**Зміст**

[1. ЗАГАЛЬНА ЧАСТИНА 4](#_Toc402716945)

[1.1 Аналіз предметної області 5](#_Toc402716946)

[1.2 Постановка задачі 5](#_Toc402716947)

[1.3 Обґрунтування вибору засобів проектування та розробки ПЗ. 6](#_Toc402716948)

[2. СПЕЦІАЛЬНА ЧАСТИНА 6](#_Toc402716949)

[2.1 Проектування загальної алгоритмічної моделі ПЗ 8](#_Toc402716950)

[2.2 Проектування інтерфейсу користувача ПЗ. 9](#_Toc402716951)

[2.2.1 Опис моделі взаємодії обраних компонентів 10](#_Toc402716952)

[2.2.2 Проектування методів та функцій ПЗ 12](#_Toc402716953)

[2.2.3 Ергономіка інтерфейсу користувача 13](#_Toc402716954)

[2.3 Засоби візуального проектування ПЗ 16](#_Toc402716955)

[3. ЗАКЛЮЧНА ЧАСТИНА 19](#_Toc402716956)

[3.1 Відлагодження і тестування програмного продукту 19](#_Toc402716957)

[3.2 Інструкція користувача](#_Toc402716958) 23

[3.3 Розробка дистрибутивного диску 27](#_Toc402716959)

[Висновок](#_Toc402716960) 30

Список використаної літератури 31

Додаток А. Код програми 32

**Вступ**

# В даний час мультимедіа-технології є бурхливо розвивається областю інформаційних технологій. В цьому напрямку активно працює значна кількість великих і дрібних фірм, технічних університетів і студій (зокрема 1ВМ, Aрр1е, Моtого1а, Philips, Sоnу, Intel та ін.). Області використання надзвичайно різноманітні: інтерактивні навчальні та інформаційні системи, САПР, розваги та ін.

# Аудиоряд може включати мова, музику, ефекти (звуки типу шуму, грому, скрипу і т.д., що об'єднуються позначенням WAVE (хвиля) . Головною проблемою при використанні цієї групи мультісреди є інформаційна ємність. Для запису однієї хвилини WAVE-звуку вищої якості необхідна пам'ять близько 10 Мбайт, тому стандартний об'єм CD (до 640 Мбайт) дозволяє записати не більше години WAVE. Для вирішення цієї проблеми використовуються методи компресії звукової інформації.

# Іншим напрямком є ​​використання в мультісреде звуків (одноголоса і багатоголоса музика, аж до оркестру, звукові ефекти) MIDI (MusicalInstrumentDigitaleInterface). В даному випадку звуки музичних інструментів, звукові ефекти синтезуються програмно-керованими електронними синтезаторами. Корекція і цифровий запис MIDI-звуків здійснюється за допомогою музичних редакторів (програм-секвенсоров). Головною перевагою MIDI є малий обсяг необхідної пам'яті - 1 хвилина MIDI-звуку займає в середньому 10 Кбайт.1. ЗАГАЛЬНА ЧАСТИНА

## Аналіз предметної області

Програвач мультимедіа (медіа-плеєр, англ. Media player) —тип комп'ютерних програм, призначених для відтворення мультимедіа файлів. Більшість програмних мультимедіа програвачів підтримують значну кількість медіа-форматів, включаючи аудіо і відео файли.Деякі мультимедіа програвачі призначені для відтворення тільки аудіо або відео файлів і називаються, відповідно, програвачі аудіо (аудіоплеєри) і програвачі відео (відеоплеєри). Розробники таких програвачів прагнуть зробити їх якомога зручнішим для відтворення відповідних форматів.

Операційна система Microsoft Windows уже містить у собі програвач медіа файлів Windows Media Player. Остання версія цього програвача, Windows Media Player 11, вбудована в операційну систему Windows Vista. Операційна система Mac OS X поставляється з убудованим програвачем QuickTime Player, який призначений для відтворення відео у форматі QuickTime, і програвачем iTunes, призначеним для відтворення багатьох медіа-форматів. Для Unix/Linux популярні мультимедіа програвачі Amarok, Rhythmbox, VLC, Audacious, MPlayer, xine і Totem.

Багато мультимедіа програвачів використовують бібліотеки, які призначені для систематизації й каталогізації музики й відео по категоріях, таким як жанр, рік запису, оцінка й ін. Гарним прикладом мультимедиа програвачів, які використають бібліотеки є Winamp, Windows Media Player, iTunes, RealPlayer і Amarok.

## 1.2 Постановка задачі

В даній задачі розглядається аудіоплеер, який має змогу відвторювати аудіофайли які знаходяться в Інтернеті, або прослуховувати непреривні інтернет-радіовіщяння.

Користувач має можливість:

* Створити або відкрити власний файл плейлісту
* Редагувати файл плейлісту, та зберегти його
* Відтворювати аудіо фали, або інтернет-радіовіщяння

Вхідною інформацією є дані адреса для доступу до аудіо ресурса.

Вихідною інформацією є відтворення аудіофайлу.

На вище вказаних даних, для реалізації програмного забезпечення було обрано інтегроване середовище розробки Embarcadero Rad Studio XE7.

## 1.3 Обґрунтування вибору засобів проектування та розробки ПЗ.

Embarcadero RAD Studio - середовище швидкої розробки додатків (RAD) для Microsoft Windows фірми Embarcadero Technologies.

Поточна версія Embarcadero RAD Studio XE6 об'єднує Delphi XE6 і C + + Builder XE6 в єдину інтегровану середовище розробки. Візуальная середа написана мові програмування Delphi для ОС Windows. Має ліцензію на розповсюдження типу: пропрієтарне програмне забезпечення.

RAD Studio XE6 - єдине рішення на ринку , яке пропонує створення додатків на основі нативного коду , що працюють на всіх чотирьох популярних платформах: Android , iOS , Windows і OS X. Створення додатків здійснюється за допомогою однієї і тієї ж бази вихідного коду , не жертвуючи якістю додатків , продуктивністю і доступністю корпоративних або хмарних ресурсів . RAD Studio XE5 дозволяє користувачам створювати нативні програми для PC , планшетів і смартфонів для розміщення їх в магазинах додатків або всередині корпоративної ІТ системи .

По-справжньому native-додатки виконуються безпосередньо на апаратному забезпеченні без механізму скриптів або інтерпретації , що дозволяє забезпечити максимальний рівень інтерактивності і якості інтерфейсу .

Нове опитування, проведене компанією Dimensional Research, випущений сьогодні, показує, що для 85 % розробників під Windows актуальне питання розробки під мобільні платформи і, в теж час, 85 % віддають перевагу нативний код, і тільки 17 % дійсно вірять, що вони зможуть створювати нативні програми для двох або більше платформ з прийнятними витратами .

Переваги RAD Studio XE5 :

* Нативная підтримка Android і iOS.Користувачі тепер можуть розробляти нативні програми під Android і iOS для смартфонів і планшетів, що дозволить їм виходити на найбільший і швидко зростаючий мобільний ринок. Платформа також забезпечує доступ до сенсорів мобільних пристроїв , послуг і даними безпосередньо на пристрої, в хмарі і всередині компанії.
* Одна кодова база, одна команда.Розробники можуть робити і прототипи , і самі нативні додатки на Delphi паралельно, це означає, що вам не потрібно керувати окремими проектами, бюджетами, командами, використовуючи різні засоби розробки і бібліотеки для кожної платформи .
* Платформа додатків для різних пристроїв.RAD Studio XE5 має всі складові для платформи додатків для різних пристроїв ( MDAP ) . Підтримка REST клієнта забезпечує взаємодію з Rest сервісами, включаючи соціальні мережі, MBaaS, платіжні сервіси, зберігання та інше. Це - відмінна внутрішньокорпоративна або хмарна платформа для даних або сервісів з доступом з вашої програми. Розробники можуть органічно вписатися в концепцію BYOD і бути спокійні, що їх програми будуть працювати не тільки на різних пристроях, а й підтримувати різні backend -сервіси, як усередині корпоративної ІТ інфраструктури , так і в хмарі.

# 2. СПЕЦІАЛЬНА ЧАСТИНА

## 2.1 Проектування загальної алгоритмічної моделі ПЗ

На основі інформації з постановки задачі був побудований загальний алгоритм роботи програми, який показує основний процес ввід даних та їх використання (Рис 1).

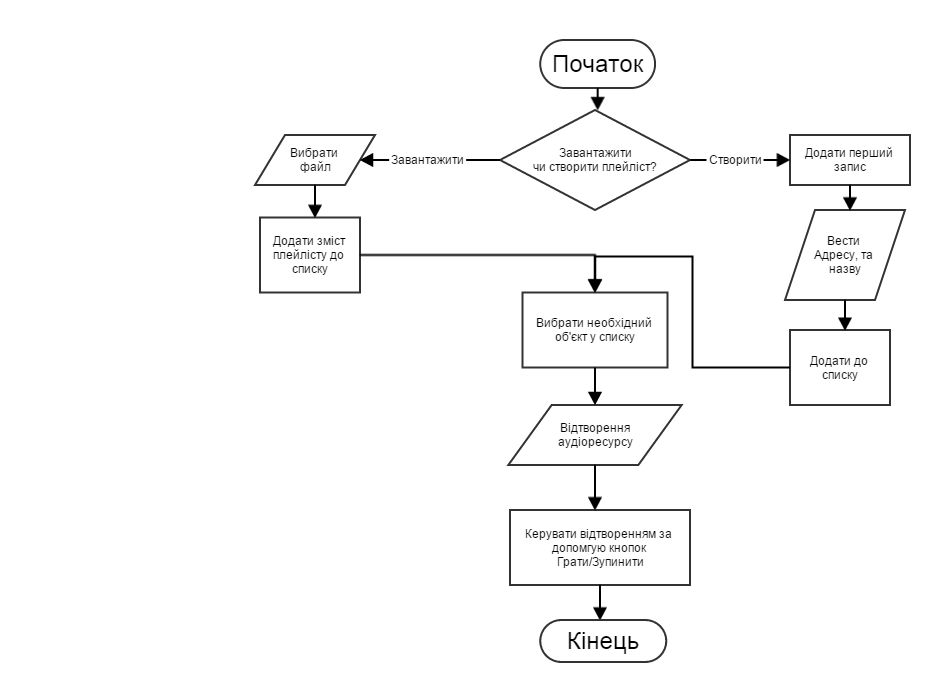


Рис. 1- Блок-схема загального алгоритму

На рисунку 1 зображено блок-схему загального алгоритму програми. Спочатку при відкриті програми ми повинні вибрати плейліст із посиланнями на аудіофайли формату (.pls), або створити новий плейліст.

Якщо користувачу необхідно буде завантажити плейліст з фалу, зявиться вікно у якому можно буде вказати на файл у файловій системі. Користувач сможе бачити лише файли тільки формату (.pls) папки, та ярлики.

У випадку створення нового плейлісту, користувачу необхідно створити перший запис для роботи із плейлістом. З’явиться вікно де користувач вкаже адресу музикального об’єкта, та його імя.

Для початку відтворення, користувачу має вибрати один із обєктів у списку, натиснути кнопку відтворення. Подальше керування відтворюється за допомогою додаткових елементів інтерефейсу (такі як регулятор гучності, перемотка…).

Музикальні об’єкти можно редагувати та виключати зі списку на панелі. Для редагування з’явиться вікно де користувач зможе редагувати поточно виділений запис. Також присутня можливість зберегти плейліст зі змінами що внес користувач. Користувачу буде надана можливість вибрати місце збережння плейлісту за допомогою додаткового діалогового вікна.

## 2.2 Проектування інтерфейсу користувача ПЗ.

Інтерфейс користувача, він же користувальницький інтерфейс (UI -. Англ користувальницький інтерфейс) - різновид інтерфейсів, в якому одна сторона представлена ​​людиною (користувачем), інша - машиною / пристроєм. Являє собою сукупність засобів і методів, за допомогою яких користувач взаємодіє з різними, найчастіше складними, машинами, пристроями і апаратурою.

Вельми часто термін застосовується по відношенню до комп'ютерних програм, однак під ним може матися на увазі набір засобів, методів і правил взаємодії будь-якої системи, керованої людиною.

Інтерфейс двонаправлений (інтерактивний) - коли пристрій, отримавши команди від користувача і виконавши їх, видає інформацію користувачу наявними у неї засобами - візуальними, звуковими, тактильними і т. п. (Прийнявши яку, користувач видає влаштуванню наступні команди наданими в його розпорядження коштами :. Кнопки, перемикачі, регулятори, сенсори, голосом, і т д.).

Оскільки інтерфейс є сукупність, тобто він складається з елементів, які, самі по собі, також можуть складатися з елементів (так, екран дисплея може містити в собі інші вікна, які, в свою чергу, можуть містити панелі, кнопки та інші інтерфейсні елементи ). Особливе і окрему увагу в інтерфейсі користувача традиційно приділяється його ефективності і зручності користування (юзабельних). Зрозумілий, зручний, дружній - його основні характеристики.

### 2.2.1 Опис моделі взаємодії обраних компонентів

Для реалізації поставленої задачі були обрані наступні компоненти:

* BitButton
* TImage
* EditBox
* TrackBar
* ProgressBar
* MainMenu

Усі компоненті так чи інакше взаємодіють між-собою.

Для реалізації кнопок контролю над медіа-програвачем було принято рішення використовувати компонент TImage замість TButton (та його аналогів), тому що – реалізації анімації кнопки натискання неможливо зі звичайним компонентом кнопки, тому малюнок у компоненті TImage змінюється залежно від ситуації.

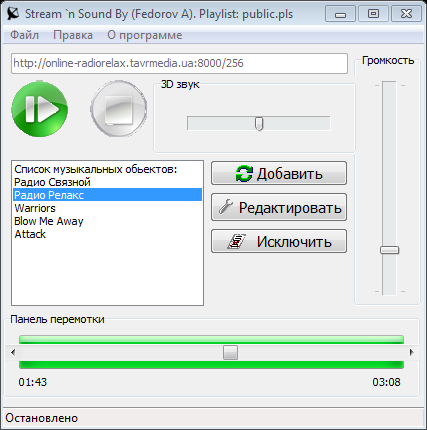


Рис. 2 - Компонент TImage

Компонент EditBox найзручніший компонент для вводу текстової інформації та її відображення. В проекті він використовується для відображення URL адресу поточного аудіо-обєкта. У цьому випадку редагування компонента забороненно, а використовуеться лише для відображення. Цей компонент був вибраний замість Label, тому ще створює свій фон на якому видно текст, що явно виглядає краще. Також цей компонент використовуеться для зміни імені плейлісту.

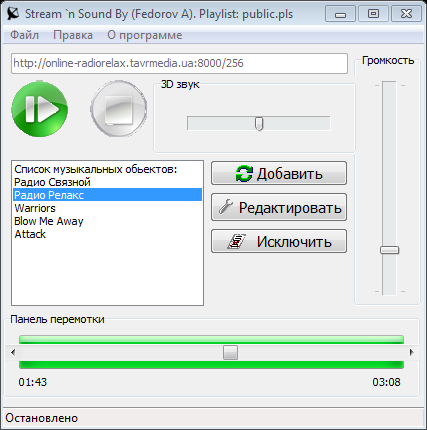


Рис. 3- Компонент EditBox

TrackBar – (повзунок) дуже зручний компонент для зміни значень заданого діапазону. Користувач може задати значення у цьомі діапозаноі користуючись лише вказівноком миші. У проекті повзунок використовуеться при регулюванні звука та встановленні 3D звучання, та перемотування.

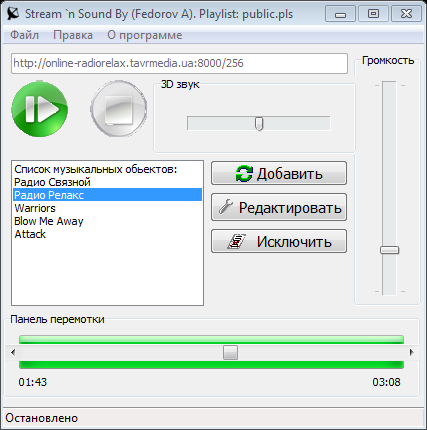


Рис. 4- Компонент TrackBar

ProgressBar – (шкала заповнення) реалізується у буффері завантаження.

Відображає скільки відстоків файлу було завантежно і готово до програвання, разом із компонентом ScrollBar візуально помітно до якої миті можно перемотати аудіофайл.

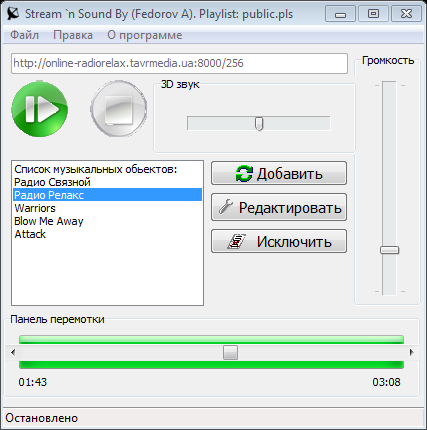


Рис. 5- Компонент ProgressBar та ScrollBar

MainMenu – компонент створю меню з кнопками, при натисканні яких свторються контекстні меню. Дуже зручно та звично для будь-якого користувача, тому що присутній майже у кожній программі.

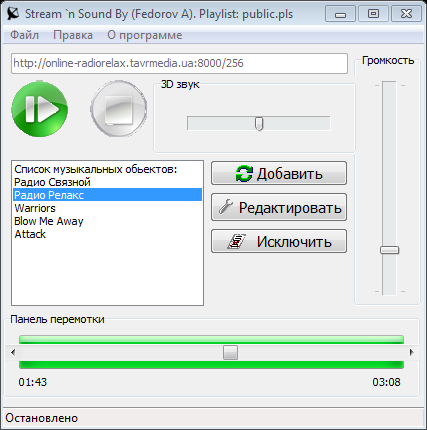


Рис. 6- Компонент MainMenu

### 2.2.2 Проектування методів та функцій ПЗ

Для реалізації поставленої задачі, було прийнято рішення розробити класс Playlist із наступними методами. Та тип данних (стркутуру) MusicObject.

MusicObject\* SearchID(int);

Шукає обєкт у стеку із заданним id. Та повертає на нього вказівник.

void CountRecords();

Підраховує кількість записів у стеку.

void OutRecords();

Виводить усю інформацію зі стеку.

void DeleteID(int);

Виключає обєкт зі стеку.

void Wipe();

Видаляє всю інформацію про поточний плейліст

String SearchParametr(String); MusicObject Fill(int); void FillFromFile();

FillFromFile заповню кожний елемент стеку з використанням функції Fill що впроваджує необіхдним обєктом. В свою чергу Fill – створю обєкт и заповнює його використовую функцію SearchParametr – де параметр це інформація що шукати у файлі.

void Save();void Save(String);

Зберігає плейліст до файлу, або викликає вікно для вибору місця збереження.

void Add(MusicObject\*);

Додає новий музикальних обєкт до стеку.

void RebuildID();

Перераховує і корректує ID згідно порядку наростання, без пропусків. Необхідно після видалення, або додавання.

void BlackOut();

Внеклассова функція, що робить недоступним усі компоменти. Використовується підчас свторення нового плейлісту

Функція зміна імені плейлісту:

Фунція змінює імя цього плейлісту, та його відображення у заголовку головного вікна.

Функція відкриває вікно для вибору файла до зчитування. Потім програма заповнює стек музикальной списку за допомгою іншої функціїї FillFromFile()

Функція записує усю інформацію зі стеку до файлу, із форматом Title(Ідентифікаційний номер елементу)=(Імя елементу) File(Ідентифікаційний номер елементу)=(Адреса елементу)

### 2.2.3 Ергономіка інтерфейсу користувача

Ергономіка інтерфейсу користувача виглядає наступним чином.

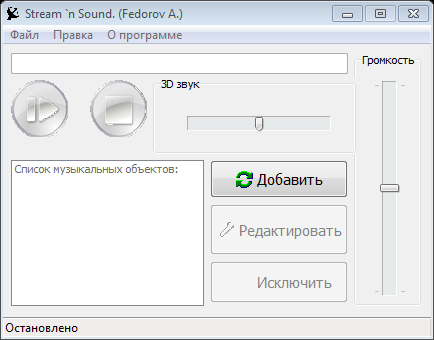


Рис. 7- Головне вікно програми.

При запуску програми користувач відразу повинен вибрати плейліст з яким буде працювати. Вікно вибору файлу є стандартним. Майже усі програми використовують саме цей стиль вибору файлу, тому користувач вже буде знати усі можливості цього діалогового вікна.

Можна також створити новий плейіст без завантаження файлу, для цього треба натисниту кнопку «Добавить» або Створити плейліст. У першому випадку користувач побаче вікно з питанням про створення нового плейлісту.

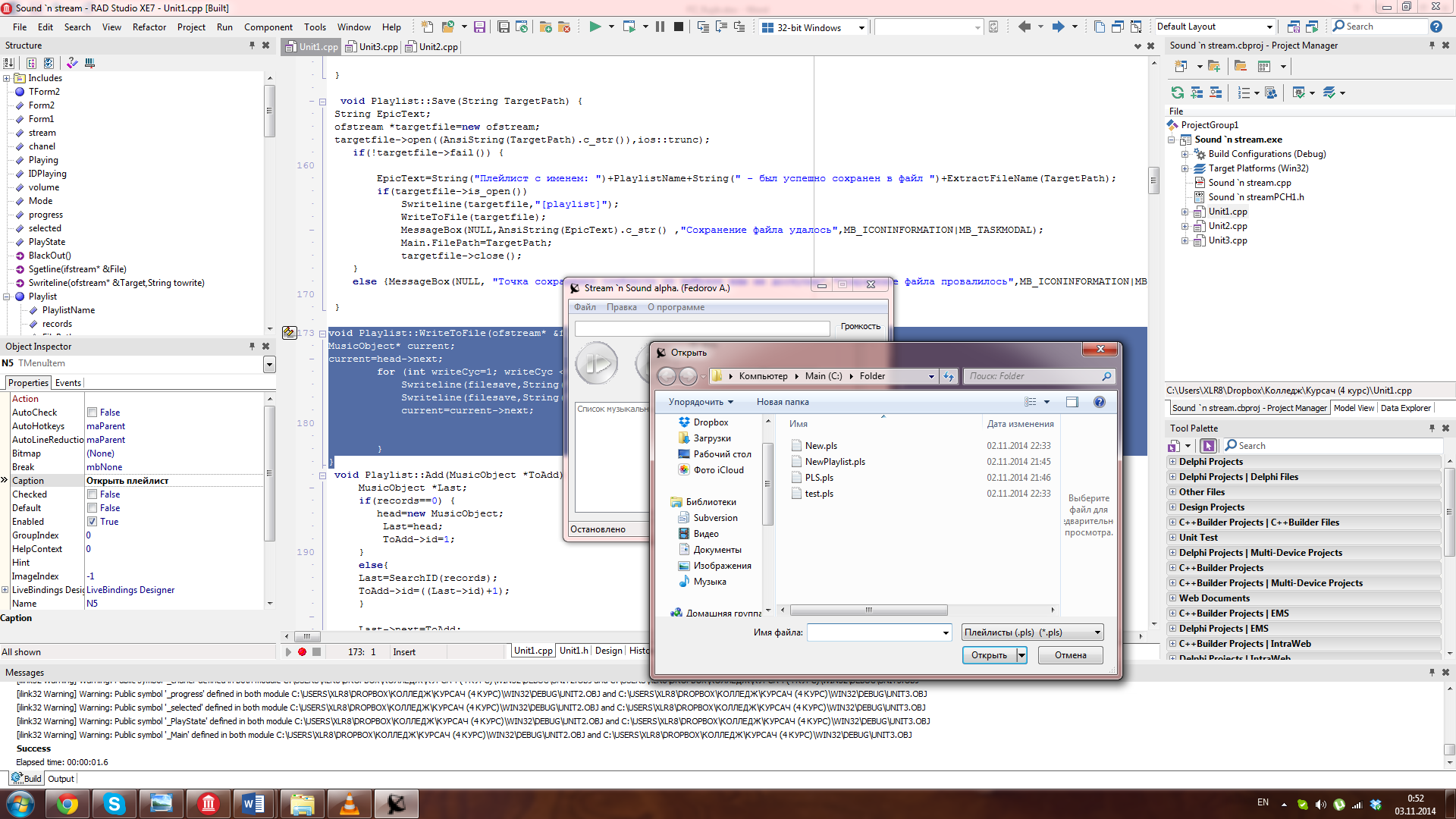


Рис. 8- Віно вибору файла

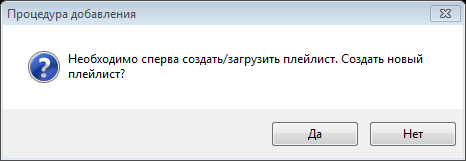


Рис. 9- Створення нового плейлісту

Вікно додавання та редагування елементів виглядають однаково, тому що необхідна інформації однакова. Якщо вибране редагування то до полів заноситься інформації про виділений об’єкт, це дозволяє біль наглядно користувачу побачити, що він редагує.

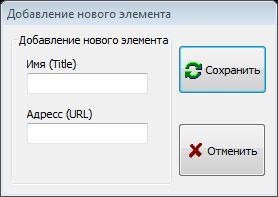


Рис. 10- Кнопки дій

Головне меню програми має меню що знаходиться зверху, і має схожість з будь якої іншою програмою.

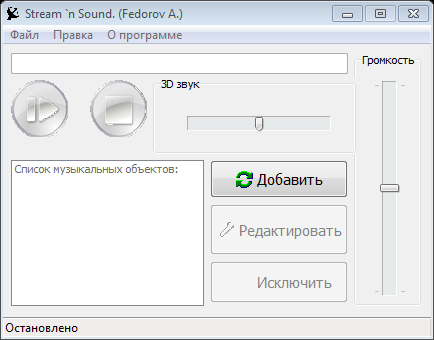
ї

Рис. 11 - Головне меню

Великі кнопки керування є найважливішими елементами тому вони великі. Список музикальних об’єктів зроблений великим, щоб умістити велику кількість об’єктів, але якщо цього не достатньо він має здатність розшитись додавши до себе повзунок прокрутки.

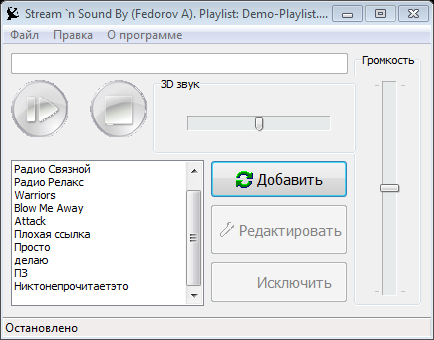


Рис. 12 - Список музикальних обєктів

Три кнопки – «Добавить», «Редактировать», «Исключить» знаходяться не далеко від списку музикальних об’єктів, тому що напряму залежать від вибору у ньому.

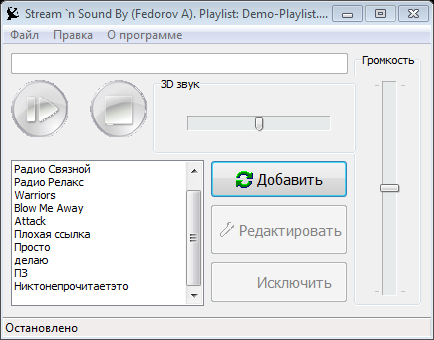


Рис. 13 - Кнопки редагування плейлісту

Гучність займає праву частину по всій висоті програми, щоб користувачу було простіше точніше нагладжувати звук саме під його потреби

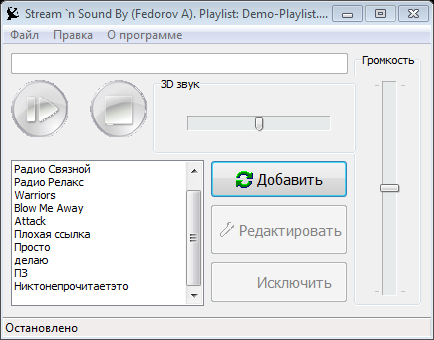


Рис. 14 - Панель гучності

## 2.3 Засоби візуального проектування ПЗ

Засобами візуального проектування в Embarcadero Rad Studio XE7, перш за все є його візуальні компоненти.

Далі будуть розглянуті основні типи візуальних компонентів, які були застосовані для розробки даного програмного забезпечення.

Вкладка Standart.

На цій вкладці розташовані стандартні компоненти , такі як кнопки, фрейми, меню і т . Д .

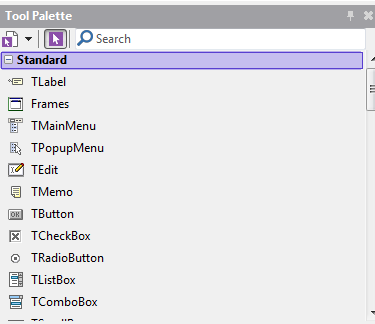


Рис. 15 - Компоненти панелі Standart

* MainMenu (Головне меню) - призначений для створення головного

меню програми. З цим та іншими компонентами ми познайомимося ближче в процесі створення нових програм.

* PopupMenu (Спливне меню) - призначений для створення

спливаючого меню деяких компонентів. Зазвичай за допомогою цього ком-компонента створюється контекстне меню.

* Label (Етикетка) - створює на формі текстову мітку або напис.
* Edit (Редагування) - створює на формі поле для редагування тек

текстового рядка.

* Memo (Поле) - відображає на формі поле для редагування текстових

рядків. Зазвичай служить для створення редакторів і полів для виводу блоків даних.

* Button (Кнопка) - є найпоширенішим компонентом. Служить для

створення в додатку різних прямокутних кнопок з текстовим написом.

* CheckBox (Осередок стану) - дозволяє створювати на формі додатка

осередок з двома станами (без галочки і з галочкою) і рядком назви. Клацання лівою кнопкою миші по цьому компоненту під час роботи програми викликає зміну стану компонента на протилежне. В програмі завжди можна дізнатися стан цього компонента і тим самим виконувати ту чи іншу дію.

* RadioButton (Радіокнопка) - створює кругле поле з двома станами (з

точкою і без крапки) і текстовим рядком, яка б пояснила її призначення в програмі. Зазвичай кілька таких компонентів, розташованих на формі, дозволяють перемкнути тільки один елемент з групи.

* ListBox (Вікно списку) - створює прямокутне поле для відображення

текстових рядків з можливістю їх вибору, додавання або видалення при роботі програми.

* ComboBox (Комбінований список) - дозволяє створювати на формі

елемент, що є комбінацією рядка введення і списку, що випадає для вибору. Фактично об'єднує в собі компоненти ListBox і Edit.

* ScrollBar (Лінійка прокрутки) - створює елемент, схожий на лінійку з

бігунком і кнопками для прокрутки вікна, до якого відноситься цей елемент. Крім того, з його допомогою можна змінювати в межах деякого заданого інтервалу значення величини якого-небудь параметра.

* GroupBox (Вікно групи) - служить для створення області, візуально

об'єднуючою на формі декілька інтерфейсних елементів.

* RadioGroup (Група радіокнопок) - дозволяє створювати на формі

контейнер у вигляді прямокутної рамки для об'єднання групи взаємовиключних радіокнопок.

* Panel (Панель) - створює порожню область, на якій можна розмістити

інші компоненти. Як правило, використовується для створення панелі інструментів в програмі.

* Action List (Список дій) - здійснює управління взаємодією між

інтерфейсними елементами і логікою програми.

# 3. ЗАКЛЮЧНА ЧАСТИНА

## 3.1 Відлагодження і тестування програмного продукту

Debug - Програма-відладчик, яку використовують для перевірки і налагодження виконуваних файлів. Використовувалася при операційній системі MS-DOS. Під пізніші версії операційних систем працює через емулятор MS-DOS і має обмежені можливості. Також іноді називають процес налагодження програми.

Дана програма є консольним додатком і призначена для створення або зміни коду файлів. За допомогою неї можна створювати прості програми під MS-DOS і відстежувати їхню роботу. Даний відладчик знаходиться на найнижчому рівні компіляторів assembler. Але володіє непоганими можливостями такими як перегляд, зміна пам'яті та отримання стану регістрів.

Відкладка у Embarcadero Rad Studio починається з виставлення BreakPoint у коді програмии. Натиснувши лівіше від коду, біля нумерації строк – зявляеться червона точка.

Тому програма, після запуску, припинить виконання коду точно на початку рядка і чекатиме дій з налагодження.

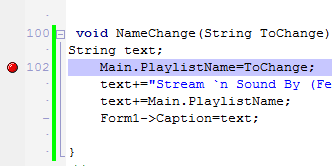


Рис. 16 -Додавання BreakPoint

Дії що зазвичай створюються під час налогдження:

* Покрокове виконання програми. Для цього слід натиснути кнопку Step Over на панелі керування або клавішу клавіатури F8. Програма виконуватиметься через підрядник. Функції користувача будуть виконуватися цілком. Для построчного виконання внутрішнього коду функцій користувача слід натиснути кнопку Trace Into на панелі керування або клавішу клавіатури F7. Тоді покрокове виконання включатиме і рядки функцій;
* Продовження виконання програми до кінця. Для цього слід натиснути кнопку запуску програми (зелений трикутник) або клавішу F9 (до рядка з курсором F4);
* Введення додаткових точок зупину. Після введення натискається кнопка запуску для досягнення наступного зупину або кінця програми;

Під час переривання програми в точці зупинки можна переглядати вміст змінних і змінювати їх значення. Перегляд зручно виконувати, зупиняючи курсор миші над відповідної змінної в тексті програми.

Тестування програмного продукту:

Завдання 1. Відкрити файл плейлістів, та відворити радіо-поток.

Після натискання кнопки Файл-открыть плейлист, зявилося віно в якому користувачу необхідно вибрати файл. Вибір файлів обмежений лише форматом .pls

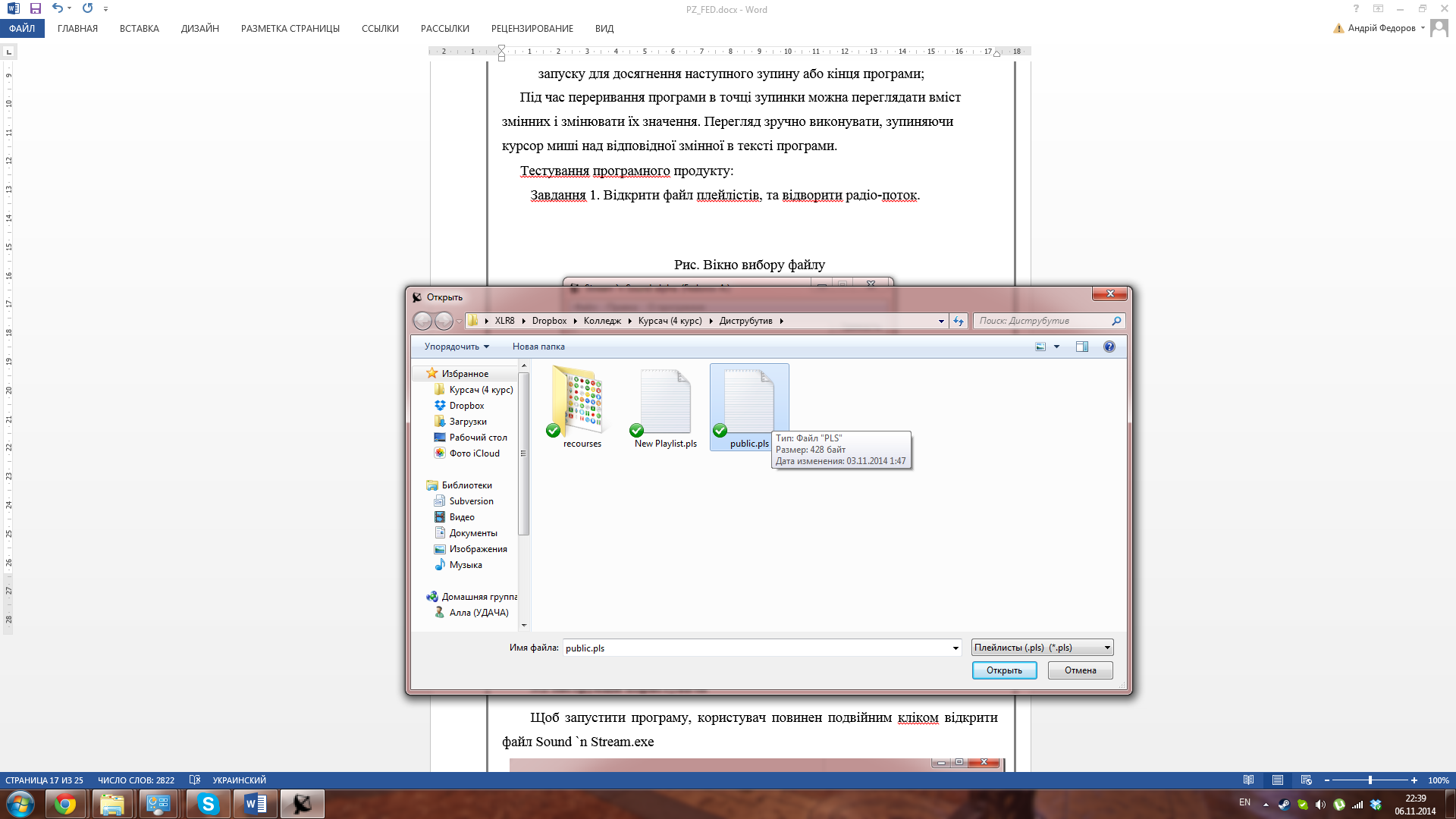


Рис. 17 - Вікно вибору файлу

Якщо користувач спробує вибрати некоректний файл, або файл .pls с некоректним змістом – программа повідомить про помилку.

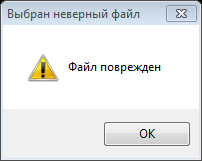


Рис. 18 - Помилка при виборі неправильного файлу

Вибравши корректний плейліст, користувач матиме список відтворюваних обєктів. Вибраши один із них, і натиснувши кнопку відвторення почнеться відтворення.

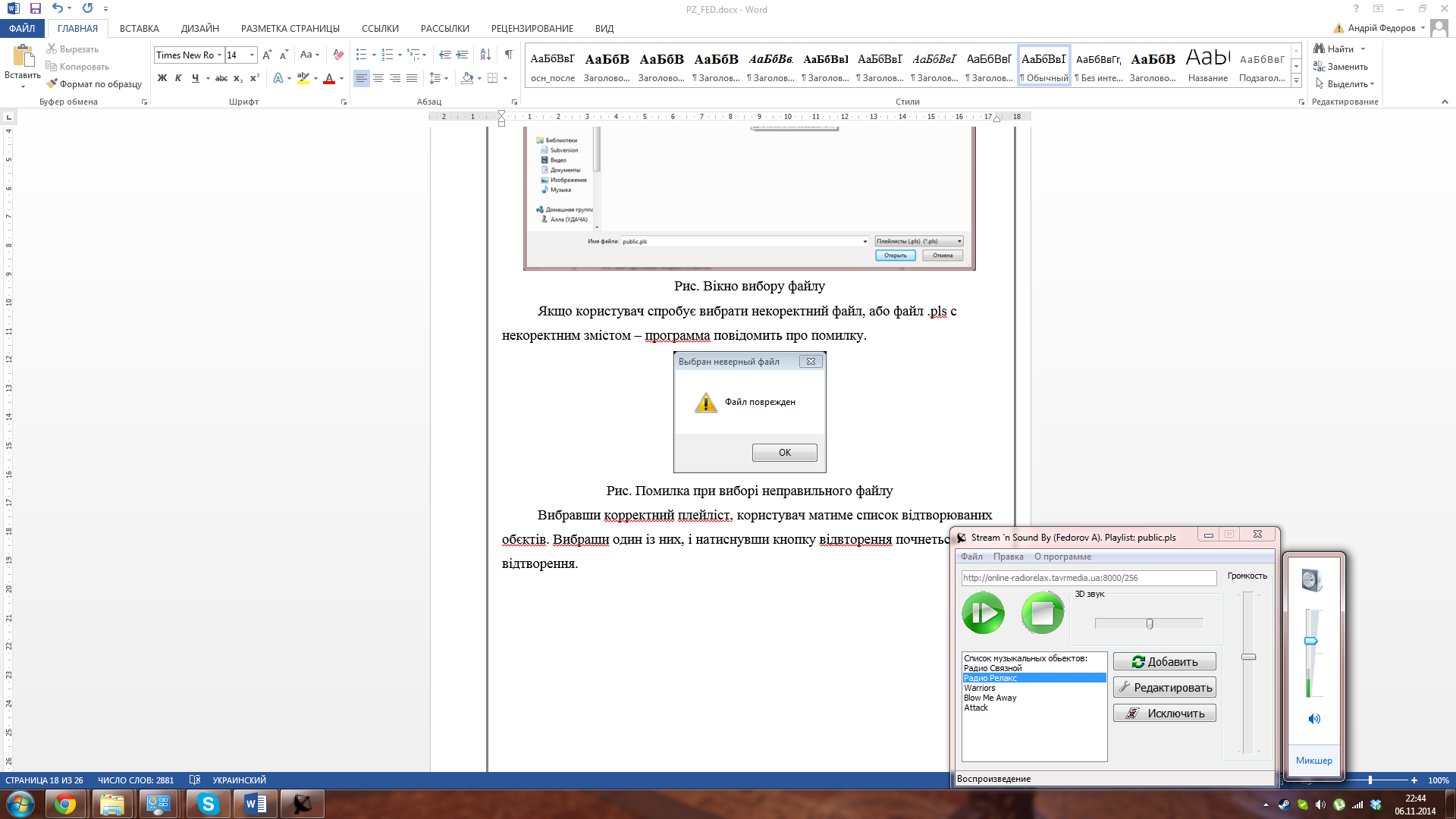


Рис. 19 - Відтворення музикального обєкту

Завдання 2. Змінити плейліст, відредагувавши та виключивши деякі обєкти, потім зберегти у новому плейлісті.

Вибравши музикальний обєкт у списку, кнопки «Редактировать» та «Исключить» становляться активними, що дозволяють працювати над цими обєктами. Натиснувши на кнопку редагувати, ми отримуємо форму для редагування, в якій вже заповнені дані о вибраном обєкті. Обовязком полем є URL (Адреса) , якщо його не заповнити – виникне помилке.

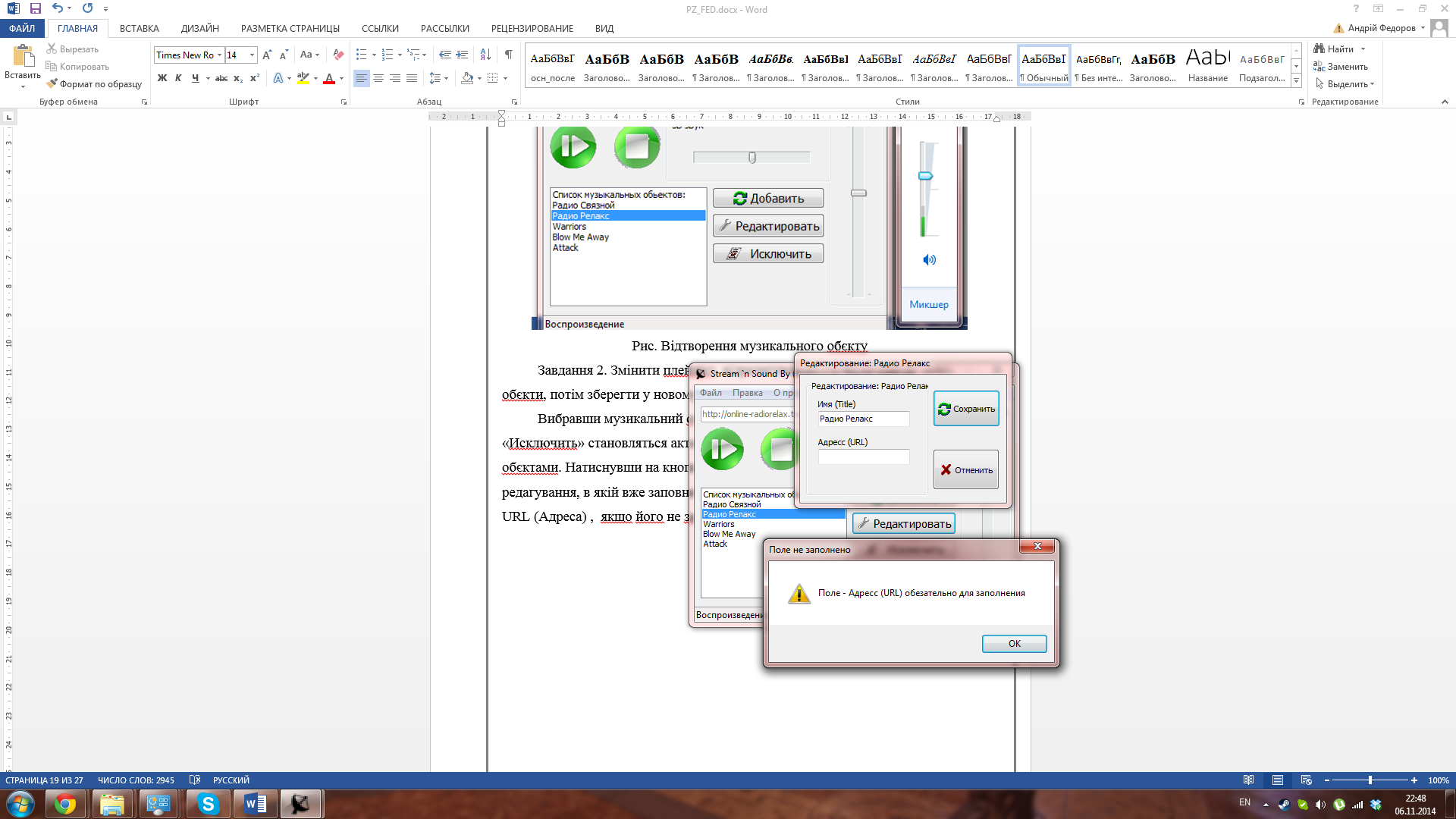


Рис. 20 - Форма редагування.

Видаленне може бути здіснене, завдяки виділенню будь якого обєкту і натискання кнопки «Исключить».

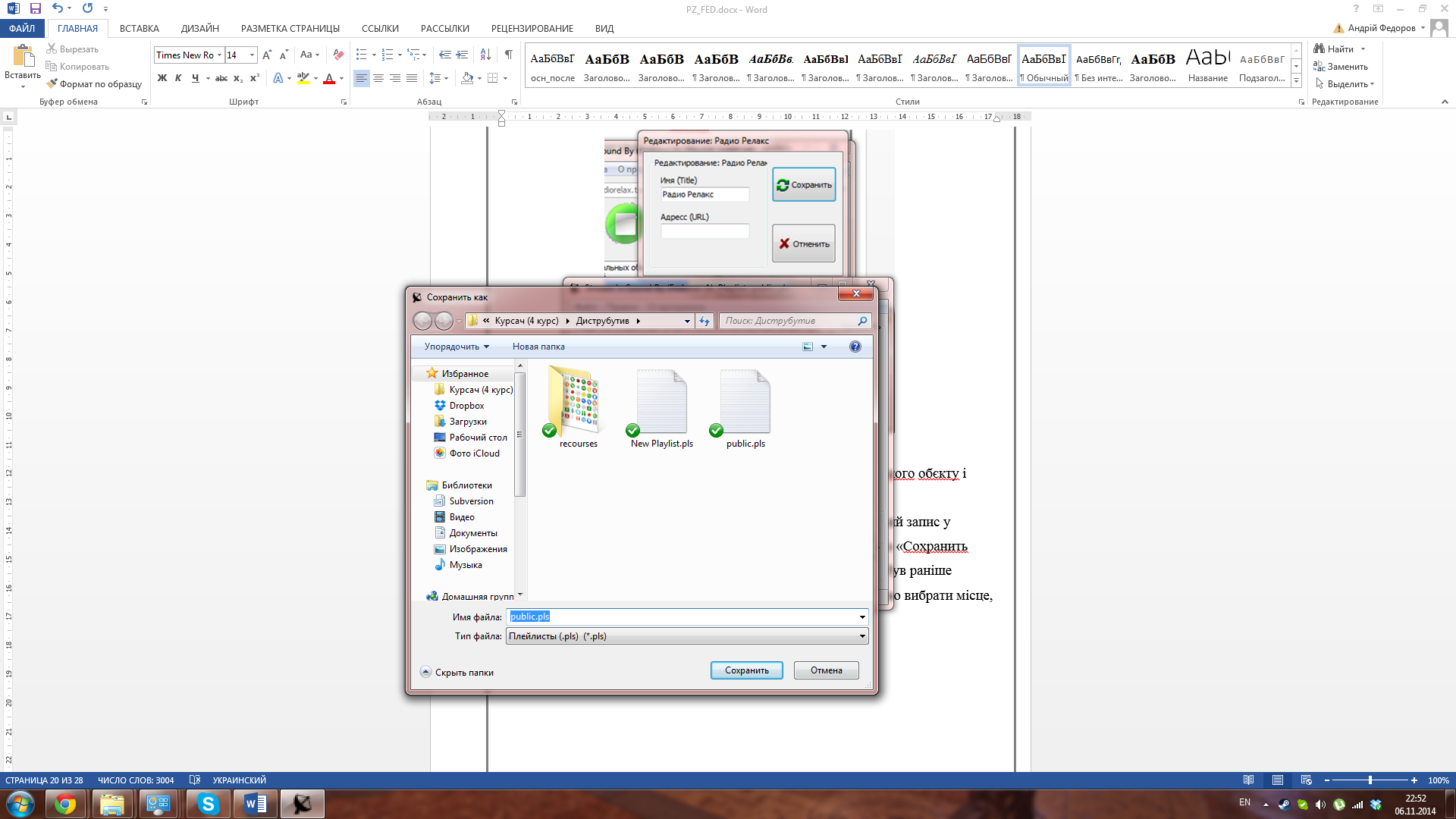


Рис. 21 - Зберігання плейлісту

Збергіання можливе якщо залившись хоч один музикальний запис у плейлісті. Зберігання можливе у двох варіантах. «Сохранить» та «Сохранить как». У першому випадку обєкт зберігаеться до того файлу що був раніше відкритий/збережений. У іншому випадку користувачи необхідно вибрати місце, та імя для нового плейлісту.

## 3.2 Інструкція користувача

Щоб запустити програму, користувач повинен подвійним кліком відкрити файл Sound `n Stream.exe

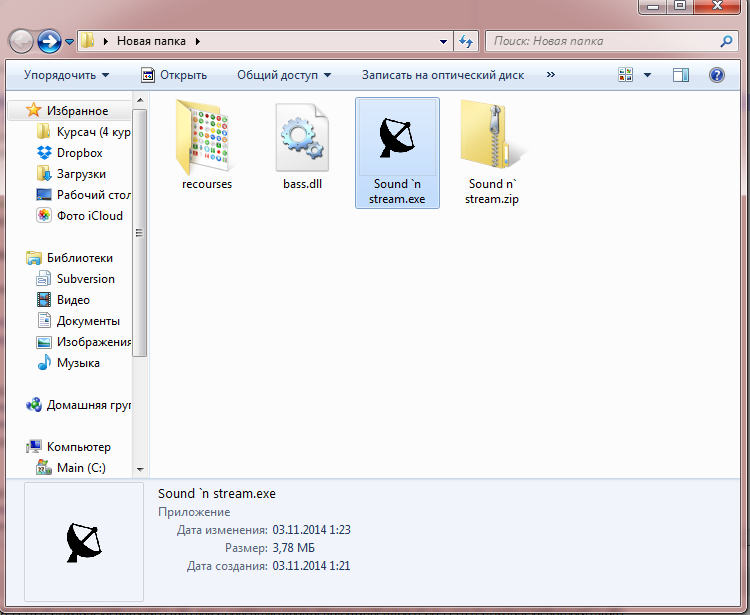


Рис. 22 - Файл Project1.exe

Після запуску програми користувач побаче головне вікно програми

Для початку роботи необхідно завантажити, або створити плейліст.

Завантажити плейліст можливно натиснувши (Файл->Открыть Плейлист), далі користувачу необхідно буде вказати файл формату вказати файл формату (.pls)

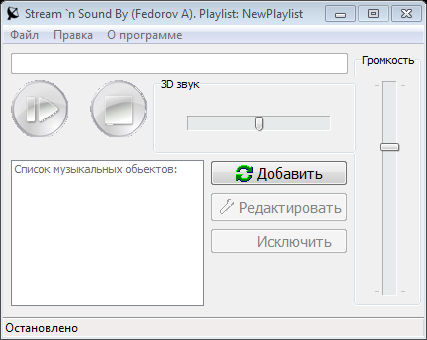


Рис. 23 - Головне вікно програми

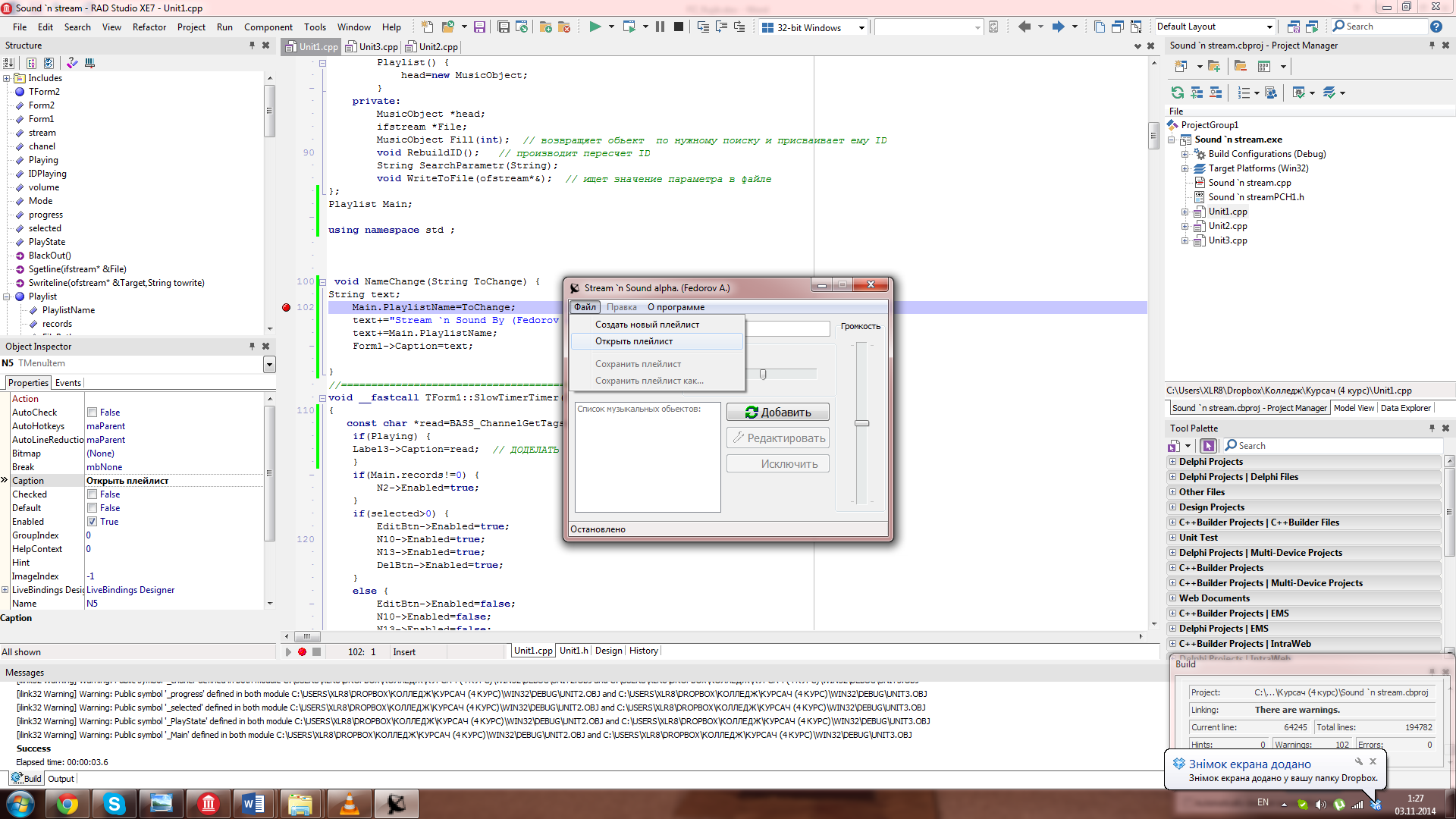


Рис. 24 -Кнопка «Открыть плейлист»

Якщо необхідно створити новий плейліст, то виберіть Файл-> «Создать новий плейлист», або додайте новий запис за домогою кнопки додати, та пітвердіть вашу дії, щодо створення нового плейлісту

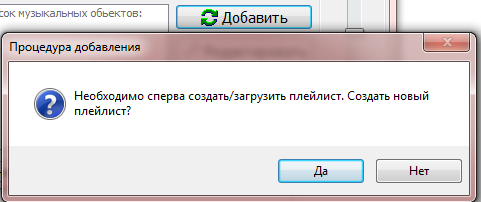


Рис. 25 - Запит створення нового плейлісту

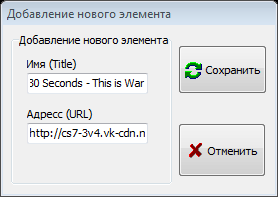


Рис. 26 -Додавання нового елементу

Збереження файлу проходить за допомогою натискання файл Файл->Сохранить, або Файл->Сохранить как.

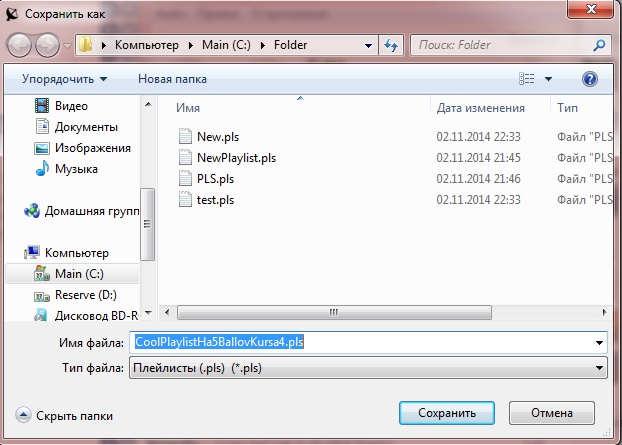


Рис. 27 - Вікно збереження плейлісту

Перший збергає у файл, якщо плейліст вже був збережений або зчитаний з нього, Другий дозволяє вибрати куди зберегти файл.

Відтворення виконується за допомогою кнопок керування. Грати та Зупинити/Пауза.



Рис. 28 - Кнопки керування

Для того щоб вибрати інший обєкт з плейлісту для відтворення, клацніть по ньому зі списку, та нажміть кнопку відтворення

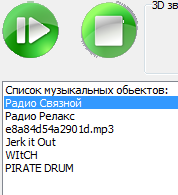


Рис. 29 - Вибір об’єкта

## 3.3 Розробка дистрибутивного диску

Smart Install Maker - це умовно-безкоштовна утиліта з закритим вихідним кодом, яка надає користувачам потужний і простий у використанні інструмент для створення високоефективних інсталяторів в 32-бітових операційних системах Microsoft Windows.

Smart Install Maker - потужна і зручна програма для створення інсталяторів. Володіє простим і інтуїтивно зрозумілим інтерфейсом, а також повним набором необхідних функцій для створення професійних інсталяторів з мінімальним розміром, високим ступенем стиснення файлів і приємним інтерфейсом. Підтримують чудовий формат стиснення Cab, що дозволяє створювати дійсно компактні дистрибутиви.

Крім стандартного мінімуму, Smart Install Maker дозволяє редагувати системний реєстр і INI-файли, створювати програмні ярлики, запускати асоційовані і виконувані файли, реєструвати нові шрифти і ActiveX компоненти, відображати тексти інформації і ліцензійної угоди. Серед інших можливостей Smart Install Maker, можна виділити створення дєїнсталлятора, ярликів в зазначених директоріях системи, реєстрацію нових шрифтів або розширень в операційній системі, підтримку ActiveX і змінних, перевірку встановленого .NET Framework, поділ інсталятора на настановні диски, детальну настройку інформації про установочном файлі , системних вимог (наприклад, заборонити установку програми в Windows 95) а також його зовнішнього вигляду та багато іншого.

Використовуючи Smart Install Maker був створений інсталяційний пакет.

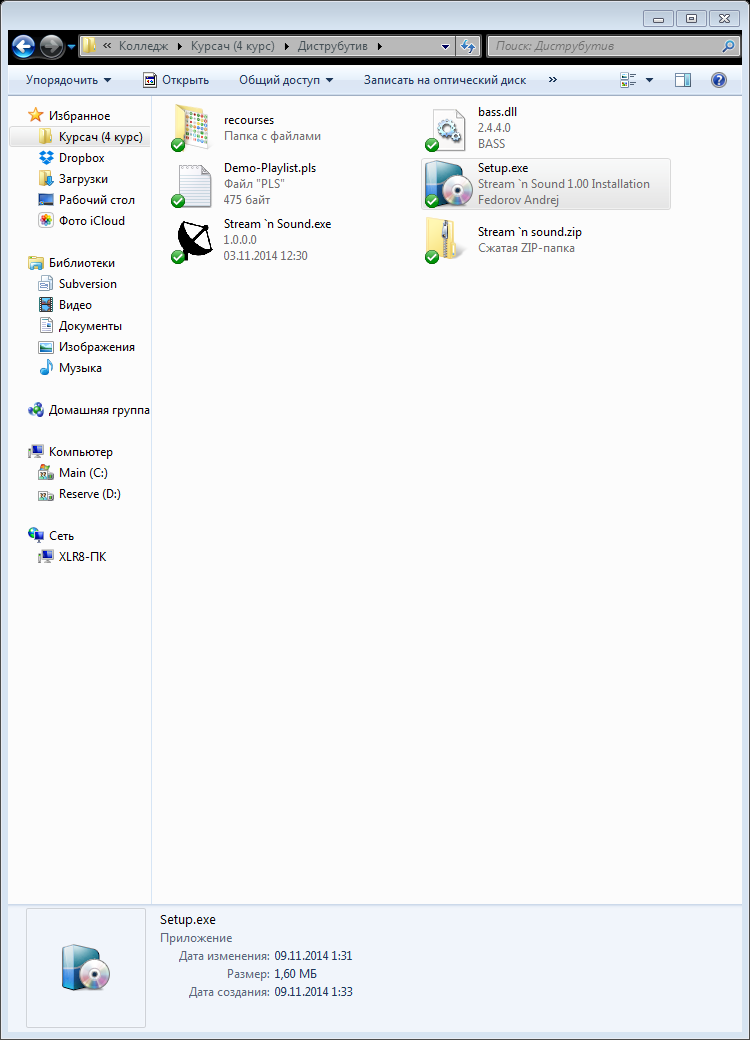


Рис. 30 - Програма-інсталятор.

Щоб почати встановлення програми, необхідно запустити програму-інсталятор. Вибрати теку куди буди встановлена програма.

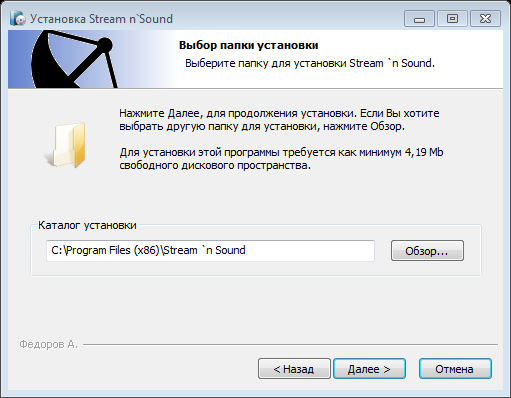


Рис. 31 - Вибір папки інсталяції

Після прогресу інсталяції, ви можете запустити програму заливши помітку біля пункту «Запустить Stream `n Sound».

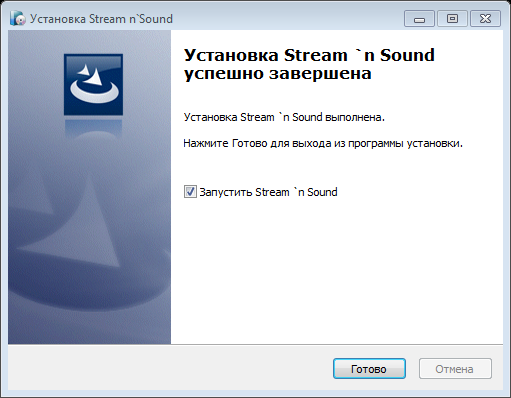


Рис. 32- Завершення процесу інсталяції

## Висновок

Під час виконання задачі, був розроблений аудіо програвач на операційну систему Windows, який здатен відтворювати музикальні файли, що знаходяться віддалено, та прослуховувати непереривні інтернет потоки.

Була реалізована функція праці з плейлістами формату (.pls) згідно з вимог форматування цього типу, що дозволяє іншим плеЄрам також считувати плейлісти який створив даний програмний продукт, та навпаки. Для відтворення музикальних файлів, присутня можливість перомутвання поточної позиції у аудіофайлі.

Інтерфейс програмного продукту був виконаний для звичайної аудиторії користувачів, без необхідності додатовких навичок у використанні.

## Список використаної літератури

1. Архангельский А.Я. «Программирование в C++ Builder», Москва, «Бином», 2010г, -1304с
2. Гради Буч «Объектно-ориентированный анализ и проектирование», «Невский Диалект», Москва, 2000г, -359с
3. Дейтел Х. «Как программировать на С++», Москва, «Бином», 2003г,-1152 с
4. Лаптев В.В « С++. Объектно-ориентированное программирование», Санкт-Петербург, «БХВ-Петербург», 2007 г, -288 с.
5. Прата Стивен «Язык программирования C++.», Москва, «Вильямс», 2012 г, -400 с
6. Романов Е.Л «Практикум по программированию на С++», Санкт-Петербург, «БХВ-Петербург», 2004 г, -678 с.

## Додаток А. Код програми

#include <vcl.h>

#include "bass.h"

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include "Unit3.h"

#include "Unit4.h"

#include <iostream>

#include <iostream.h>

#include <fstream>

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <stdio.h>

#include <sysmac.h>

#include <string>

#include <clocale>

#include <locale>

#include <windows.h>

#include <algorithm>

#include <math.h>

USEFORM("Unit2.cpp",Form2);

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma comment (lib, "bass.lib")

#pragma resource "\*.dfm"

TForm1 \*Form1;

HSTREAM stream;

HCHANNEL chanel;

int Playing=false;

int IDPlaying=0;

float volume=5;

String Mode="";

int progress;

int selected;

String PlayState;

bool Failcheck=0;

void BlackOut();

//=====================================================

String Sgetline(ifstream\* &File) {

char read[256];

String result;

File->getline(read,256);

result=read;

return result;

}

//=====================================================

void Swriteline(ofstream\* &Target,String towrite) {

char WriteVal[256];

AnsiString sText = towrite;

strcpy( WriteVal, sText.c\_str() );

Target->write(WriteVal,towrite.Length());

sText="\n";

strcpy( WriteVal, sText.c\_str() );

Target->write(WriteVal,sText.Length());

}

class Playlist {

public:

String PlaylistName;

unsigned int records;

String FilePath;

MusicObject\* SearchID(int); // ищет обьект в стеке

void CountRecords(); //подсчитывает количество записей в файле

void OutRecords(); // выводит все

void DeleteID(int); // удаляет обьект из стека и пересчитывает ИД

void Wipe(); // стриает все в этом плейлисте

void FillFromFile(); //заполняет стек

void New(String,String); // новый плейлист

void Add(MusicObject\*); // новый музыкальный обьект

void Save(); // сохраняет с окном выбора

void Save(String); // сохраняет в выбраном пути без окна

Playlist() {

head=new MusicObject;

}

private:

MusicObject \*head;

ifstream \*File;

MusicObject Fill(int); // возвращяет обьект по нужному поиску и присваивает ему ID

void RebuildID(); // производит пересчет ID

String SearchParametr(String); // ищет в файле строку, и выделяет нужное

void WriteToFile(ofstream\*&); // записывает весь стек в файл

};

Playlist Main;

using namespace std ;

void NameChange(String ToChange) {

String text;

Main.PlaylistName=ToChange;

text+="Stream `n Sound By (Fedorov A). Playlist: ";

text+=Main.PlaylistName;

Form1->Caption=text;

}

//=====================================================

void \_\_fastcall TForm1::SlowTimerTimer(TObject \*Sender)

{

if(Main.records!=0) {

N2->Enabled=true;

}

if(selected>0) {

EditBtn->Enabled=true;

N10->Enabled=true;

N13->Enabled=true;

DelBtn->Enabled=true;

}

else {

EditBtn->Enabled=false;

N10->Enabled=false;

N13->Enabled=false;

DelBtn->Enabled=false;

}

}

//=====================================================

void Playlist::Save() {

String TargetPath;

String EpicText;

ofstream \*targetfile=new ofstream;

Form1->SaveDialog1->Filter="Плейлисты (.pls) |\*.pls";

Form1->SaveDialog1->DefaultExt=".pls";

Form1->SaveDialog1->FileName=PlaylistName;

if(Form1->SaveDialog1->Execute()) {

TargetPath=Form1->SaveDialog1->FileName;

targetfile->open((AnsiString(TargetPath).c\_str()),ios::trunc);

EpicText=String("Плейлист с именем: ")+PlaylistName+String(" - был успешно сохранен в файл ")+ExtractFileName(TargetPath);

if(targetfile->is\_open())

Swriteline(targetfile,"[playlist]");

WriteToFile(targetfile);

MessageBox(NULL,AnsiString(EpicText).c\_str() ,"Сохранение файла удалось",MB\_ICONINFORMATION|MB\_TASKMODAL);

Main.FilePath=TargetPath;

targetfile->close();

}

else {MessageBox(NULL, "Точка сохранения плейлиста не выбрана или недоступна","Сохранение файла провалилось",MB\_ICONINFORMATION|MB\_TASKMODAL); return; }

}

//=====================================================

void Playlist::Save(String TargetPath) {

String EpicText;

ofstream \*targetfile=new ofstream;

targetfile->open((AnsiString(TargetPath).c\_str()),ios::trunc);

if(!targetfile->fail()) {

EpicText=String("Плейлист с именем: ")+PlaylistName+String(" - был успешно сохранен в файл ")+ExtractFileName(TargetPath);

if(targetfile->is\_open())

Swriteline(targetfile,"[playlist]");

WriteToFile(targetfile);

MessageBox(NULL,AnsiString(EpicText).c\_str() ,"Сохранение файла удалось",MB\_ICONINFORMATION|MB\_TASKMODAL);

Main.FilePath=TargetPath;

targetfile->close();

}

else {MessageBox(NULL, "Точка сохранения плейлиста не выбрана или не доступна","Сохранение файла провалилось",MB\_ICONINFORMATION|MB\_TASKMODAL); return; }

}

//=====================================================

void Playlist::WriteToFile(ofstream\* &filesave) {

MusicObject\* current;

current=head->next;

for (int writeCyc=1; writeCyc <= records; writeCyc++) {

Swriteline(filesave,String("Title")+String(current->id)+String("=")+current->Name);

Swriteline(filesave,String("File")+String(current->id)+String("=")+current->URL);

current=current->next;

}

}

//=====================================================

void Playlist::Add(MusicObject \*ToAdd) {

MusicObject \*Last;

if(records==0) {

head=new MusicObject;

Last=head;

ToAdd->id=1;

}

else{

Last=SearchID(records);

ToAdd->id=((Last->id)+1);

}

Last->next=ToAdd;

records++;

RebuildID();

OutRecords();

Form1->StopButton->Enabled=true;

Form1->PlayButton->Enabled=true;

Form1->ItemsHolder->Enabled=true;

Form1->N6->Enabled=true;

Form1->N7->Enabled=true;

}

//=====================================================

void Playlist::New(String PlayName,String InFilePath) {

String FormCaption;

BASS\_StreamFree(stream);

Playing=0;

FormCaption+="Stream `n Sound By (Fedorov A). Playlist: ";

BlackOut();

PlaylistName=PlayName;

FilePath=InFilePath;

FormCaption+=PlayName;

Form1->Caption=FormCaption;

}

//=====================================================

void Playlist::RebuildID() {

MusicObject\* Reading;

int count;

Reading=head->next;

for(count=1;count<=records;count++)

{

Reading->id=count;

Reading=Reading->next;

}

}

//=====================================================

void Playlist::OutRecords() {

MusicObject \*q;

Form1->ItemsHolder->Clear();

int count;

q=head->next;

Form1->N10->Clear();

Form1->ItemsHolder->Clear();

Form1->ItemsHolder->Items->Add("Cписок музыкальных объектов:");

for(count=1;count<=records;count++) {

Form1->ItemsHolder->Items->Add(q->Name);

q=q->next;

}

}

//=====================================================

void Playlist::DeleteID(int id) {

MusicObject\* prev;

MusicObject\* tobedeleted;

MusicObject\* temp;

if(id==IDPlaying) {

Playing=0;

BASS\_StreamFree(stream);

Form1->Height=340;

}

if(id<=records) {

if(id==1) {

prev=head;

}

else {

prev=SearchID((id-1));

}

tobedeleted=SearchID(id);

prev->next=tobedeleted->next;

records--;

RebuildID();

OutRecords();

if(records==0) {

Form1->N6->Enabled=false;

Form1->N7->Enabled=false;

}

}}

//=====================================================

MusicObject\* Playlist::SearchID(int IDReq) {

unsigned int cycle=1;

int IDobject;

MusicObject\* Checking=new MusicObject;

Checking=head;

do{

Checking=Checking->next;

IDobject=Checking->id;

if(cycle>records) {

ShowMessage("Что то пошло не так. Количество попыток больше чем самая большая запись");

break;

}

cycle++;

} while(IDobject!=IDReq);

return Checking;

}

//=====================================================

void Playlist::FillFromFile() {

unsigned int count=1;

head=new MusicObject;

MusicObject\* q=new MusicObject;

MusicObject t;

head->next=q;

for(count=1;count<=records;count++) {

t=Fill(count);

if(!t.bad) {

q->id=t.id;

q->Name=t.Name;

q->URL=t.URL;

q->bad=t.bad;

q->next=new MusicObject;

}

q=q->next;

}

OutRecords();

Form1->ItemsHolder->Enabled=True;

Form1->N6->Enabled=true;

Form1->N7->Enabled=true;

Form1->SlowTimerTimer(Form1);

}

//=====================================================

void Playlist::CountRecords() {

int max=0;

String SearchLabel;

String SearchResult;

do{

max++;

SearchLabel="File"; SearchLabel+=max;

SearchResult= SearchParametr(SearchLabel);

} while(SearchResult!="===NOTHING===");

max--;

records=max;

}

//=====================================================

MusicObject Playlist::Fill (int id) {

MusicObject Returning;

String SearReq;

String func;

int Clen;

Returning.id=id;

SearReq="File"; SearReq+=id;

func=SearchParametr(SearReq);

if(func=="===NOTHING===") {

Returning.bad=true;

return Returning;

}

Returning.URL=func;

SearReq="Title"; SearReq+=id;

func=SearchParametr(SearReq);

if(func=="===NOTHING===") {

Clen=Returning.URL.Length();

Returning.Name=Returning.URL.SubString(Clen-17,Clen);

}

else{Returning.Name=func;}

Returning.bad=false;

Returning.id=id;

return Returning;

}

//=====================================================

String Playlist::SearchParametr(String Par) {

delete File;

File=new ifstream;

File->open(AnsiString(FilePath).c\_str());

int dlina;

Par=Par+"=";

dlina=Par.Length();

String line;

String Cutted;

String result;

int LineLen;

int moretest;

bool Cycle;

bool SEOF;

int tries=0;

do {

tries++;

SEOF=File->eof();

if(SEOF) {File->close(); return "===NOTHING===";}

line=Sgetline(File);

LineLen=line.Length();

Cutted= line.SubString(0,Par.Length());

Cycle=(Cutted!=Par);

}while (Cycle);

if (LineLen>Par.Length()) {

result= line.SubString((dlina+1),100);

return result;

}

else{File->close(); return "===NOTHING===";}

File->close();

return "===NOTHING===";

}

//=====================================================

void Playlist::Wipe() {

for (int count=records;count>0; count--) {

DeleteID(records);

}

head->next=NULL;

PlaylistName="";

FilePath="";

Form1->N6->Enabled=false;

Form1->N7->Enabled=false;

}

//=====================================================

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

void PlayButtonSet (String state) {

if(Form1->Edit1->Text=="") {

Form1->PlayButton->ShowHint=True;

Form1->PlayButton->Picture->LoadFromFile("recourses\\img\\false\_play.png");

}

else{

Form1->PlayButton->Picture->LoadFromFile("recourses\\img\\"+state+"\_play.png");

Form1->PlayButton->ShowHint=False;

}}

//=====================================================

void StopButtonSet (String state) {

if(Playing==0)

{

if(BASS\_ChannelBytes2Seconds(stream,Form1->Peremotka->Max)==-1) {

Form1->StopButton->Picture->LoadFromFile("recourses\\img\\false\_stop.png");

}

}

else{

Form1->StopButton->Picture->LoadFromFile("recourses\\img\\"+state+"\_stop.png");

}

}

//=====================================================

String TimeTransf(int seconds) {

int minutes=0;

String Result;

if(seconds>0) {

while(seconds>59) {

minutes++;

seconds-=60;

}

}

Result+=minutes/10;

Result+=minutes%10;

Result+=":";

Result+=seconds/10;

Result+=seconds%10;

return Result;

}

//=====================================================

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender) {

int len=BASS\_StreamGetFilePosition(stream, BASS\_FILEPOS\_END);

int buf=BASS\_StreamGetFilePosition(stream, BASS\_FILEPOS\_BUFFER);

BASS\_ChannelSetAttribute(stream,BASS\_ATTRIB\_VOL,volume);

volume= ((VolumeBar->Max)/10)-((VolumeBar->Position)/(VolumeBar->Max\*1.0));

progress=buf\*100.0/len;

Buffer->Position=progress;

if(Playing==1) {

Peremotka->Position=BASS\_ChannelGetPosition(stream,0);

}

if(BASS\_ChannelGetLength(stream,0)-1 <10000000000) {

Taskbar1->ProgressValue=BASS\_ChannelGetPosition(stream,0);

}

Label1->Caption=TimeTransf(BASS\_ChannelBytes2Seconds(stream,Peremotka->Position));

}

//=====================================================

void \_\_fastcall TForm1::Edit1Change(TObject \*Sender)

{

if(Edit1->Text=="") {

PlayButton->ShowHint=True;

PlayButton->Picture->LoadFromFile("recourses\\img\\false\_play.png");

}

else{

PlayButton->Picture->LoadFromFile("recourses\\img\\on\_play.png");

PlayButton->ShowHint=False;

}

}

//=====================================================

void \_\_fastcall TForm1::PlayButtonMouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

PlayButtonSet("off");

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::PlayButtonMouseUp(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int X, int Y)

{

PlayButtonSet("on");

}

//---------------------------------------------------------------------------

void ConfirmForm2 () {

int Clen;

Clen=Form2->URLText->Text.Length();

if(Clen>0) {

if (Mode.Compare("edit")==0) {

if(Form2->TitleEdit->Text.Length()==0) {

Form2->TitleEdit->Text=Form2->URLText->Text.SubString(Clen-17,Clen);

}

Send->Name= Form2->TitleEdit->Text;

Send->URL= Form2->URLText->Text;

}

else if(Mode.Compare("add")==0) {

if(Form2->TitleEdit->Text.Length()==0) {

Form2->TitleEdit->Text=Form2->URLText->Text.SubString(Clen-17,Clen);

}

Send->Name= Form2->TitleEdit->Text;

Send->URL= Form2->URLText->Text;

Main.Add(Send);

}

}

else{MessageBox(NULL, "Поле - Адрес (URL) обязательно для заполнения","Поле не заполнено",MB\_ICONWARNING|MB\_TASKMODAL);return;}

Form1->ItemsHolder->Clear();

Main.OutRecords();

Form2->Hide();

selected=0;

Form1->SlowTimerTimer(Form1);

}

void CALLBACK Check(const void \*buffer, DWORD length, void \*user)

{

Failcheck=0;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::PlayButtonClick(TObject \*Sender)

{

int ErrCode;

String Text;

Text=Edit1->Text;

if(Playing!=3|selected!=IDPlaying) {

Failcheck=1;

BASS\_Start();

BASS\_ChannelStop(chanel);

BASS\_StreamFree(stream);

char \*url=AnsiString(Text).c\_str();

stream=BASS\_StreamCreateURL(url, 0, BASS\_STREAM\_STATUS, Check, 0);

ErrCode=BASS\_ErrorGetCode();

if(ErrCode!=0) {

StatusBar1->Panels->Items[0]->Text=String(Main.SearchID(selected)->Name+" не может быть воспроизведена. проверьте правильность.") ;

Taskbar1->ProgressValue=Taskbar1->ProgressMaxValue;

Taskbar1->ProgressState=3;

StopButtonSet("false");

StopButton->Enabled=false;

return;

}

chanel=BASS\_ChannelPlay(stream,TRUE); // только в новом потоке (после очистки)

Playing=1;

TrayIcon1->BalloonTitle="Воспроизведение:";

TrayIcon1->BalloonHint= Main.SearchID(selected)->Name;

TrayIcon1->ShowBalloonHint();

StatusBar1->Panels->Items[0]->Text=String("Воспроизведение: ("+Main.SearchID(selected)->Name+")") ;

Taskbar1->ProgressValue=Taskbar1->ProgressMaxValue;

Taskbar1->ProgressState=1;

StopButton->Enabled=true;

}

if(Playing==3) {

BASS\_Start();

Playing=1;

StopButtonSet("on");

StopButton->Enabled=true;

IDPlaying= ItemsHolder->ItemIndex;

Taskbar1->ProgressState=2;

StatusBar1->Panels->Items[0]->Text=String("Воспроизведение: ("+Main.SearchID(selected)->Name+")") ;

return;

}

SlowTimer->OnTimer(Form1);

IDPlaying= ItemsHolder->ItemIndex;

if(Playing) {

if(BASS\_ChannelGetLength(stream,0)-1 <10000000000) {

Peremotka->Max=BASS\_ChannelGetLength(stream,0)-1 ;

Taskbar1->ProgressMaxValue=BASS\_ChannelGetLength(stream,0)-1 ;

PeremotkaGroup->Visible=True;

Form1->Height=430;

Taskbar1->ProgressState=2;

}

else{PeremotkaGroup->Visible=false;Form1->Height=340;}

}

Label2->Caption=TimeTransf(BASS\_ChannelBytes2Seconds(stream,Peremotka->Max));

Playing=1;

StopButtonSet("on");

}

//=====================================================

void \_\_fastcall TForm1::StopButtonMouseDown(TObject \*Sender, TMouseButton Button,

TShiftState Shift, int X, int Y)

{

StopButtonSet("off");

}

//=====================================================

void \_\_fastcall TForm1::SoundPosBarChange(TObject \*Sender)

{

float soundPos;

soundPos= ((SoundPosBar->Position))/10.0;

BASS\_ChannelSetAttribute(stream,BASS\_ATTRIB\_PAN,soundPos);

}

//=====================================================

void \_\_fastcall TForm1::PeremotkaScroll(TObject \*Sender, TScrollCode ScrollCode, int &ScrollPos)

{

BASS\_Pause();

BASS\_ChannelSetAttribute(stream,BASS\_ATTRIB\_VOL,0);

BASS\_ChannelSetPosition(stream, Peremotka->Position, 0);

if(Playing==1) {

BASS\_Start();

}

else{BASS\_Pause();}

}

//=====================================================

String Converter(string Normal) {

String result;

result=Normal.c\_str();

return result;

}

//=====================================================

void \_\_fastcall TForm1::ItemsHolderClick(TObject \*Sender)

{

if(selected== ItemsHolder->ItemIndex) { return; }

selected=ItemsHolder->ItemIndex;

if(selected>0) {

Edit1->Text=Main.SearchID(ItemsHolder->ItemIndex)->URL;

EditBtn->Enabled=true;

DelBtn->Enabled=true;

N10->Enabled=true;

}

}

//=====================================================

void BlackOut() {

Form1->Caption="Stream `n Sound. (Fedorov A.)";

Form1->N2->Enabled=false;

Form1->StopButtonClick(Form1);

StopButtonSet("false");

Form1->StopButton->Enabled=false;

Form1->PlayButton->Enabled=false;

Form1->PeremotkaGroup->Visible=false;

Form1->Height=340;

Form1->N10->Clear();

Form1->ItemsHolder->Clear();

Form1->ItemsHolder->Items->Add("Список музыкальных объектов:");

Form1->ItemsHolder->Enabled=false;

Form1->EditBtn->Enabled=false;

Form1->DelBtn->Enabled=false;

Form1->N10->Enabled=false;

Form1->Edit1->Text="";

IDPlaying=0;

selected=0;

Main.Wipe();

Send=NULL;

}

//=====================================================

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

setlocale(LC\_ALL,"Russian");

VolumeBar->Position=volume;

if(!BASS\_Init (-1, 44100, BASS\_DEVICE\_3D , 0, NULL))

{MessageBox(NULL,"Инициализия библиотеки BASS не удалось. Дальнейшое продолжение роботы невозможно","BASS провалился",MB\_ICONERROR);

Application->Terminate();

}

else{StatusBar1->Panels->Items[0]->Text="Ждем ввод";}

if(Edit1->Text=="") {

PlayButton->ShowHint=True;

PlayButton->Picture->LoadFromFile("recourses\\img\\false\_play.png");

}

else{

PlayButton->Picture->LoadFromFile("recourses\\img\\on\_play.png");

PlayButton->ShowHint=False;

}

BlackOut();

}

//=====================================================

void \_\_fastcall TForm1::N5Click(TObject \*Sender)

{

OpenDialog1->Filter="Плейлисты (.pls) |\*.pls";

if (OpenDialog1->Execute()) {

BlackOut();

Main.FilePath=OpenDialog1->FileName;

ifstream \*test=new ifstream;

test->open(AnsiString(Main.FilePath).c\_str());

if(Sgetline(test)!="[playlist]"|test->fail()) {

test->close();

MessageBox(NULL, "Файл поврежден","Выбран неверный файл",MB\_ICONWARNING|MB\_TASKMODAL); Form2->Hide();

return;

}

Main.New(ExtractFileName(Main.FilePath),Main.FilePath);

StopButton->Enabled=true;

PlayButton->Enabled=true;

Main.CountRecords();

Main.FillFromFile();

TrayIcon1->BalloonTitle="Stream `n Sound";

TrayIcon1->BalloonHint=String("Плейлист "+ExtractFileName(Main.FilePath)+" подключен. Кол-во записей: "+Main.records);

TrayIcon1->ShowBalloonHint();

}

else {MessageBox(NULL, "Файл не выбран","Выбор файла был прерван",MB\_ICONINFORMATION|MB\_TASKMODAL); Form2->Hide(); }

}

//=====================================================

void \_\_fastcall TForm1::EditBtnClick(TObject \*Sender)

{

String PlusPlus="Редактирование: ";

if(selected>0) {

Mode="edit";

Send=Main.SearchID(selected);

PlusPlus+=Send->Name;

Form2->GroupBox1->Caption=PlusPlus;

Form2->Caption=PlusPlus;

Form2->TitleEdit->Text=Send->Name;

Form2->URLText->Text=Send->URL;

Form2->Show();

}

else {Mode="";}

}

//=====================================================

void \_\_fastcall TForm1::StopButtonClick(TObject \*Sender)

{

if(BASS\_ChannelGetLength(stream,0)-1 <10000000000) {

BASS\_Pause();

Playing=3;

Taskbar1->ProgressState=4;

StatusBar1->Panels->Items[0]->Text=String("("+Main.SearchID(IDPlaying)->Name+") на Паузе");

}

else{

BASS\_StreamFree(stream);

Playing=0;

Taskbar1->ProgressState=0;

StatusBar1->Panels->Items[0]->Text="Остановлено";

}

StopButtonSet("false");

StopButton->Enabled=false;

}

//=====================================================

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

Main.Wipe();

}

//=====================================================

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Main.OutRecords();

}

//=====================================================

void \_\_fastcall TForm1::StopButtonMouseUp(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int X, int Y)

{

if(Playing==1) {

StopButtonSet("on");

}

else if(Playing==0) {StopButtonSet("false");}

}

//=====================================================

void \_\_fastcall TForm1::DelBtnClick(TObject \*Sender)

{

if (selected>0) {

Main.DeleteID(selected);

selected=0;

}

SlowTimerTimer(Form1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::AddBtnClick(TObject \*Sender)

{

int dialog;

String PlusPlus="Добавление нового элемента";

if(Main.PlaylistName.Length()>0) {

Mode="add";

Send=new MusicObject;

Form2->GroupBox1->Caption=PlusPlus;

Form2->Caption=PlusPlus;

Form2->TitleEdit->Text="";

Form2->URLText->Text="";

Form2->Show();

}

else {

dialog = MessageBox(NULL,"Необходимо сперва создать/загрузить плейлист. Создать новый плейлист?", "Процедура добавления", MB\_YESNO|MB\_ICONQUESTION|MB\_TASKMODAL);

if(dialog==6) {

Main.New("NewPlaylist",String());

AddBtnClick(Form1);

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N13Click(TObject \*Sender)

{

EditBtnClick(Form1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N4Click(TObject \*Sender)

{

Main.New("New Playlist",String());

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N9Click(TObject \*Sender)

{

Form1->AddBtnClick(Form1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N11Click(TObject \*Sender)

{

BlackOut();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N7Click(TObject \*Sender)

{

Main.Save();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N6Click(TObject \*Sender)

{

if(!Main.FilePath.IsEmpty()) {

Main.Save(Main.FilePath);

}

else{ Main.Save();}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N15Click(TObject \*Sender)

{

Form3->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormResize(TObject \*Sender)

{

if (Form1->WindowState == wsMinimized) {

Form1->Hide();

Form2->Hide();

TrayIcon1->BalloonTitle="Stream `n Sound (Свернут)";

TrayIcon1->BalloonHint="для открытия нажмите по иконке";

TrayIcon1->ShowBalloonHint();

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TrayIcon1Click(TObject \*Sender)

{

Form1->Show();

Form1->WindowState=wsNormal;

Taskbar1->ProgressState=1;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::TrayIcon1BalloonClick(TObject \*Sender)

{

TrayIcon1Click(Form1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormClose(TObject \*Sender, TCloseAction &Action)

{

TrayIcon1->Visible=false;

BlackOut();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N3Click(TObject \*Sender)

{

Form4->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N10Click(TObject \*Sender)

{

if (selected>0) {

Main.DeleteID(selected);

selected=0;

}

SlowTimerTimer(Form1);}