i+rzad*25],tekst_przyc_wcisn);
                    ShowMouse();
```

```
                    do {pozycja_kursora();} while
(nr_przycisku!=0);
                    HideMouse();
                    if (j==0)
klawisz(10+i*25,395+j*20,(i+1)*25+5,390+(j+1)*20,kolor,nazw_przeb[i],tekst
_przyc);
                    else
klawisz(10+i*25,395+j*20,(i+1)*25+5,390+(j+1)*20,kolor,nazw_przeb[i+25],te
kst_przyc);
                    ShowMouse();
                    rysuj_przebieg();
                    zezw_odtw=1;
                }
            }
        }

        /* obsługa Syntezy */

        if (nr_przycisku==1 &&
            x>220 && x<420 && y>10 && y<30)
        {
            HideMouse();
                ShowMouse();
            do {pozycja_kursora();} while (nr_przycisku!=0);
            HideMouse();
            ShowMouse();
            flaga_przeb_czasowe=0;
            zezw_odtw=0;
```

```
            flaga_synt_alof=1;
            flaga_opis_synt=0;
            koniec_pliku=0;
            synt_alof();
        }

        /* obsluga >> w Synteza alofonowa */

        if (nr_przycisku==1 && flaga_synt_alof==1 &&
            koniec_pliku!=1 &&
            x>10 && x<110 && y>450 && y<470)
        {
            HideMouse();

klawisz_wcisn(10,450,110,470,kolor,">>",tekst_przyc_wcisn);
            ShowMouse();
            do {pozycja_kursora();} while (nr_przycisku!=0);
            HideMouse();
            klawisz(10,450,110,470,kolor,">>",tekst_przyc);
            setfillstyle(SOLID_FILL,kolor);
            bar(12,212,628,438);
            poz_litery_y=0;
            poz_litery_x=0;

            do
            {
                wypisz_tekst(20,220);
                if (litery[0]==EOF) koniec_pliku=1;
            } while (poz_litery_y<=20 && koniec_pliku!=1);
```

```
                ShowMouse();
            }
            if (nr_przycisku==1 &&
                x>430 && x<630 && y>10 && y<30)
            {
                HideMouse();
                klawisz_wcisn(430,10,630,30,kolor,"Opis
syntezera",tekst_przyc_wcisn);
                ShowMouse();
                do {pozycja_kursora();} while (nr_przycisku!=0);
                HideMouse();
                    ShowMouse();
                flaga_przeb_czasowe=0;
                zezw_odtw=0;
                flaga_opis_synt=1;
                flaga_synt_alof=0;
                koniec_pliku=0;
                opis_synt();
            }
            if (nr_przycisku==1 && flaga_opis_synt==1 &&
                koniec_pliku!=1 &&
                x>10 && x<110 && y>450 && y<470)
            {
                HideMouse();

    klawisz_wcisn(10,450,110,470,kolor,">>",tekst_przyc_wcisn);
                ShowMouse();
                do {pozycja_kursora();} while (nr_przycisku!=0);
                HideMouse();
```

```
            setfillstyle(SOLID_FILL,kolor);
            bar(12,62,628,438);
            poz_litery_y=0;
            poz_litery_x=0;

            do
            {
                  wypisz_tekst(20,70);
                  if (litery[0]==EOF) koniec_pliku=1;
            } while (poz_litery_y<=35 && koniec_pliku!=1);

            klawisz(10,450,110,470,kolor,">>",tekst_przyc);
            ShowMouse();
      }

      /* obsluga Koniec */

      if (nr_przycisku==1 &&
            x>530 && x<630 && y>450 && y<470)
      {
            HideMouse();

klawisz_wcisn(530,450,630,470,kolor,"Koniec",tekst_przyc_wcisn);
            ShowMouse();
            do {pozycja_kursora();} while (nr_przycisku!=0);
            HideMouse();
                  ShowMouse();
            flaga_exit=1;
            zezw_odtw=0;
```

```
            }
        } while (flaga_exit!=1);

        flaga_exit=0;
        ekran();
}

void opis_synt(void)
{
        HideMouse();
        fclose(op_syn);
                == NULL)
        {
            blad("Unable to open *** ");
        }
        setcolor(kolor);
        bar(5,32,634,448);
        bar(5,445,115,475);

        klawisz_wcisn(10,40,630,55,kolor_wcisn,"Opis syntezera",kolor_opisu);
        klawisz_wcisn(10,60,630,440,kolor," ",tekst_przyc_wcisn);
        klawisz(10,450,110,470,kolor,">>",tekst_przyc_wcisn);

        flaga_opis_synt=1;
        poz_litery_y=0;
        poz_litery_x=0;

        do
        {
```

```
            wypisz_tekst(20,70);
            if (litery[0]==EOF) koniec_pliku=1;
        } while (poz_litery_y<=35 && koniec_pliku!=1);

        ShowMouse();
}

void synt_alof(void)
{
        HideMouse();
        fclose(syn_alof);
            setcolor(kolor);
        bar(5,32,634,448);
        bar(5,445,115,475);

        setcolor(kolor_tekstu);
        rectangle(70,90,210,170);
        rectangle(250,90,390,170);
        rectangle(430,90,570,170);

        line(20,130,70,130);
        line(210,130,250,130);
        line(390,130,430,130);
        line(570,130,620,130);

        line(65,125,70,130);
        line(65,135,70,130);
        line(245,125,250,130);
        line(245,135,250,130);
```

```
        line(425,125,430,130);
        line(425,135,430,130);
        line(615,125,620,130);
        line(615,135,620,130);

        setcolor(kolor_tekstu);
        outtextxy(20,120,"Tekst");
        outtextxy(575,120,"Mowa");
        outtextxy(500-strlen("Blok")*4,125,"Blok");
        outtextxy(500-strlen("syntezy")*4,135,"syntezy");

        poz_litery_y=0;
        poz_litery_x=0;
        flaga_opis_synt=1;

        do
        {
                wypisz_tekst(20,220);
                if (litery[0]==EOF) koniec_pliku=1;
        } while (poz_litery_y<=20 && koniec_pliku!=1);

        ShowMouse();
}

void przeb_czasowe(void)
{
        HideMouse();

        setcolor(kolor);
```

```
        bar(5,32,634,448);
                for (i=0;i<50;i++)
        {
                memset(nazw_przeb[i],0,3);
                strncpy(nazw_przeb[i],nazw_zbioru[i],2);
        }

        for (j=0;j<2;j++)
        {
                for (i=0;i<25;i++)
                {
                        if                                              (j==0)
klawisz(10+i*25,395+j*20,(i+1)*25+5,390+(j+1)*20,kolor,nazw_przeb[i],tekst
_przyc);
                        else
klawisz(10+i*25,395+j*20,(i+1)*25+5,390+(j+1)*20,kolor,nazw_przeb[i+25],te
kst_przyc);
                }
        }
        ShowMouse();
}
void rysuj_przebieg(void)
{

int     k=0;
int     l;
int     m=32;
int     co_ile;
int     probka_1=0;
```

```
int     probka_2=0;

        HideMouse();

        setfillstyle(SOLID_FILL,kolor);
        bar(12,62,625,328);
        bar(12,367,303,383);
        bar(317,367,628,383);
        setcolor(kolor_opisu);
        outtextxy(158-
strlen(itoa(floor(dlug_zbioru[i+rzad*25]/14),string,10))*4,372,itoa(floor(dlug_z
bioru[i+rzad*25]/14),string,10));
        outtextxy(463-
strlen(itoa(max_sample[i+rzad*25],string,10))*4,372,itoa(max_sample[i+rzad*
25],string,10));
        setcolor(LIGHTRED);
        line(15,195,615,195);
        setcolor(YELLOW);
        setlinestyle(SOLID_LINE,0,NORM_WIDTH);

        co_ile=floor(dlug_zbioru[i+rzad*25]/600);

        do
        {
                probka_1=peekb(FP_SEG(lpVoiceBuf[i+rzad*25]),m);

                if (probka_1>=0)
                {
                        probka_1=128-probka_1;
```

```
            }
            else
            {
                probka_1=-128-probka_1;
            }


            for (l=0;l<co_ile;l++)
            {
                m++;
            }

            line(k+20,195+probka_2,k+21,195+probka_1);
            k++;
            m++;
            probka_2=probka_1;

        } while ((m-31)<dlug_zbioru[i+rzad*25]);

        ShowMouse();
}

void pomoc(void)
{
        unsigned int        rozm_ramki,ramka_seg;
        void far      *ramka_pomoc;

        rozm_ramki = imagesize(160,140,480,340);
```

```
if (allocmem((unsigned)((rozm_ramki + 15) >> 4),&ramka_seg) == -1 )
{
        FP_SEG(ramka_pomoc) = ramka_seg;
        FP_OFF(ramka_pomoc) = 0;
}
else
   blad("Memory allocation error");

HideMouse();

getimage(160,140,480,340);

setfillstyle(SOLID_FILL,LIGHTRED);
bar(160,140,480,340);
klawisz_wcisn(170,150,470,165,RED,"Pomoc",kolor_opisu);
klawisz_wcisn(170,175,470,330,LIGHTRED," ",kolor_tekstu);
klawisz(400,300,460,320,LIGHTRED,"Ok",tekst_przyc);
if ((pom = fopen("\\syntezer\\pomoc.txt", "rt"))
        == NULL)
poz_litery_x=0;
poz_litery_y=0;
flaga_wybrany=0;
flaga_opis_synt=0;
flaga_pomoc=1;
setcolor(WHITE);
do
{
        wypisz_tekst(180,180);
} while (litery[0]!=EOF);
```

```
ShowMouse();
do
{
        pozycja_kursora();

        /* obsluga Ok w Pomoc */

        if (nr_przycisku==1 /*&&
                x>400 && x<460 && y>300 && y<320*/)
        {
                HideMouse();

klawisz_wcisn(400,300,460,320,LIGHTRED,"Ok",YELLOW);
                ShowMouse();
                do {pozycja_kursora();} while (nr_przycisku!=0);
                HideMouse();
                klawisz(400,300,460,320,LIGHTRED,"Ok",WHITE);
                ShowMouse();
                flaga_exit=1;
        }
} while (flaga_exit!=1);

HideMouse();
putimage(160,140,ramka_pomoc,COPY_PUT);
ShowMouse();
freemem(ramka_seg);
flaga_pomoc=0;
flaga_exit=0;
```

```
        fclose(pom);
}

void czytaj_znak(void)
{
  while(bioskey(1)==0);
  key[0]=(char)((bioskey(0) & 0xff));
}
void edycja(void)
{
        memset(wyraz,0,75);
        i=0;
        setcolor(kolor_tekstu);
        outtextxy(30,277,"_");
        while(czytaj_znak(),key[0]!=0xd)
        {
                switch(key[0])
                {
                        case 0x8 :
                                {
                                        if (i>0) i--;
                                        wyraz[i+1]=0;
                                        setfillstyle(SOLID_FILL,BLACK);
                                        bar(30+8*i,272,30+8*(i+2),288);
                                        outtextxy(30+8*i,277,"_");
                                };break;
                        case '`' : {
                                        czytaj_znak();
                                        {
```

```
                        if (i<72)
                        {
                            setfillstyle(SOLID_FILL,BLACK);
                            bar(30+8*i,272,30+8*(i+1),288);
                            pol_lit();
                            outtextxy(30+8*i,277,"_");
                        }
                    }
                };break;
            default :
                {
                    if (i<72)
                    {
                        setfillstyle(SOLID_FILL,BLACK);
                        bar(30+8*i,272,30+8*(i+1),288);
                        wyraz[i]=key[0];
                        setcolor(kolor_tekstu);
                        outtextxy(30+8*i,277,key);
                        outtextxy(30+8*(i+1),277,"_");
                        i++;
                    }
                };break;
        }
    }
    dlug_edycji=i;
    setfillstyle(SOLID_FILL,BLACK);
    bar(30+8*i,272,30+8*(i+1),288);

}
```

```
void pol_lit(void)
{
        setcolor(kolor_tekstu);
        switch (key[0])
        {
            case 'A': {
                    outtextxy(30+i*8,277,"A");
                    outtextxy(34+i*8,279,",");
                    wyraz[i]='¤';
                    i++;
              };break;
            case 'E': {
                    outtextxy(30+i*8,277,"E");
                    outtextxy(34+i*8,279,",");
                    wyraz[i]='"';
                    i++;
              };break;
            case 'O' : {
                    outtextxy(30+i*8,277,"O");
                    outtextxy(31+i*8,269,",");
                    wyraz[i]='ŕ';
                    i++;
              };break;
            case 'S': {
                    outtextxy(30+i*8,277,"S");
                    outtextxy(31+i*8,269,",");
                    wyraz[i]='—';
                    i++;
```

```
 };break;
case 'L': {
        outtextxy(30+i*8,277,"L");
        line(32+8*i,281,35+8*i,278);
        wyraz[i]='ť';
        i++;
 };break;
case 'Z': {
        outtextxy(30+i*8,277,"Z");
        outtextxy(31+i*8,269,".");
        wyraz[i]='˝';
        i++;
 };break;
case 'X': {
        outtextxy(30+i*8,277,"Z");
        outtextxy(31+i*8,269,",");
        wyraz[i]='Ť';
        i++;
 };break;
case 'C': {
        outtextxy(30+i*8,277,"C");
        outtextxy(31+i*8,269,",");
        wyraz[i]='Ź';
        i++;
 };break;
case 'N': {
        outtextxy(30+i*8,277,"N");
        outtextxy(31+i*8,269,",");
        wyraz[i]='ă';
```

```
        i++;
  };break;
case 'a': {
        outtextxy(30+i*8,277,"a");
        outtextxy(33+i*8,279,",");
        wyraz[i]='Ą';
        i++;
  };break;
case 'e': {
        outtextxy(30+i*8,277,"e");
        outtextxy(33+i*8,279,",");
        wyraz[i]='©';
        i++;
};break;

case 'c': {
        outtextxy(30+i*8,277,"c");
        outtextxy(32+i*8,270,",");
        wyraz[i]='†';
        i++;
};break;
case 'l': {
        outtextxy(30+i*8,277,"l");
        line(32+8*i,281,35+8*i,278);
        wyraz[i]='□';
        i++;
};break;
case 'n': {
        outtextxy(30+i*8,277,"n");
```

```
        outtextxy(32+i*8,270,",");
        wyraz[i]='ä';
        i++;
};break;
case 'o': {
        outtextxy(30+i*8,277,"o");
        outtextxy(32+i*8,270,",");
        wyraz[i]='˜';
        i++;
};break;
case 's': {
        outtextxy(30+i*8,277,"s");
        outtextxy(32+i*8,270,",");
        wyraz[i]='□';
        i++;
};break;
case 'z': {
        outtextxy(30+i*8,277,"z");
        outtextxy(30+i*8,270,".");
        wyraz[i]='l';
        i++;
};break;
case 'x': {
        outtextxy(30+i*8,277,"z");
        outtextxy(32+i*8,270,",");
        wyraz[i]='«';
        i++;
};break;
default : break;
```

```
        }
}

void save_screen(void)
{
  unsigned  size;
  int       ystart=0, yend, yincr, block;

  yincr = (maxy+1) / 4;
  yend = yincr;
  size = imagesize(0, ystart, maxx, yend);

  for (block=0; block<=3; block++)
  {
    if ((buf[block] = farmalloc(size)) == NULL)
    {
        blad("Not enough heap space");
    }

    getimage(0, ystart, maxx, yend, buf[block]);
    ystart = yend + 1;
    yend += yincr + 1;
  }
}

void restore_screen(void)
{
  int  ystart=0, yend, yincr, block;
```

```
    yincr = (maxy+1) / 4;
    yend = yincr;
    cleardevice();
    for (block=0; block<=3; block++)
    {
      putimage(0, ystart, buf[block], COPY_PUT);
      /*farfree(buf[block]);*/
      ystart = yend + 1;
      yend += yincr + 1;
    }
}


void blad(char tekst_bledu[35])
{
        ShowMouse();
        HideMouse();
        setfillstyle(SOLID_FILL,BLACK);
        bar(0,0,639,479);
        klawisz(170,135,470,250,LIGHTRED," ",tekst_przyc);
        klawisz(260,220,380,240,LIGHTRED,"Koniec",WHITE);
        klawisz_wcisn(180,145,460,160,RED,"ERROR MESSAGE",WHITE);
        klawisz_wcisn(180,170,460,210,LIGHTRED,tekst_bledu,WHITE);
        ShowMouse();
        do
        {
                pozycja_kursora();
                if (nr_przycisku==1 && x>260 && x<380 && y>220 && y<240)
                        flaga_exit=1;
```

```
    } while (flaga_exit!=1);
    HideMouse();
    klawisz_wcisn(260,220,380,240,LIGHTRED,"Koniec",BLACK);
    ShowMouse();
    do { pozycja_kursora(); } while (nr_przycisku!=0);
    HideMouse();
    klawisz(260,220,380,240,LIGHTRED,"Koniec",WHITE);
    delay(500);
    setfillstyle(SOLID_FILL,BLACK);
    bar(0,0,639,479);
    ShowMouse();
    closegraph();
    clrscr();
    exit(EXIT_FAILURE);
}
```

Podsumowanie

Wydaje się, że dla osiągnięcia wysokich standardów konstruowanego narzędzia informatycznego – symulatora operacyjno taktycznych działań powietrznych - wymagane było wyposażenie go w funkcje multimedialne. Sądzę, że brak dźwięku w tym symulatorze:

- dla wybranych komunikatów;
- dla wyeksponowania ważnych zdarzeń podczas grania,

istotnie zubożyłby ten system.

Opanowując procedury implementacji dźwięku do programów komputerowych, rozwiązano szereg istotnych problemów:

- opracowano wzorcowe komunikaty,
- skomponowano sample wybuchy i dołączono je do programów (odpowiednich zdarzeń);
- opracowane zostały wydajne mechanizmy dystrybucji danych multimedialnych.

Podsumowując – kilka uwag natury bardziej ogólnej. W moim przekonaniu wszelkie nowe i niekonwencjonalne rozwiązania organizacyjne i techniczne - na jakie oczywiście pozwala jednolita technologia - mają coraz bardziej decydujący wpływ na jakość nauczania (a przecież ten system ma być eksploatowany na Wydziale Lotnictwa i Obrony Powietrznej w Akademii Obrony Narodowej w Warszawie, wspomagając wybrane procesy dydaktyczne).

Wiedza, wysokie kompetencje, multimedia, show i wiele innych – to wyróżniki nowej konwencji kształcenia. Bryan Travis, matematyk z Los Alamos, przebywając w Warszawie na sympozjum Microsoft przedstawiając wykład na temat nowego kompilatora przebrany był za Neo, głównego bohatera filmu Matrix. Informatyk Jan Madey z Uniwersytetu Warszawskiego tłumacząc

działanie systemu operacyjnego wykorzystuje studentów jako żywe procedury i mikrorozkazy[14].

Uważam, że nowoczesne symulatory w zastosowaniach wojskowych powinny być nasycone dużą liczbą efektów multimedialnych, co odpowiednio zrealizowane uatrakcyjni obsługę systemu.

Na koniec odwołam się do kilku pobudzających do refleksji impulsów.

Kompetencje współczesnego oficera/studenta/słuchacza kreowane są poprzez pewnego rodzaju spektakl, najczęściej przemyślany i celowo zaaranżowany przez wykładowców. Celem takiego spektaklu jest aktywizacja, motywacja i utrzymanie koncentracji odbiorcy przez jak najdłuższy czas (nie tylko jak wykazują badania zaledwie kilka/kilkanaście minut). Co więcej - umiejętne stosowanie różnorodnych mediów buduje najbardziej pożądane z punktu widzenia efektywności kształcenia, twórcze interakcje nadawca-odbiorca.

W moim przekonaniu właśnie w tym kierunku będą ewoluować przemiany w kształceniu. Być może jest to trend krótkookresowy, ale więcej przesłanek wskazuje na to, iż tendencja ta przerodzi się w megatrend. Społeczeństwa informacyjne, w tym kadra studiująca w Akademii Obrony Narodowej oczekują i są gotowe zapłacić za efektywne kształcenie realizowane w oparciu o nowoczesne symulatory. Właśnie symulatory z elementami multimedialnymi przyczynią się do najbardziej znaczących (pozytywnych) postępów w naukach wojskowych..

---

[14] Zych J., Systemy multimedialne w zastosowaniach wojskowych, V Międzynarodowa Konferencja Naukowa MEDIA A EDUKACJA Kompetencje medialne współczesnego człowieka, EMPI², Poznań 2004.

Literatura

1. Brachmański Stefan: Obiektywne metody oceny jakości transmisji mowy, Krajowe Sympozjum Telekomunikacji 2000.
2. Bromirski M.: Nowoczesne systemy telekonferencji multimedialnych, Telecom Forum 11/99, 12/99, 1/00.
3. Burakowski Wojciech: Sieci IP z jakością obsługi, Krajowe Sympozjum Telekomunikacji 2000.
4. Craig Marven, Gillian Ewers: Zarys cyfrowego przetwarzania sygnałów, Wydawnictwa Komunikacji i Łączności, Warszawa 1999
5. Czyżewski Andrzej: Dźwięk cyfrowy, Akademicka Oficyna Wydawnicza EXIT, Warszawa 1998
6. Domański Marek: Zaawansowane techniki kompresji obrazów i sekwencji wizyjnych, Wydawnictwo Politechniki Poznańskiej, Poznań 2000
7. Haykin Simon: Systemy telekomunikacyjne cz.1, Wydawnictwa Komunikacji i Łączności, Warszawa 1998
8. Hulicki Zbigniew: Systemy komunikacji multimedialnej, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków
9. Klink Janusz: Systemy komunikacji multimedialnych, Krajowe Sympozjum Telekomunikacji 1999.
10. Krzemień Tomasz: Subiektywna ocena jakości obrazu telewizyjnego jednobodźcową metodą skalowania proporcji, STM 2000
11. Rec. ITU-T G.722: 7 [kHz] audio-coding within 64 kbit/s, Melbourne 1988
12. Rec. ITU-T G.723.1: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s, Geneva 1996
13. Rec. ITU-T G.728: Coding of speech at 16 kbit/s using low-delay code excited linear prediction, Geneva 1992

14. Rec. ITU-T G.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear-prediction, Geneva 1996
15. Rec. ITU-T H.261: Video codec for audiovisual services at px64 kbit/s, Helsinki 1993
16. Ruchała Radosław: Wideokonferencje w sieciach IP – możliwości a praktyka, Studia Informatica 2000.
17. Sochan Arkadiusz: Laboratoryjne metody badania sieci ATM, Zeszyty naukowe Politechniki Śląskiej 1999.
18. Trzaskowska Jolanta: Subiektywna ocena jakości mowy w sieciach telekomunikacyjnych, Krajowe Sympozjum Telekomunikacji 2000
19. Zdrodowski B., Zych J. Model działań powietrznych, etap I, Model taktycznych działań powietrznych, AON, Warszawa 2002.
20. Zdrodowski B., Zych J. Model działań powietrznych, etap II, Rozpoznanie i zarządzanie zasobami w modelu działań powietrznych, AON, Warszawa 2002.
21. Zdrodowski B., Zych J. Model działań powietrznych, etap III, Teren w modelu działań powietrznych, AON, Warszawa 2003.
22. Zdrodowski B., Zych J. Model działań powietrznych, etap IV, Warunki meteorologiczne w modelu działań powietrznych, AON, Warszawa 2003.
23. Zdrodowski B., Zych J. Symulator operacyjno-taktycznych działań powietrznych - GAMBLER, tom 1, Koncepcja realizacji projektu, AON, Warszawa 2003.
24. Zdrodowski B., Zych J. Symulator operacyjno-taktycznych działań powietrznych - GAMBLER, tom 2, Założenia funkcjonalno-techniczne symulatora operacyjno-taktycznych działań powietrznych, AON, Warszawa 2003.
25. Zdrodowski B., Zych J. Symulator operacyjno-taktycznych działań powietrznych - GAMBLER, tom 3, Wymagania bazy technologicznej i

oprogramowania symulatora operacyjno-taktycznych działań powietrznych, AON, Warszawa 2003.

26. Zych J., Computerised Simulation Game, Proceedings of The Regional Conference on Military Communication and Information Systems, Zegrze, Poland, October 8-10 2003.

27. Zych J., Cybernetyczny aspekt przetwarzania informacji, Przegląd Wojsk Lotniczych i Obrony Powietrznej, nr 12, Poznań 2003.

28. Zych J., Meteorological Aspects of the Gambit War Game, Metoc Services' Task in NATO Operations, Missions and Exercises, ISBN 83-99997-22-5, Poznań, Poland 2003.

29. Zych J., Systemy multimedialne w zastosowaniach wojskowych, V Międzynarodowa Konferencja Naukowa MEDIA A EDUKACJA Kompetencje medialne współczesnego człowieka, EMPI^2, Poznań 2004.

30. Zych. J., Model walki sił obrony powietrznej szczebla taktycznego, Rozprawa doktorska, Akademia Obrony Narodowej, Warszawa 2002.