

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра програмних засобів
(повна назва кафедри, циклової комісії)

КУРСОВИЙ ПРОЕКТ (РОБОТА)

з дисципліни «Об'єктно-орієнтоване програмування»
(назва дисципліни)

на тему: «Розробка редактора плану приміщень»

Студентів 2 курсу
КНТ-127, КНТ-137 груп
напряму підготовки Комп'ютерні науки
спеціальності інженерія
програмного забезпечення
Бережний О. Ю.
(прізвище та ініціали)

Козлов В. В.
(прізвище та ініціали)

Судаков В. Р.
(прізвище та ініціали)

Керівник Доцент к.н.т., Табунщик Г. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

_____ Табунщик Г. В.
(підпис) (прізвище та ініціали)

_____ Миронова Н. О.
(підпис) (прізвище та ініціали)

_____ Каплієнко Т. І.
(підпис) (прізвище та ініціали)

м. Запоріжжя
2018 рік

ЗАВДАННЯ

на курсовий проект студентів

Бережного О. Ю., Козлова В. В., Судакова В. Р.

1. Тема проекту: Розробка редактора плану приміщень
2. Термін здачі студентами закінченого проекту: 28 грудня 2018
3. Вихідні дані до проекту: Реалізувати редактор плану приміщень
дані: інформація про розміри об'єктів, позиції та кути нахилу.
вихідні дані: файл проекту та зображення спроектованої кімнати
4. Зміст розрахунково-пояснювальної записки:
 1. Аналіз предметної області;
 2. Аналіз програмних засобів;
 3. Основні рішення з реалізації компонентів системи;
 4. Керівництво програміста;
 5. Керівництво оператора;

Висновки

Додаток А. Текст програми

Додаток Б. Інтерфейс програми

Додаток В. Слайди презентації
5. Дата видачі завдання: 16 вересня 2018 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів курсового проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	Аналіз індивідуального завдання.	1-2 тиждень	
2.	Аналіз програмних засобів, що будуть використовуватись в роботі.	3-4 тиждень	
3.	Аналіз структур даних, що необхідно використати в курсовій роботі.	4-5 тиждень	
4.	Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача.	5-6 тиждень	
5.	Оформлення відповідних пунктів пояснювальної записки.	6-8 тиждень	Розділи 1,2 ПЗ
6.	Аналіз вимог до апаратних засобів	9 тиждень	
7.	Розробка програмного забезпечення	10-15 тиждень	
8.	Оформлення, відповідних пунктів пояснювальної записки.	9-16 тиждень	Розділи 3-5 ПЗ
9.	Захист курсової роботи.	17 тиждень	

Студент _____

Бережний О. Ю. _____

Студент _____

Козлов В. В. _____

Студент _____

Судаков В. Р. _____

Керівник _____

Табунщик Г. В. _____

«_____» _____ 2018 р.

РЕФЕРАТ

Мета роботи – розробка програмного забезпечення для проектування та редагування плану приміщень або внутрішнього інтер'єру.

Завданням курсового проекту є розробка програми з мети роботи мовою C++ в інтегрованому середовищі розробки Qt Creator.

Проведено аналіз предметної області, досліджено аналогічне програмне забезпечення, а також більшість існуючих методів та програмних засобів для вирішення завдання.

Досліджено особливості мови програмування та середовища розробки.

Здійснено опис прийнятих рішень, реалізованих класів, наведено опис полів та методів реалізованих класів.

Ключові слова: Об'єктно-орієнтоване програмування, програмування, середовище розробки, клас, об'єкт, алгоритм, база даних, планувальник приміщень, програміст, оператор.

ЗМІСТ

Завдання на курсовий проект студентів	2
Календарний план	3
Реферат	4
Зміст	5
Вступ.....	7
1 Аналіз предметної області.....	8
1.1 Огляд існуючих методів вирішення завдання.....	8
1.2 Огляд існуючих планувальників	9
1.2.1 Smart Draw	9
1.2.2 Floor Planner	10
1.2.3 Planner 5D	11
1.3 Постановка завдання.....	12
2 Аналіз програмних засобів.....	13
2.1 Огляд особливостей мови програмування.....	13
2.2 Огляд особливостей обраного компілятора	14
2.3 Огляд можливостей та особливостей обраної системи керування базами даних - SQLite	14
2.4 Огляд особливостей застосованого текстового формату збереження даних - JSON	15
2.5 Огляд класів Qt, що використовуються в роботі	17
2.6 Висновки з розділу	20
3 Основні рішення з реалізації компонентів системи	21
3.1 Основні рішення щодо уявлення даних системи.....	21

	6
3.2 Основні розроблені алгоритми	33
3.3 Особливості реалізації системи	34
3.4 Результати тестування системи	38
3.5 Основні рішення щодо збереження та відтворення інформації.....	39
3.5.1 Запити SQL, що були використані	39
3.5.2 Основні рішення використані при взаємодії з БД	40
3.5.3 Основні рішення щодо збереження даних в JSON файлах.....	41
3.6 Висновки з розділу	41
4 Керівництво програміста.....	43
4.1 Призначення та умови застосування програми	43
4.2 Характеристики програми.....	43
4.3 Звертання до програми	47
4.4 Вхідні та вихідні дані.....	48
4.5 Повідомлення.....	49
5 Інструкція користувача.....	50
5.1 Призначення програми	50
5.2 Умови використання програми.....	50
5.3 Як запустити програму	50
5.4 Виконання програми.....	51
5.5 Повідомлення користувачу	54
Висновки	56
Перелік посилань.....	57
Додаток А Текст програми.....	58
Додаток Б Інтерфейс програми.....	101
Додаток В Слайди презентації.....	103

ВСТУП

Актуальність роботи полягає у дослідженні існуючих програм для проектування плану приміщень, аналізі їх переваг та недоліків, необхідності створення програмного продукту з простим та інтуїтивно зрозумілим інтерфейсом, та можливістю збереження готового проекту для подальшого редагування.

Метою даного курсового проекту є розробка програмного забезпечення, що реалізує можливість проектування та редагування плану і внутрішнього інтер'єру приміщень.

Задачі курсового проекту: аналіз предметної області, розробка відповідних структур даних, проектування графічного інтерфейсу, розробка програми мовою C++ в інтегрованому середовищі розробки Qt Creator та її тестування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі розглянуто існуючі методи вирішення завдання та огляд програм, що реалізують проектування та редагування планів приміщення.

1.1 Огляд існуючих методів вирішення завдання

Програма для проектування та редагування плану приміщень призначена для заощадження часу, фізичних та матеріальних сил людей, які планують будувати приміщення або бажають змінити інтер'єр у ньому. Тобто програма для проектування та редагування плану приміщень для зручної роботи повинна мати інтуїтивно зрозумілий інтерфейс, базу «шаблонних» предметів інтер'єру та підтримку української мовної локалізації.

Проектування — процес створення проекту, прототипу, прообразу майбутнього об'єкта.

Редагування — приведення об'єкта редагування у відповідність із чинними у певний час нормами, а також його творча оптимізація, метою чого є отримання заданого соціального ефекту.

План – креслення, що відбиває на площині в умовних знаках і певному масштабі предмет, споруду та ін..

Приміщення – будівля, квартира чи окрема кімната, а якій поміщається хто-, що-небудь, яка використовується або буде використовуватись для чогось.

Інтер'єр – архітектурно й художньо оздоблена внутрішня частина будинку, приміщення, що забезпечує комфорт людині та визначає функціональне призначення приміщення.

Інтуїтивний зрозумілий інтерфейс — сукупність засобів і правил, що забезпечують взаємодію користувача із програмним забезпеченням, яке

наслідуює розповсюджені і звичні для багатьох принципи побудови взаємодії. При роботі з таким інтерфейсом користувач використовує звичні для нього дії особливо не задумуючись над тим що та як зробити, його дії ніби інтуїтивні, звідки й назва.

База «шаблонних» предметів інтер'єру – це набір найуживаніших елементів інтер'єру: двері, ліжка, шафа, лампа, телевізор та т. ін..

Мовна локалізація — культурна адаптація продукту до особливостей певної країни, регіону чи групи населення.

1.2 Огляд існуючих планувальників

1.2.1 Smart Draw

Офіційний сайт програми: www.smartdraw.com.

Переваги даного програмного продукту:

- наявність 7 днів безкоштовного використання продукту;
- можливість вибору форми приміщення (квадратна, прямокутна)

перед створенням проекту;

- наявність готових шаблонів приміщень;
- схожість інтерфейсу до MS Office;
- функція застосування текстур до фігур;
- підтримка «гарячих клавіш»;
- демонстрація розмірів об'єкту під час проектування;

Недоліками даного програмного продукту:

- часова обмеженість безкоштовного використання;
- відсутність функції завдання розмірів площини перед створенням проекту;
- погана читабельність розмірів на фоні;
- відсутність української локалізації.

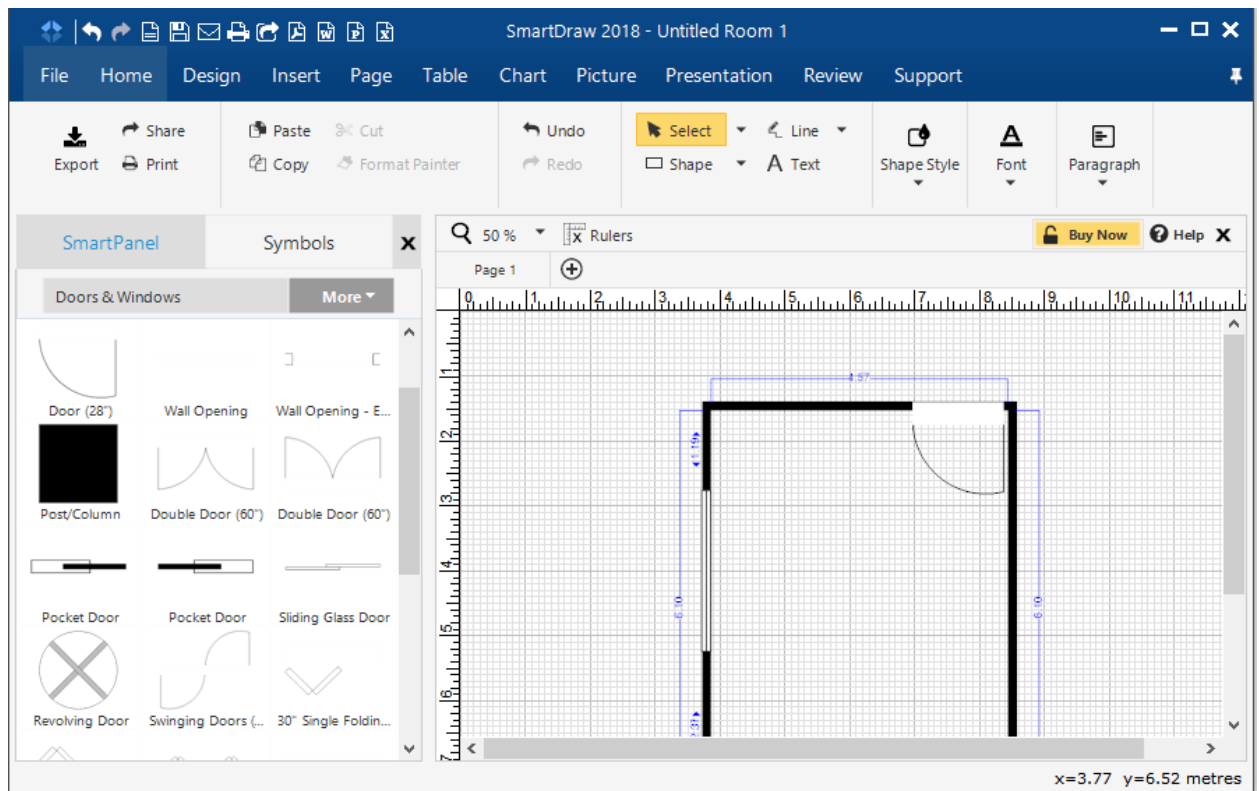


Рисунок 1.2.1 – Інтерфейс програми Smart Draw

1.2.2 Floor Planner

Офіційний сайт програми: www.floorplanner.com.

Переваги даного програмного продукту:

- наявність функції швидкого перегляду поточного плану;
- легкодоступність елементарних елементів (стіни, вікна, двері тощо);
- наявність меню операцій у верхній частині екрану;
- безкоштовність;
- докладна довідка користувача.

Недоліками даного програмного продукту:

- відсутність десктопної версії продукту;
- відсутність розмірів об'єктів;
- не підтримуються «гарячі клавіші»;
- відсутність української локалізації.

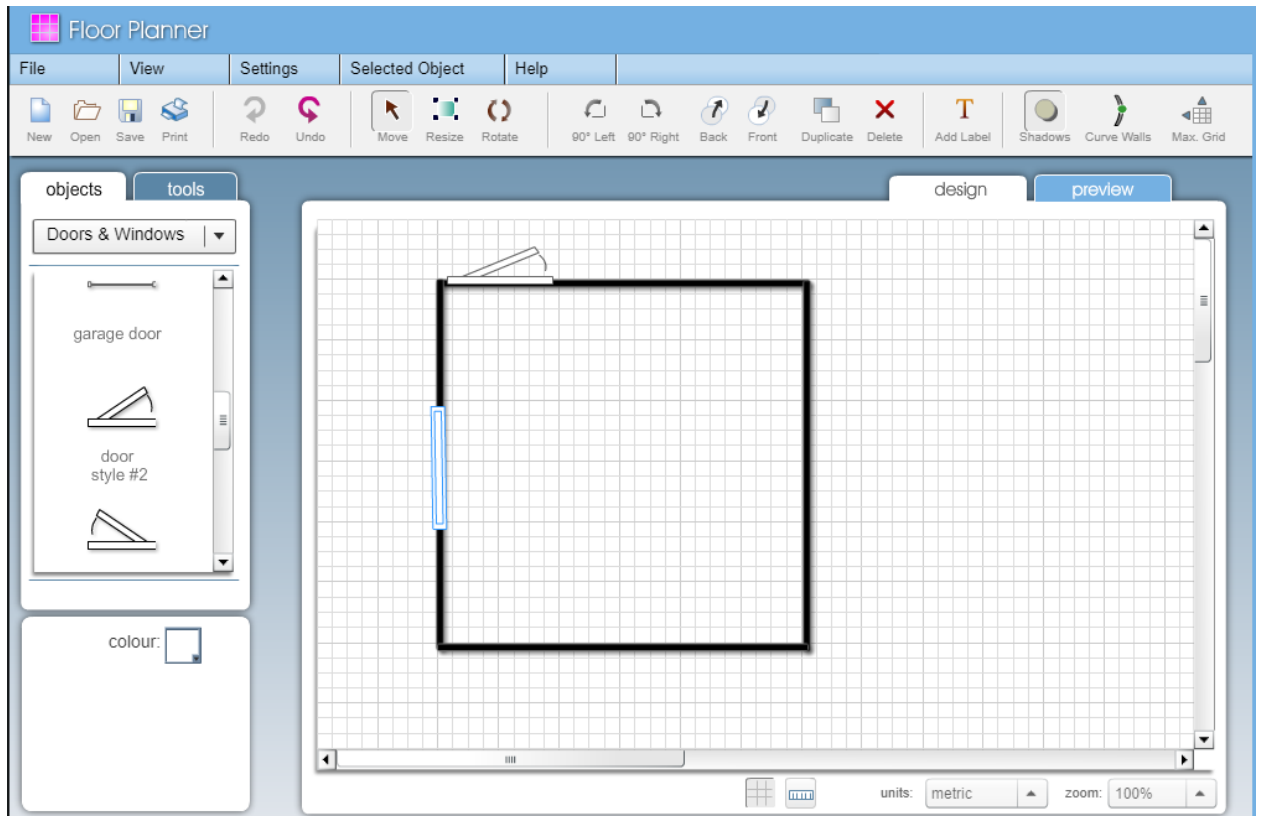


Рисунок 1.2.2 – Інтерфейс програми Floor Planner

1.2.3 Planner 5D

Офіційний сайт програми: www.planner5d.com.

Переваги даного програмного продукту:

- велика база предметів інтер'єру;
- підтримка декількох мов;
- безкоштовність;
- кросплатформність;

Недоліками даного програмного продукту:

- орієнтованість у більшому ступені на 3-х вимірний простір;
- не підтримуються «гарячі клавіші»;
- відсутність української локалізації.

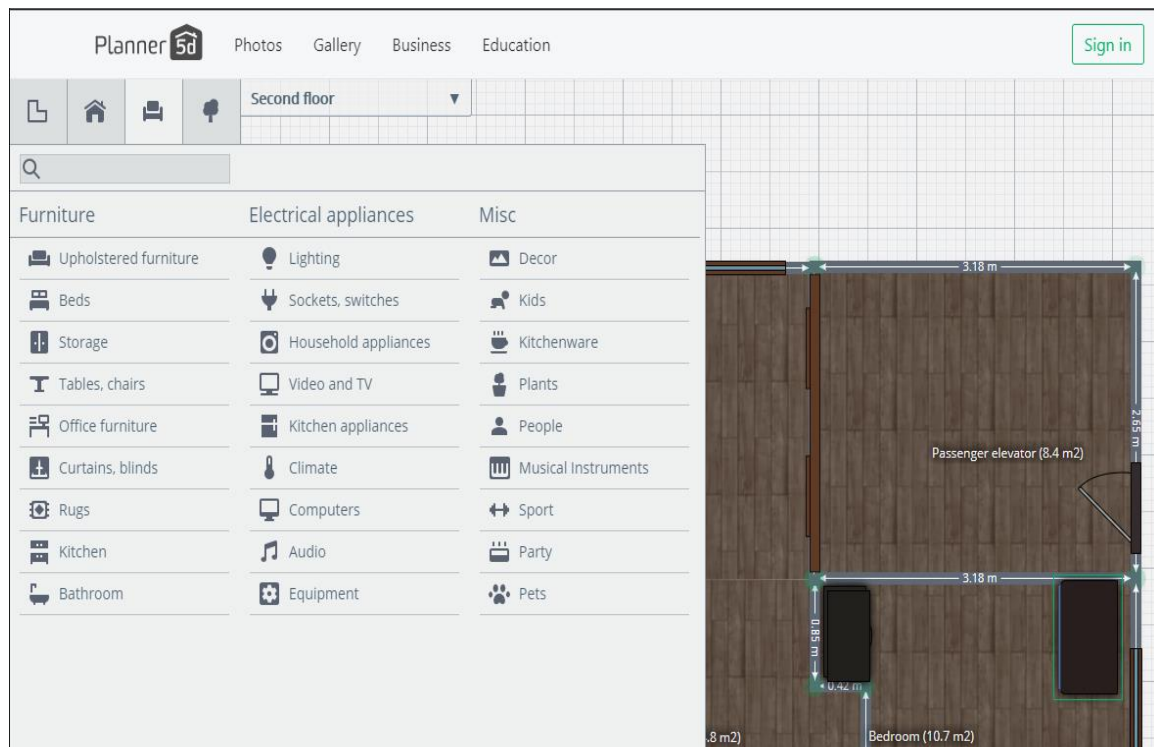


Рисунок 1.2.3 – Інтерфейс програми Planner 5D

1.3 Постановка завдання

Задача поставлена перед програмою – надати користувачу інструменти для швидкого створення та редагування плану приміщень.

Функції, що будуть реалізовані системою:

- проектування форми приміщення (стін);
- додавання елементів інтер'єру;
- можливість зберігати проект та експортувати результат роботи у зображення;
- можливість відкривати попередньо збережені проекти;
- можливість змінювати масштаб робочої області;
- можливість змінювати мову інтерфейсу в програмі (англійська, російська, українська);
- робота з програмою за допомогою «гарячих клавіш»;
- «гумовий» макет (розмір вмісту змінюється разом з розмірами вікон, щоб вони були завжди «заповнені»).

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ

В даному розділі розглянуті основні особливості програмних засобів, за допомогою яких реалізовано курсовий проект.

2.1 Огляд особливостей мови програмування

C++ – це мова програмування високого рівня з підтримкою декількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної. Розроблена Б'ярном Страуструпом 1979 року [1].

Мова C++ має синтаксис, заснований на синтаксисі C.

Основними відмінностями C++ від C є:

- підтримка об'єктно-орієнтованого програмування;
- підтримка узагальненого програмування (шаблони);
- оновлена та розширена стандартна бібліотека;
- додаткові типи даних;
- обробка винятків;
- вбудовані функції;
- перевантаження операторів;
- перевантаження імен функцій;
- посилення і оператори управління вільно розподіленою пам'яттю [2].

Швидкість роботи програм на C++ практично не поступається програмам на C, при цьому C++ реалізовано більше можливостей і засобів. На мові C++ розробляють програми для найрізноманітніших платформ і систем. Окрім цього існує можливість роботи на низькому рівні з пам'яттю, адресами, портами, можливість створення узагальнених алгоритмів для різних типів даних, їхня спеціалізація, і обчислення на етапі компіляції, з використанням шаблонів.

2.2 Огляд особливостей обраного компілятора

Qt Creator — кросплатформна вільна IDE для розробки на C, C++ і QML [3]. Розроблено для роботи з фреймворком Qt. Включає в себе графічний інтерфейс відладчика і візуальні засоби розробки інтерфейсу. Підтримувані компілятори: GCC, Clang, MinGW, MSVC, Linux ICC, GCCE, RVCT, WINSCW. Інтегроване середовище розробки, призначене для створення кросплатформних застосунків з використанням бібліотеки Qt. Відповідно підтримується розробка програм мовою C++.

Основне завдання Qt Creator — спростити розробку програми за допомогою фреймворку Qt на різних платформах.

В Qt Creator реалізовано автодоповнення, в тому числі ключових слів, введених в стандарті C++ 11 (починаючи з версії 2.5), підсвічування коду. Також, починаючи з версії 2.4, є можливість завдання стилю вирівнювання, відступів і постановки дужок [4]. Це все спрощує сприйняття коду.

IDE підтримує системи контролю версії, такі як Git, Subversion, Mercurial, CVS, Bazaar, Perforce.

2.3 Огляд можливостей та особливостей обраної системи керування базами даних - SQLite

SQLite — полегшена реляційна система керування базами даних.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

Бібліотека SQLite написана мовою C. Проте розроблено механізм підключення й роботи з БД через цю бібліотеку використовуючи інші мови програмування, зокрема мову C++.

Основними особливостями SQLite є:

- транзакції атомарні, послідовні, ізольовані, і міцні навіть після збоїв системи і збоїв живлення;
- не потребує ані установки, ані адміністрування;
- база даних зберігається в одному крос-платформовому файлі на диску;
- підтримка BLOBів;
- малий розмір коду: менше ніж 350KB повністю налаштований, і менш з опущеними додатковими функціями;
- швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій
- відносно простий, легкий у використанні API
- крос-платформовість;
- кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується.

2.4 Огляд особливостей застосованого текстового формату збереження даних - JSON

JSON — це текстовий формат збереження даних та обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Розробив і популяризував формат Дуглас Крокфорд.

JSON має переваги XML - дозволяє складні структури в атрибутах і займає менше місця.

JSON будується на двох структурах:

– набір пар назва/значення. У різних мовах це реалізовано як об'єкт, запис, структура, словник, хеш-таблиця, список з ключем або асоціативним масивом;

– впорядкований список значень. У багатьох мовах це реалізовано як масив, вектор, список, або послідовність.

У JSON використовуються такі їхні форми:

Об'єкт — це послідовність пар назва/значення. Об'єкт починається з символу { і закінчується символом } . Кожне значення слідує за : і пари назва/значення відділяються комами.

Масив — це послідовність значень. Масив починається символом [і закінчується символом]. Значення відділяються комами.

Значення може бути рядком в подвійних лапках, або числом, або логічними true чи false, або null, або об'єктом, або масивом. Ці структури можуть бути вкладені одна в одну.

Рядок — це послідовність з нуля або більше символів юнікода, обмежена подвійними лапками, з використанням escape-послідовностей, що починаються зі зворотної косої риски \. Символи представляються простим рядком.

2.5 Огляд класів Qt, що використовуються в роботі

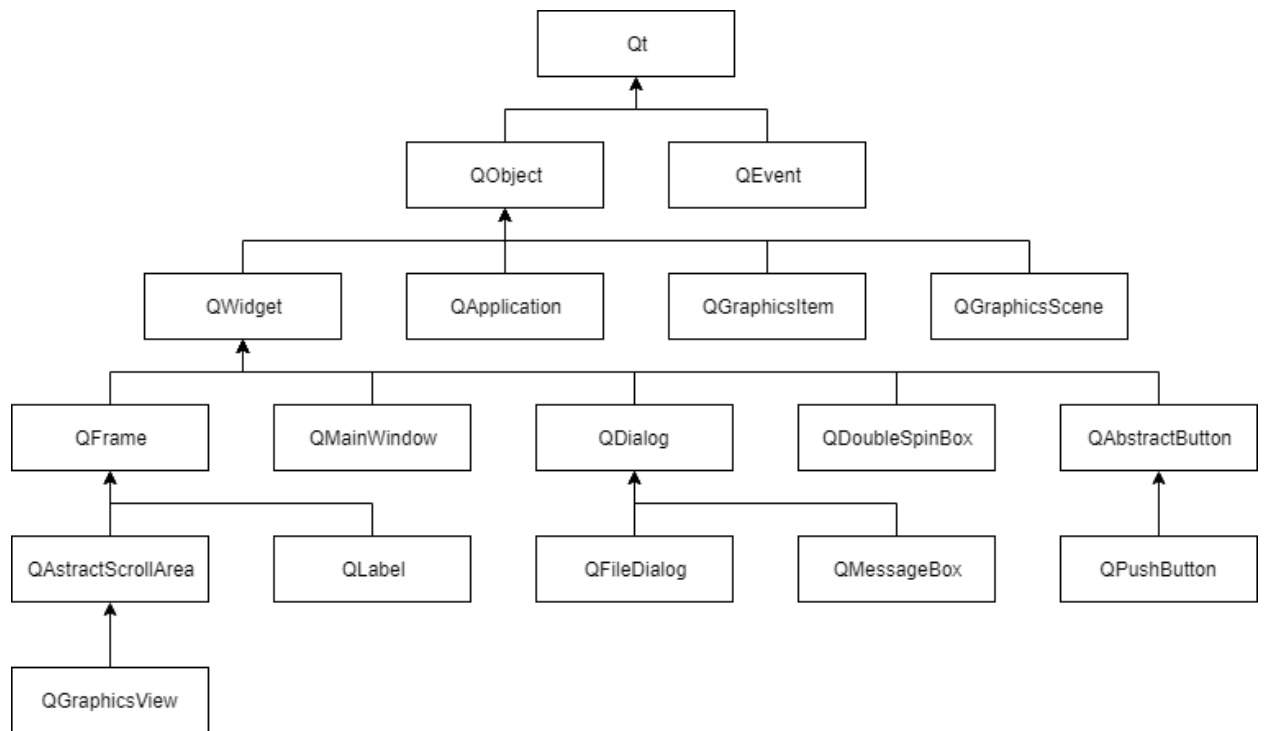


Рисунок 2.1 – Ієрархія основних класів Qt

Qt має власну ієрархію класів. Ієрархія основних класів наведена на рис. 2.1, до яких належать такі класи [5]: Qt, QObject, QEvent, QWidget, QApplication, QGraphicsItem, QGraphicsScene, QFrame, QMainWindow, QDialog, QFileDialog, QMessageBox, QDoubleSpinBox, QAbstractButton, QPushButton, QAbstractScrollArea, QGraphicsView.

Клас QApplication керує логікою GUI. Він містить основний цикл подій, в якому всі події від віконної системи та інших джерел обробляються і координуються. Він також обробляє ініціалізацію і завершення програми та забезпечує управління сесіями. Крім того, QApplication обробляє більшість загальносистемних і загальних програмних налаштувань.

Клас QObject є базовим для всіх об'єктів Qt і центром об'єктної моделі Qt. Головна особливість в цій моделі – це дуже потужний механізм для зв'язку об'єктів, званий сигналами і слотами. Ви можете з'єднати сигнал зі слотом і роз'єднати. Не має ні конструктора копіювання, ні оператора присвоювання.

Клас QMainWindow надає головне вікно програми. Має власний компоновальник, в який ви можете додавати QToolBar и, QDockWidget, QMenuBar, і QStatusBar. Компоновальник має центральну область, яка може бути зайнята будь-яким віджетом.

Клас QFile надає інтерфейс для читання і запису файлів, і являє собою пристрій вводу/виводу для читання і запису тексту і бінарних файлів і ресурсів. QFile можуть бути використані окремо або, що більш зручно, з QTextStream або QDataStream.

Клас QMessageBox надає цілу серію статичних методів, за допомогою яких можна створювати вікна повідомлень. Ці методи надають підтримку для трьох рівнів: інформаційного, застережливого, критичного. Тип вікна вибирається залежно від обставин. Вікна можуть містити до трьох кнопок.

Клас QWidget є базовим для всіх об'єктів для користувача інтерфейсу. QWidget має безліч функцій-членів, але деякі з них мають досить обмежену функціональність: наприклад, QWidget має властивість керуючу шрифтом, але ніколи не використовує його безпосередньо. Є безліч його підкласів, які забезпечують реальну функціональність, такі як QPushButton, QListWidget, QTabWidget і т.д.

Віджет QPushButton являє собою командну кнопку. Кнопка, або командна кнопка, є найбільш часто використовуваним віджетом в будь-якому графічному інтерфейсі. Натискання кнопки вказує комп'ютеру, що необхідно виконати дію або відповісти на запитання.

Клас QFileDialog забезпечує простий та зручний діалог, для обрання файлів або каталогу користувачем. Вхідним значенням може бути рядок, число або елемент зі списку.

Клас QEvent є базовим класом для всіх класів повідомлень. Об'єкти повідомлень містять параметри повідомлень. Головний цикл обробки повідомлень Qt отримує нативні повідомлення віконної системи з черги повідомлень, конвертує їх в QEvents, і пересилає конвертовані повідомлення в QObject.

Клас `QGraphicsItem` є базовим класом для всіх графічних елементів в `QGraphicsScene`. Він надає зручну основу для написання своїх власних елементів. Вона включає визначення геометрії елемента, виявлення зіткнень, реалізацію його відтворення і взаємодія елементів за допомогою покажчиків. `QGraphicsItem` є частиною каркаса графічного представлення.

Клас `QGraphicsScene` надає поверхню для управління великою кількістю графічних 2D елементів. Цей клас є контейнером для `QGraphicsItem`. Він використовується разом з `QGraphicsView` для демонстрації зображень, таких як лінії, прямокутники, текст або навіть власні елементи на двовірній поверхні. `QGraphicsScene` також надає функціональність, яка дозволить вам ефективно визначати положення елементів і які елементи видимі всередині довільної області сцени.

Клас `QGraphicsView` надає віджет для відображення вмісту `QGraphicsScene`. `QGraphicsView` відображає вміст `QGraphicsScene` у прокручуваній області. `QGraphicsView` є частиною каркаса графічного представлення.

Клас `QDoubleSpinBox` надає віджет спін-боксу, що приймає дійсні числа. `QDoubleSpinBox` дозволяє користувачеві вибирати значення, натискаючи кнопки вгору та вниз або натискаючи клавіші "Вгору" та "Вниз" на клавіатурі, щоб збільшити або зменшити поточне значення. Користувач може також ввести значення вручну.

Клас `QSettings` надає постійні платформонезалежні налаштування програми. Зазвичай користувачі очікують, що додаток буде запам'ятовувати свої настройки (розміри і позиції вікон, параметри і т.д.) між сесіями. Ця інформація часто зберігається в системному реєстрі в Windows і в файлах налаштувань XML в Mac OS X. В Unix-системах, у відсутності стандарту, більшість додатків (включаючи програми KDE) використовують текстові INI-файли. `QSettings` - це абстракція навколо цих технологій, що дозволяє зберегти і відновити налаштування програми портативним способом. Він також підтримує користувацькі формати зберігання.

Клас `QTranslator` забезпечує підтримку інтернаціоналізації для виведення тексту. Об'єкт цього класу містить набір перекладів з вихідної мови на цільову мову. `QTranslator` надає функції для пошуку перекладів у файлі перекладу. Файли перекладу створюються за допомогою `Qt Linguist`. Найбільш поширеним способом використання `QTranslator` є: завантажити файл перекладу, встановити його за допомогою `QApplication::installTranslator()` і використати через `QObject::tr()`.

2.6 Висновки з розділу

У даному розділі було описано особливості мови програмування, середі розробки, обраного компілятора та системи керування базами даних. Окрім цього у ньому описані класи бібліотеки `Qt`, що були використані у ході розробки програми: `Qt`, `QObject`, `QEvent`, `QWidget`, `QApplication`, `QGraphicsItem`, `QGraphicsScene`, `QFrame`, `QMainWindow`, `QDialog`, `QFileDialog`, `QMessageBox`, `QDoubleSpinBox`, `QAbstractButton`, `QPushButton`, `QAbstractScrollArea`, `QGraphicsView`.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В даному розділі розглянуті основні рішення розроблені класи, основні розроблені алгоритми, рішення щодо розробки інтерфейсу користувача, рішення щодо збереження даних та використання бази даних.

3.1 Основні рішення щодо уявлення даних системи

В ході роботи над програмою курсового проекту було розроблено класи:

- AboutWindow – форма з інформацією про програмний продукт
- DataBase – клас для роботи з SQLite
- ItemEditWindow – форма редагування властивостей об'єкта
- FileOpen – клас з реалізацією відкриття файлів
- FileSave – клас з реалізацією збереження файлів
- FileWay – базовий клас для FileOpen та FileSave
- FormCreate – форма створення нового проекту
- GraphicItem – клас графічного об'єкта
- ItemByColor – клас графічного об'єкта представленого кольором
- ItemByPicture – клас графічного об'єкта представленого зображенням
- JsonData – клас для роботи з файлами *.json
- MainWindow – головна форма програми
- MdiChild – форма з графічною сценою проекту
- Points – клас з визначеними точками графічних об'єктів

Класову структуру програми зображено на рисунку 3.1.

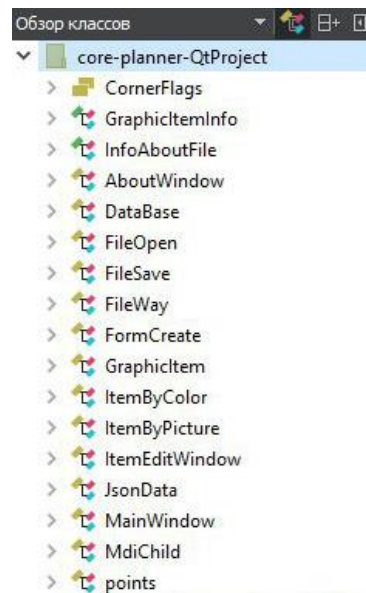


Рисунок 3.1 – Класова структура програми

Клас AboutWindow містить у собі опис програмного забезпечення та контакти авторів. Дані та методи класу наведені в табл. 3.1.

Таблиця 3.1 – Дані та методи класу AboutWindow

Поля та методи класу	Опис
1	2
<i>private:</i>	
Ui::AboutWindow *ui;	Покажчик на графічний інтерфейс
QPixmap backgroundImage;	Зображення фону
void setupOtherUi();	Метод генерації графічного інтерфейсу
<i>private slots:</i>	
void on_pushButtonClose_clicked();	Обробка закриття
void on_labelSystemNick_linkActivated (const QString &link);	Обробка натискання на посилання
void on_labelx8357238_linkActivated (const QString &link);	Обробка натискання на посилання
void on_labelvaleriyzp_linkActivated (const QString &link);	Обробка натискання на посилання
<i>public:</i>	
AboutWindow(QWidget *parent = 0);	Конструктор
~AboutWindow();	Деструктор

Клас DataBase містить у собі засоби для роботи з базою даних через систему керування базами даних SQLite. Дані та методи класу наведені в табл. 3.2.

Таблиця 3.2 – Дані та методи класу DataBase

Поля та методи класу	Опис
1	2
<i>private:</i>	
QSqlDatabase dataBase;	
int maxItems;	Ліміт записів
int itemsLeft;	Кількість виконаних запитів
int itemsLeftDataBase();	Кількість виконаних запитів
bool createTable();	Створення таблиці БД
bool openDataBase();	Відкриття БД
bool restoreDataBase();	Відновлення БД
bool removeRecord (const QString &fWay);	Видалення запису
<i>public:</i>	
DataBase();	Конструктор
~DataBase();	Деструктор
void connectToDataBase();	З'єднання з БД
bool insertIntoTable (const QString &fName, const QString &fData, const QString &fWay, const double &fSquare);	Вставка в таблицю
void closeDataBase();	Закриття БД
void WriteTableToArray (InfoAboutFile *Array, int MaximumCount, int *CurrentCount);	Запис таблиці у масив

Клас ItemEditWindow містить у собі реалізацію елементів керування для зміни властивостей об'єктів сцени проекту. Дані та методи класу наведені в табл. 3.3.

Таблиця 3.3 – Дані та методи класу ItemEditWindow

Поля та методи класу	Опис
1	2
<i>private:</i>	
Ui::ItemEditWindow *ui;	Показчик на графічний інтерфейс
<i>private slots:</i>	
void on_pushButtonApply_clicked();	Обробка натиску на кнопку
<i>public:</i>	
ItemEditWindow();	Конструктор

Продовження таблиці 3.3

ItemEditWindow (MainWindow *Main, MdiChild *Parent, QWidget *parent = 0);	Конструктор
~ItemEditWindow();	Деструктор
MdiChild *FormWithScene;	Показчик на дочірню форму
MainWindow *MainWin;	Показчик на базову форму

Клас FileOpen містить у собі реалізацію функцій для відкриття проекту. Дані та методи класу наведені в табл. 3.4.

Таблиця 3.4 – Дані та методи класу FileOpen

Поля та методи класу	Опис
1	2
<i>private:</i>	
MainWindow *Main;	Показчик на базову форму
<i>public:</i>	
FileOpen(MainWindow *Parent);	Конструктор
~FileOpen();	Деструктор
void SetWay();	Встановлення шляху до файлу
void OpenFile();	Відкриття файлу
void OpenFileByWay (QString Way, QString FileName);	Відкриття файлу за шляхом

Клас FileSave містить у собі реалізацію функцій для збереження проекту. Дані та методи класу наведені в табл. 3.5.

Таблиця 3.5 – Дані та методи класу FileSave

Поля та методи класу	Опис
1	2
<i>public:</i>	
FileSave();	Конструктор
~FileSave();	Деструктор
MdiChild *ParenWithScene;	Показчик на дочірню форму
QString HistoryFileWay;	Шлях до файлу з адресами проектів
void SetWay();	Встановлення шляху

Продовження таблиці 3.5

void Save(MainWindow *Main);	Зберігання файлу
void SaveAs(MainWindow *Main);	Зберігання файлу за новими форматами
void SaveToHistory();	Зберігання до файлу з адресами проектів
void InitHistory();	Ініціалізація файлу з адресами проектів
void LoadFromHistory(int Version);	Завантаження інформації з файлу з адресами
void DeleteHighVersions(int Version);	Видалення старих записів

Клас FileWay містить у собі опис адреси та назви файлу. Дані та методи класу наведені в табл. 3.6.

Таблиця 3.6 – Дані та методи класу FileWay

Поля та методи класу	Опис
1	2
public:	
FileWay();	Конструктор
FileWay(QString WayToFile);	Конструктор
~FileWay();	Деструктор
QString Way;	Шлях до файлу
QString FileName;	Назва файлу
QString GetFileNameFromWay (QString AdressFile);	Отримання назви файлу з його шляху
virtual void SetWay() = 0;	Встановлення шляху

Клас FormCreate містить у собі реалізацію інтерфейсу для створення нових проектів чи навігації по нещодавнім. Дані та методи класу наведені в табл. 3.7.

Таблиця 3.7 – Дані та методи класу FormCreate

Поля та методи класу	Опис
1	2
private:	
InfoAboutFile *Files;	Інформація про файл
int CountOfFiles;	Кількість файлів
Ui::FormCreate *ui;	Показник на графічний інтерфейс
int WayToOpen;	

Продовження таблиці 3.7

<i>private slots:</i>	
void SetFilesButtons();	Встановлення тексту кнопок
void SetButtonsTextAlign();	Встановлення вирівнювання тексту
void on_pushButtonCreateNew_clicked();	Обробка натискання на кнопку
void on_pushButtonProjectName_clicked();	Обробка натискання на кнопку
void on_pushButtonProjectTime_clicked();	Обробка натискання на кнопку
void on_pushButtonProjectSquare_clicked();	Обробка натискання на кнопку
void on_pushButton_1_clicked();	Обробка натискання на кнопку
void on_pushButton_2_clicked();	Обробка натискання на кнопку
void on_pushButton_3_clicked();	Обробка натискання на кнопку
void on_pushButton_4_clicked();	Обробка натискання на кнопку
void on_pushButton_5_clicked();	Обробка натискання на кнопку
void on_pushButtonOpen_clicked();	Обробка натискання на кнопку
<i>public:</i>	
FormCreate();	Конструктор
~FormCreate();	Деструктор
MainWindow *MainWin;	Показчик на базову форму

Клас GraphicItem містить у собі реалізацію поведінки графічного об'єкта на сцені проекту. Дані та методи класу наведені в табл. 3.8.

Таблиця 3.8 – Дані та методи класу GraphicItem

Поля та методи класу	Опис
1	2
<i>public:</i>	
GraphicItem();	Конструктор
GraphicItem (double setWidth, double setHeight, MdiChild *MdiChildParent, QObject *parent = 0);	Конструктор
~GraphicItem();	Деструктор
points MidlePoints;	Координати середніх точок
double Width;	Ширина
double Height;	Висота
int FlagCorner;	Флаг куту
bool IsEditNow;	Флаг можливості редагування
bool IsResizeModNow;	Флаг зміни розміру
int SizeOfPoint;	Розмір точки
double MinimumWidth;	Мінімальна ширина
double MinimumHeight;	Мінімальна висота

Продовження таблиці 3.8

double CurrentAngleOfRotation;	Поточний кут нахилу
QPointF ClickedPointOnScene;	Координата натиснення на сцену
MdiChild *ParenWithScene;	Показчик на дочірню форму
bool IsSceneEdited;	Флаг редагування сцени
virtual QString GetType() = 0;	Отримання типу об'єкту
double GetHeight();	Отримання висоти об'єкту
double GetWidth();	Отримання ширини об'єкту
double GetRotationAngle();	Отримання куту нахилу об'єкту
void SetRotationAngle (double RotationAngleToSet);	Встановлення куту нахилу
void SetHeight(double HeightToSet);	Встановлення висоти
void SetWidth(double WidthToSet);	Встановлення ширини
void SetUnActive();	Встановлення неактивності
virtual GraphicItemInfo GetItemInfo() = 0;	Отримання інформації
QPointF m_shiftMouseCoords;	Координати миші
QRectF boundingRect() const;	Прямокутник
void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) = 0;	Метод малювання
void mouseMoveEvent (QGraphicsSceneMouseEvent *event);	Обробка руху миші
void mousePressEvent (QGraphicsSceneMouseEvent *event);	Обробка натискання миші
void mouseReleaseEvent (QGraphicsSceneMouseEvent *event);	Обробка відпускання клавіші миші
void mouseDoubleClickEvent (QGraphicsSceneMouseEvent *event);	Обробка подвійного кліку миші
void hoverMoveEvent (QGraphicsSceneHoverEvent *event);	Обробка наведення миші
void ResetCursor (double CursorX, double CursorY, bool MouseMove);	Повернути курсор

Клас ItemByColor містить у собі засоби відображення об'єкту представленого кольоровою палітрою. Дані та методи класу наведені в табл. 3.9.

Таблиця 3.9 – Дані та методи класу ItemByColor

Поля та методи класу	Опис
1	2
<i>private:</i>	
QColor ColorOfItem;	Колір
void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);	Метод малювання
<i>public:</i>	
ItemByColor();	Конструктор
ItemByColor(double setWidth, double setHeight, QColor setColor, MdiChild *MdiChildParent, QObject *parent = 0);	Конструктор
~ItemByColor();	Деструктор
QString GetType();	Отримання типу
GraphicItemInfo GetItemInfo();	Отримання інформації графіки

Клас ItemByPicture містить у собі засоби відображення об'єкту представленого зображенням. Дані та методи класу наведені в табл. 3.10.

Таблиця 3.10 – Дані та методи класу ItemByPicture

Поля та методи класу	Опис
1	2
<i>private:</i>	
QString NameOfPicture;	Назва зображення
QImage PictureOfItem;	Зображення
void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);	Метод малювання
<i>public:</i>	
ItemByPicture();	Конструктор
ItemByPicture(double setWidth, double setHeight, QString PictureName, MdiChild *MdiChildParent, QObject *parent = 0);	Конструктор
~ItemByPicture();	Деструктор
QString GetType();	Отримання типу
GraphicItemInfo GetItemInfo();	Отримання інформації

Клас `JsonData` містить у собі засоби для роботи з файлом у форматі `*.json`, а саме: зчитування, видалення, змінення. Дані та методи класу наведені в табл. 3.11.

Таблиця 3.11 – Дані та методи класу `JsonData`

Поля та методи класу	Опис
1	2
<i>private:</i>	
<code>QJsonDocument doc;</code>	Зміст з файлу
<code>QJsonParseError docError;</code>	Повідомлення про помилку
<i>public:</i>	
<code>JsonData();</code>	Конструктор
<code>~JsonData();</code>	Деструктор
<code>void inFromFile (const QString&fWay, MdiChild *Parent);</code>	Запис до сцени з файлу об'єктів
<code>void outToFile (const QString& fWay, MdiChild *Parent);</code>	Запис проекту у файл
<code>void outHistoryToFile (const QString& fWay, MdiChild *Parent);</code>	Запис історії у файл
<code>void outInitHistoryFile (const QString& fWay, MdiChild *Parent);</code>	Ініціалізація файлу історії
<code>void InitializeSceneFromFile (MdiChild *Parent,QJsonArray &docArr);</code>	Ініціалізація сцени з файлу
<code>void InitScenFromFile (const QString&fWay, MdiChild *Parent, int Version);</code>	Завантаження сцени з файлу
<code>void DeleteHighVersions (const QString&fWay, int Version);</code>	Видалення версій

Клас `MainWindow` містить у собі реалізацію відображення панелі інструментів для планування приміщень та області відкритих проектів. Дані та методи класу наведені в табл. 3.12.

Таблиця 3.12 – Дані та методи класу `MainWindow`

Поля та методи класу	Опис
1	2
<i>private:</i>	
<code>Ui::MainWindow *ui;</code>	Показчик на елементи інтерфейсу
<code>int openProjects = 0;</code>	Кількість відкритих проектів
<code>QStandardItemModel *modelBuildTools;</code>	Модель списку інструментів будування

Продовження таблиці 3.12

QSortFilterProxyModel *proxyModelBuildTools;	Сортування
QStandardItemModel *modelFurnitureTools;	Модель списку інструментів меблі
QSortFilterProxyModel *proxyModelFurnitureTools;	Сортування
QStandardItemModel *modelDecorTools;	Модель списку інструментів декору
QSortFilterProxyModel *proxyModelDecorTools;	Сортування
int FilesCount = 0;	Кількість файлів
void setupOtherUi();	Встановлення інших об'єктів
void setupBuildTools();	Встановлення інструментів
void setupFurnitureTools();	Встановлення фурнітури
void setupDecorTools();	Встановлення декору
QTranslator* translator;	Об'єкт перекладача
void changeTranslator (QString postfix);	Метод, що підключає перекладач з новим словником
void changeEvent(QEvent *event);	Метод для обробки подій
<i>private slots:</i>	
void on_actionNew_triggered();	Обробка створення нового проекту
void on_actionOpen_triggered();	Обробка відкриття проекту
void on_actionSave_triggered();	Обробка збереження проекту
void on_actionSaveAs_triggered();	Обробка збереження проекту за варіантами
void on_actionClose_triggered();	Обробка закриття проекту
void on_actionAbout_triggered();	Обробка виклику форми про програму
void on_listViewBuildTools_clicked (const QModelIndex &index);	Обробка активації списку інструментів будування
void on_listViewFurnitureTools_clicked (const QModelIndex &index);	Обробка активації списку інструментів меблі
void on_listViewDecorTools_clicked (const QModelIndex &index);	Обробка активації списку інструментів декору
void on_actionUk_triggered();	Обробка обрання української локалізації
void on_actionEn_triggered();	Обробка обрання англійської локалізації
void on_actionRu_triggered();	Обробка обрання російської локалізації
void on_actionUndo_triggered();	Обробка відміни

Продовження таблиці 3.12

void on_actionRedo_triggered();	Обробка повтору
void on_actionDelete_triggered();	Обробка видалення
void on_actionClear_triggered();	Обробка очистки
void on_actionZoomIn_triggered();	Обробка приближення
void on_actionZoomOut_triggered();	Обробка віддалення
void on_actionSetItemSize_triggered();	Обробка встановлення розміру об'єкту
public:	
explicit MainWindow (QWidget *parent = 0);	Конструктор
~MainWindow();	Деструктор
int countOpen();	Кількість відкритих
MdiChild *activeMdiChild() const;	Показчик на дочірню форму
DataBase database;	База даних SQLite
int uniqueProjectID = 1;	Унікальний ідентифікатор
void loadProject(QWidget *widget);	Завантаження проекту

Клас MdiChild містить у собі реалізацію інтерфейсу з графічною сценою відкритого проекту. Дані та методи класу наведені в табл. 3.13.

Таблиця 3.13 – Дані та методи класу MdiChild

Поля та методи класу	Опис
1	2
private:	
Ui::MdiChild *ui;	Показчик на графічний інтерфейс
int Width, Height;	Ширина та висота
QVector<QGraphicsLineItem*> HorizontalLines;	Масив горизонтальних ліній
QVector<QGraphicsLineItem*> VerticalLines;	Масив вертикальних ліній
void DrawHorizontalLines();	Малювання горизонтальних ліній
void DrawVerticalLines();	Малювання вертикальних ліній
public:	
explicit MdiChild(QWidget *parent);	Конструктор
~MdiChild();	Деструктор
GraphicItem *LastActive = 0;	Показчик на останній активний об'єкт
QGraphicsScene *scene;	Показчик на графічну сцену
void addItem(QString Name, double setWidth, double setHeight);	Метод додавання об'єкту за ім'ям

Продовження таблиці 3.13

void addItem(QColor ItemColor, double setWidth, double setHeight);	Метод додавання об'єкту за кольором
void UpdateScene();	Оновлення сцени
void SaveAsImage (QString NameOfFile);	Метод збереження за варіантами
void setScaleToView (double X, double Y);	Метод встановлення масштабу
void DrawGrid();	Метод малювання сітки
void DeleteGrid();	Метод видалення сітки
void SetSceneParametrs (int setWidth, int setHeight);	Метод встановлення параметрів сцени
FileSave SaveFileInfo;	Збереження інформації в файл
int VersionOfScene;	Версія сцени
int MaximumVersionOfScene;	Ліміт сцен
bool AfterUndo;	Флаг здійснення відміни

Клас Points містить у собі значення і функції задавання точок з об'єкту. Дані та методи класу наведені в табл. 3.14.

Таблиця 3.14 – Дані та методи класу Points

Поля та методи класу	Опис
1	2
<i>private:</i>	
QPointF TopMinimumCenter;	Верхня точка центру об'єкту за мінімальної висоти
QPointF LeftMinimumCenter;	Ліва точка центру об'єкту за мінімальної висоти
QPointF RightMinimumCenter;	Права точка центру об'єкту за мінімальної висоти
QPointF BottomMinimumCenter;	Нижня точка центру об'єкту за мінімальної висоти
<i>public:</i>	
Points();	Конструктор
~Points();	Деструктор
void SetTopMinimumCenter (QPointF point);	Встановлення верхньої точки центру об'єкту за мінімальної висоти
void SetLeftMinimumCenter (QPointF point);	Встановлення лівої точки центру об'єкту за мінімальної висоти
void SetRightMinimumCenter (QPointF point);	Встановлення правої точки центру об'єкту за мінімальної висоти

Продовження таблиці 3.14

<code>void SetBottomMinimumCenter (QPointF point);</code>	Встановлення нижньої точки центру об'єкту за мінімальної висоти
<code>QPointF GetTopMinimumCenter();</code>	Отримання верхньої точки центру об'єкту за мінімальної висоти
<code>QPointF GetLeftMinimumCenter();</code>	Отримання лівої точки центру об'єкту за мінімальної висоти
<code>QPointF GetRightMinimumCenter();</code>	Отримання правої точки центру об'єкту за мінімальної висоти
<code>QPointF GetBottomMinimumCenter();</code>	Отримання нижньої точки центру об'єкту за мінімальної висоти

3.2 Основні розроблені алгоритми

Під час розробки було реалізовано наступні алгоритми:

- файлове зчитування та запис у файл;
- відображення елементів візуальної форми програми;
- Сортування даних методом Шелла;
- Сортування даних бульбашковим методом;
- Сортування даних методом вставок;
- Розрахунок кута за трьома точками;
- Визначення назви файлу з повного шляху;
- Розрахунок зміни розмірів і позицій об'єкту;
- Розрахунок зміни зовнішнього виду курсору.

Створення нових та редагування вже збережених проектів дозволяє комфортно працювати над розробкою плану приміщення з можливістю зберігти та відкласти працю на потім.

3.3 Особливості реалізації системи

Особливістю реалізації системи є дуже зрозумілий дружній до користувача інтерфейс. Будь-яка людина зможе просто користуватися розробленим додатком, вибрати потрібні дані та створювати свої замітки.

На головній формі MainWindow розміщені наступні елементи, які зображені на рис. 3.2.

Object	Class
▼ MainWindow	QMainWindow
▼ centralWidget	QWidget
mdiArea	QMdiArea
▼ tabWidget	QTabWidget
▼ tabBuildTools	QWidget
listViewBuildTools	QListView
▼ tabFurnitureTools	QWidget
listViewFurnitureTools	QListView
▼ tabDecorTools	QWidget
listViewDecorTools	QListView
> menuBar	QMenuBar
statusBar	QStatusBar

Рисунок 3.2 – Елементи головної форми MainWindow

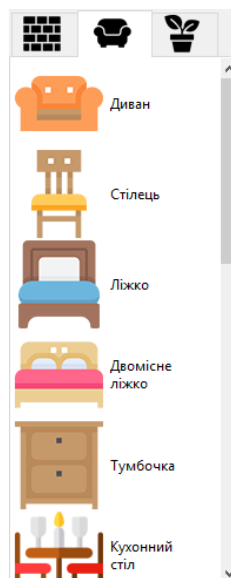


Рисунок 3.2 – Список інструментів

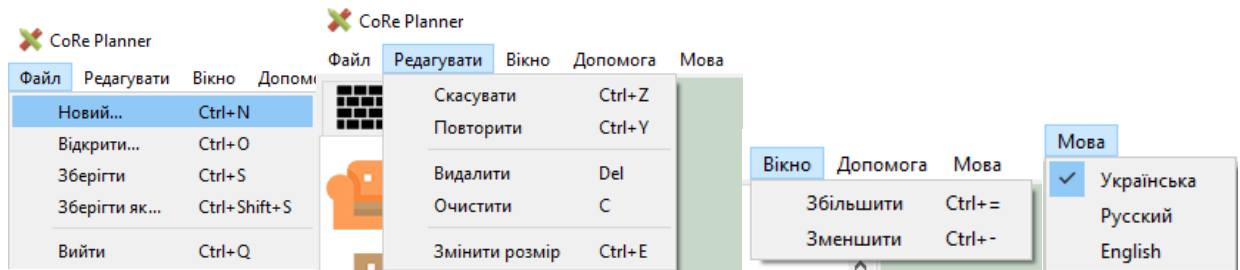


Рисунок 3.3 – Пункти головного меню

Область для дочірніх форм типу `QMdiArea` призначена для відображення розшифрованого змісту файлів проекту у окремих вікнах.

Клас `QListView` відображає списком інструменти планувальника у вигляді: піктограма, назва розміщуваного об'єкта. Він призначений для розміщення нових об'єктів на графічній сцені проекту.

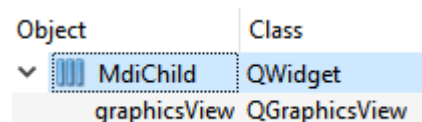
Клас `QTabWidget` зберігає згруповані за типом списки інструментів планувальника.

Клас `QMenuBar` зберігає усі необхідні об'єкти класу `QAction`, які призначені для:

- виклику форм створення нових або відкриття для редагування попередньо розроблених та збережених проектів;
- відкату змін та інших маніпуляцій з активним проектом;
- виклику довідки та інформації про програму;
- зміни мови інтерфейсу програми.

На формі `MdiChild` розміщені наступні елементи, які зображені на рис.

3.4.

Рисунок 3.4 – Елементи форми `MdiChild`

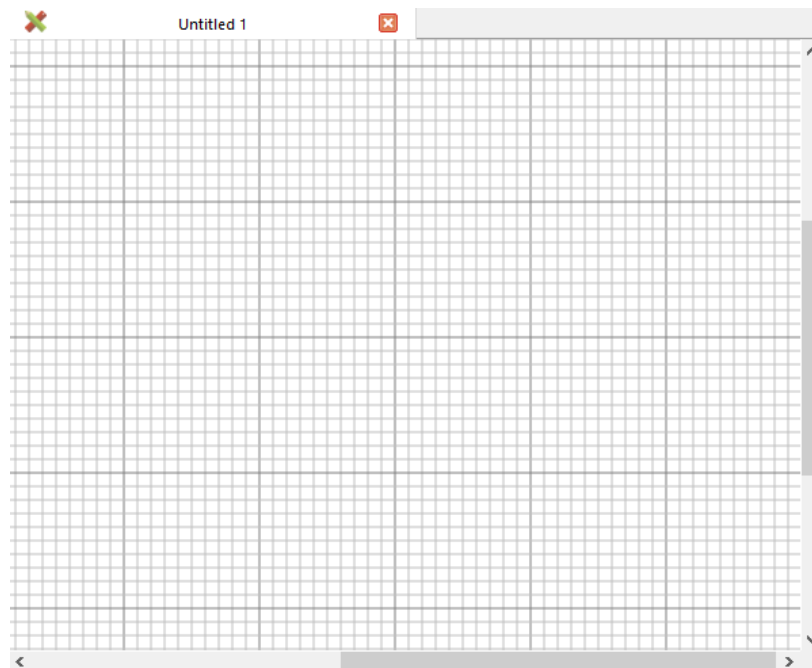


Рисунок 3.5 – Форма MdiChild

Об'єкт `graphicsView` класу `QGraphicsView` призначений для відображення сцени проекту та взаємодії з графічними об'єктами розміщеними на об'єкті класу `QGraphicsScene`.

На формі `ItemEditWindow` розміщені наступні елементи, які зображені на рис. 3.6.

Object	Class
ItemEditWindow	QWidget
groupBoxPosition	QGroupBox
doubleSpinBoxAngle	QDoubleSpinBox
doubleSpinBoxPosX	QDoubleSpinBox
doubleSpinBoxPosY	QDoubleSpinBox
label_3	QLabel
label_4	QLabel
label_5	QLabel
groupBoxSize	QGroupBox
doubleSpinBoxHeight	QDoubleSpinBox
doubleSpinBoxWidth	QDoubleSpinBox
labelHeight	QLabel
label_2	QLabel
pushButtonApply	QPushButton

Рисунок 3.6 – Елементи форми ItemEditWindow

The screenshot shows a dialog box titled 'Розміри' (Dimensions) and 'Розміщення' (Positioning). It contains five input fields with spinners:

- Висота (Height): 0,91 m.
- Ширина (Width): 2,00 m.
- Позиція X (Position X): 6,21 m.
- Позиція Y (Position Y): 4,75 m.
- Кут нахилу (Tilt angle): 0,00°

At the bottom, there is a button labeled 'Застосувати' (Apply).

Рисунок 3.7 – Елементи форми ItemEditWindow

Клас `QGroupBox` використаний для логічного групування елементів керування.

На формі `FormCreate` розміщені наступні елементи, які зображені на рис. 3.8.

Object	Class
FormCreate	QWidget
groupBoxLayoutNew	QGroupBox
doubleSpinBoxHeight	QDoubleSpinBox
doubleSpinBoxWidth	QDoubleSpinBox
labelHeight	QLabel
labelWidth	QLabel
labelCreateNew	QLabel
labelRecent	QLabel
pushButtonCreateNew	QPushButton
pushButtonOpen	QPushButton
pushButtonProjectName	QPushButton
pushButtonProjectSquare	QPushButton
pushButtonProjectTime	QPushButton
pushButton_1	QPushButton
pushButton_2	QPushButton
pushButton_3	QPushButton
pushButton_4	QPushButton
pushButton_5	QPushButton

Рисунок 3.8 – Елементи форми FormCreate

The screenshot shows a section of the `FormCreate` form titled 'Створити новий' (Create new). It contains a sub-section titled 'Розмір' (Size) with two input fields:

- Висота (Height): 10,00 m.
- Ширина (Width): 10,00 m.

Рисунок 3.9 – Частина форми FormCreate

Клас QPushButton використаний для визначення рішення щодо створення нового проекту чи відкриття нещодавнього та для відображення ім'я, дати відкриття і площі обраного нещодавнього проекту.

Клас QLabel створює підписи для допомоги орієнтування користувача.

Клас QDoubleSpinBox призначений для введення ширини та висоти нового проекту.

На формі AboutWindow розміщені наступні елементи, які зображені на рис. 3.10.

Object	Class
▼ AboutWindow	QDialog
labelBG	QLabel
labelDescription	QLabel
labelSystemNick	QLabel
labelvaleriyzp	QLabel
labelx8357238	QLabel
pushButtonClose	QPushButton

Рисунок 3.10 – Елементи форми AboutWindow

Клас QLabel слугує для відображення опису програмного продукту та для надання клікабельних посилань на аккаунти Telegram авторів.

Клас QPushButton використаний для закривання форми.

3.4 Результати тестування системи

У ході тестування програми усі помилки було виправлено. На даному етапі програма функціонує правильно та злагоджено.

На рис. 3.11 зображено повідомлення, яке виникає при неправильному відкритті файлу.

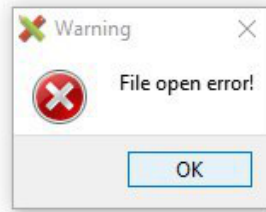


Рисунок 3.11 – Помилка відкриття файлу

3.5 Основні рішення щодо збереження та відтворення інформації.

Для роботи з даними були використана бібліотека QSql. База даних застосована для збереження історії збережених файлів.

SQLite було обрано через низку переваг, а саме: база даних зберігається в одному крос-платформовому файлі на диску, наявна підтримка BLOBів, швидший за популярні рушії клієнт-серверних баз даних для найпоширеніших операцій.

Наявна можливість збереження проекту в файл формату:

- JSON з можливістю подальшого відтворення у програмі та редагування;
- JPG, PNG, BMP.

3.5.1 Запити SQL, що були використані

У під час роботи проекту застосовані такі запити:

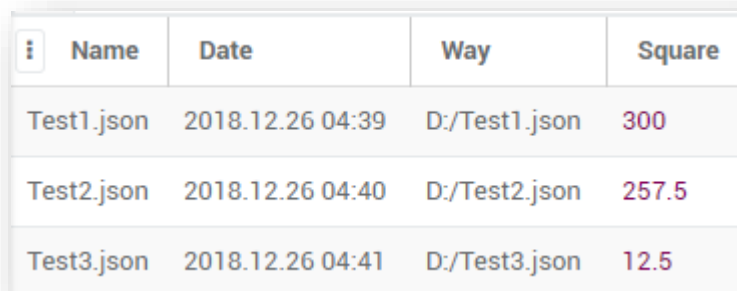
- “ CREATE TABLE ім'я_таблиці (перелік полів та їх типу) ” – запит на створення таблиці в БД;
- “ SELECT * FROM ім'я_таблиці ” - запит на отримання значень усіх полів з таблиці;
- “ INSERT INTO ім'я_таблиці VALUES (значення _1, значення _2, ..., значення _N) ” - запит на вставку вказаних значень у поля таблиці;

- “ DELETE FROM ім'я_таблиці WHERE значення_поля = параметр ”
- запит на видалення поля таблиці, у якого визначене значення певного поля дорівнює параметру.

3.5.2 Основні рішення використані при взаємодії з БД

Клас DataBase створен для поліпшення роботи з базою даних . У цьому класі реалізовано метод створення, завантаження бази даних, отримання усіх значень з таблиці, видалення певного значення за параметром.

З'єднання з базою даних забезпечує метод DataBase::createTable(), де наявна перевірка чи була створена у файлі таблиця, чи ні: якщо так – файл відкривається (DataBase::openDataBase()), якщо ні – ініціалізується (DataBase::createTable()). Структуру таблиці зазначено на рисунку 3.15.



Name	Date	Way	Square
Test1.json	2018.12.26 04:39	D:/Test1.json	300
Test2.json	2018.12.26 04:40	D:/Test2.json	257.5
Test3.json	2018.12.26 04:41	D:/Test3.json	12.5

Рисунок 3.15 – структура таблиці «FileHistory»

Таблиця «FileHistory» зберігає дані про збережені проекти. Вона має наступні поля:

- Name – ім'я збереженого проекту;
- Date – поле, яке містить дату зберігання проекту;
- Way – поле, яке містить шлях до збереженого проекту.
- Square – поле, яке зберігає площу приміщення.

БД використовується при відкритті вікна FormCreate(створення нового файлу).

При збереженні будь-якого проекту у форматі JSON викликається функція `DataBase::insertIntoTable(const QString&fName,const QString&fData,const QString&fWay,const double&fSquare)`, яка записує до БД новий член.

3.5.3 Основні рішення щодо збереження даних в JSON файлах

Клас `JsonData` створен для поліпшення роботи з базою даних . У цьому класі реалізовано метод створення, завантаження бази даних, отримання усіх значень з таблиці, видалення певного значення за параметром.

Для взаємодії з файлами формату JSON використовувалися такі методи: `JsonData::inFromFile()` та `JsonData::outFromFile()` – для збереження проекту та відкриття проекту; `JsonData::outHistoryToFile()` та `JsonData::InitScenFromFile()` – для збереження змін в файл історії змін, та для завантаження з файлу історії змін.

3.6 Висновки з розділу

В ході роботи над програмою курсового проекту було розроблено класи: `AboutWindow`, `DataBase`, `ItemEditWindow`, `FileOpen`, `FileSave`, `FileWay`, `FormCreate`, `GraphicItem`, `ItemByColor`, `ItemByPicture`, `JsonData`, `MainWindow`, `MdiChild`, `Points`.

При виконанні курсової роботи був використаний механізм сигналів та слотів. Таким чином, було розроблено декілька алгоритмів (файлове зчитування та запис у файл, відображення елементів візуальної форми програми) які відповідають тим чи іншим функціональним частинам програми, що допомагає створити максимально зручний та зрозумілий

користувачеві інтерфейс, а також організувати досить швидку реалізацію програми.

Було реалізовано взаємодію з базою даних, та з форматом JSON.

При розробці інтерфейсу програми були використані такі візуальні компоненти Qt: QPushButton, QDoubleSpinBox, QLineEdit, QTextEdit, QLabel, QGraphicsScene, QMdiArea, QListView, QTabWidget, QMenuBar та QStatusBar.

У ході тестування системи усі помилки було виправлено.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

В даному розділі розглянуті призначення, умови застосування, характеристика програми, звертання до програми, початкові та вихідні дані та представлені повідомлення.

4.1 Призначення та умови застосування програми

Призначення програми – проектування плану приміщень будівлі.

Умовою застосування для коректної роботи є використання останніх версій ПЗ, що необхідно для роботи програми, наявність дисплею, маніпулятора миші, клавіатури.

4.2 Характеристики програми

Програма виконана за допомогою мови програмування високого рівня C++ в середовищі розробки Qt Creator 5.3. Проект (рис. 4.1) містить класи, їх реалізацію, файли ресурсів та проекту, файли формату *.sql, де зберігаються дані та файли формату *.ts, де зберігаються тексти перекладів.

Система розподілена на класи з даними і функціями, які використовуються цими даними та виконують свої задачі для роботи системи. Загалом система поділяється на наступні модулі: «Програмна область», «Робоча область», «Редагування об'єкту», «Параметризація проекту», «Зворотній зв'язок».

Основою всього проекту є .pro файл. Основні особливості .pro файлу:

– QT += core gui sql – зазначення основних модулів Qt, що використовуються програмою;

– HEADERS += *.h - усі .h файли проекту. Ці файли визначають основні використані бібліотеки та класи програми і створені для роздроблення програми на модулі та окремі файли. До них належить 20 файлів програми;

– SOURCES += *.cpp - усі .cpp файли проекту. Ці файли називають вихідними файлами. До них належить 21 .cpp файл;

– FORMS += *.ui - усі форми програми. Форми відповідають за візуальний інтерфейс, з яким взаємодіє користувач. До них належать 5 форм;

– RESOURCES += *.qrc - файли ресурсів, використані програмою. В випадку розробленої програми до ресурсів належить іконки та .png зображення, що розбиті на різні папки для зручнішого доступу та кращого інтуїтивного сприйняття;

– QMAKE_CXXFLAGS += -std=c++0x – визначення стандарту мови, на якому розроблено проект;

– TRANSLATIONS += *.ts – файли перекладу, використані програмою. Програма підтримує відображення інтерфейсу на 3 мовах.

Проект включає в себе такі файли:

– aboutwindow.h - файл, що містить оголошення класу AboutWindow, який призначений для відображення на екрані інформаційного вікна з даними про програму та її авторів;

– database.h - файл, що містить оголошення класу DataBase, який призначений для роботи з SQLite, а саме: для збереження, відтворення та зміни історії файлів проекту ;

– edititemwindow.h - файл, що містить оголошення класу EditItemWindow, який призначений для відображення на екрані вікна редагування даних обраного об'єкту;

– fileopen.h - файл, що містить оголошення класу FileOpen, який призначений для відкриття вже існуючого проекту у програмі;

– filesave.h - файл, що містить оголошення класу FileSave, який призначений для збереження проекту у вигляді редагуємого файлу або зображення;

– `fileway.h` - файл, що містить оголошення класу `FileWay`, який призначений для зберігання та редагування даних про знаходження файлу у системі;

– `formcreate.h` - файл, що містить оголошення класу `FormCreate`, який призначений для відображення на екрані вікна з даними для створення нового проекту або відкриття нещодавнього;

– `graphicitem.h` - файл, що містить оголошення класу `GraphicItem`, який призначений для обробки дії користувача при роботі з об'єктами;

– `itembycolor.h` - файл, що містить оголошення класу `ItemByColor`, який призначений для роботи з прямокутним об'єктом даного кольору;

– `itembypicture.h` - файл, що містить оголошення класу `ItemByPicture`, який призначений для роботи з прямокутним об'єктом, який містить зображення;

– `jsondata.h` - файл, що містить оголошення класу `JsonData`, який призначений для роботи з JSON форматом, а саме: для збереження та відтворення проектів;

– `mainwindow.h` - файл, що містить оголошення класу `MainWindow`, який призначений для відображення на екрані головного вікна програми;

– `mdichild.h` - файл, що містить оголошення класу `MdiChild`, який призначений для відображення на екрані проекту;

– `points.h` - файл, що містить оголошення класу `Points`, який призначений для зберігання та редагування даних про мінімальні позиції центрів об'єкту, що редагується;

– `aboutwindow.cpp` - файл для реалізації класу `AboutWindow`;

– `database.cpp` - файл для реалізації класу `DataBase`;

– `itemeditwindow.cpp` - файл для реалізації класу `ItemEditWindow`;

– `fileopen.cpp` - файл для реалізації класу `FileOpen`;

– `filesave.cpp` - файл для реалізації класу `FileSave`;

– `fileway.cpp` - файл для реалізації класу `FileWay`;

- formcreate.cpp - файл для реалізації класу FormCreate;
- graphicitem.cpp - файл для реалізації класу GraphicItem;
- itembycolor.cpp - файл для реалізації класу ItemByColor;
- itembypicture.cpp - файл для реалізації класу ItemByPicture;
- jsondata.cpp - файл для реалізації класу JsonData;
- mainwindow.cpp - файл для реалізації класу MainWindow;
- mdichild.cpp - файл для реалізації класу MdiChild;
- points.cpp - файл для реалізації класу Points;
- aboutwindow.ui - файл xml формату, що містить графічний опис класу AboutWindow;
- formcreate.ui - файл xml формату, що містить графічний опис класу FormCreate;
- itemeditwindow.ui - файл xml формату, що містить графічний опис класу ItemEditWindow;
- mainwindow.ui - файл xml формату, що містить графічний опис класу MainWindow;
- mdichild.ui - файл xml формату, що містить графічний опис класу MdiChild;
- core-cursors.qrc: ResizeD1.png, ResizeD2.png, ResizeH.png, ResizeV.png, Rotation.png – набір різновидів курсору;
- core-images.qrc: bath.png, bed-double.png, bed-single.png, bedside.png, car.png, chair.png, door.png, door-garage.png, oven.png, plant.png, sofa.png, table-dining.png, table-glass.png, table-rectangle.png, table-wooden.png, table-work.png, toilet.png, window.png – набір іконок елементів меню редагувальника; cube.ico, icon.ico – набір іконок програми; about.png, splash.png – набір фону форм програми;
- scene-images.qrc: bath.png, bed-double.png, bed-single.png, bedside.png, car.png, chair.png, door.png, door-garage.png, oven.png, plant.png, sofa.png, table-dining.png, table-glass.png, table-rectangle.png, table-wooden.png, table-work.png,

toilet.png, window.png – набір зображень елементів для відображення у проекті редагувальника;

– Core_Planner_en.ts - файл xml формату, що містить переклад інтерфейсу на англійську мову;

– Core_Planner_ru.ts - файл xml формату, що містить переклад інтерфейсу на російську мову;

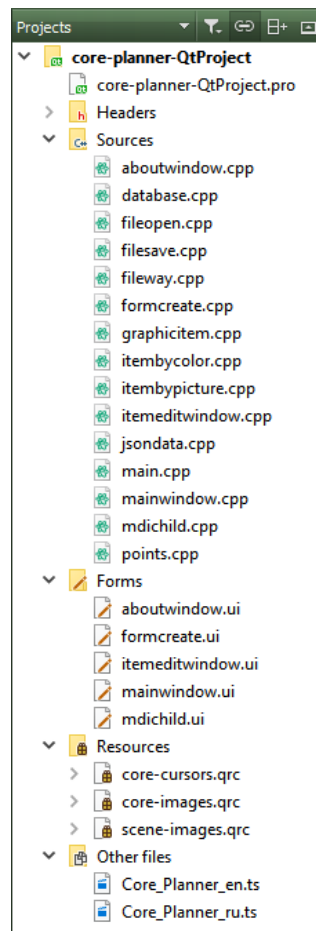


Рисунок 4.1 – Структура проекту

4.3 Звертання до програми

Для звернення, по-перше, необхідно розархівувати теку з програмою та розташувати її у каталог, до якого має доступ оператор. По-друге, для запуску програми у папці core-planner-QtProject (рис.4.2) треба запустити файл core-planner-QtProject.pro, дочекатися запуску проекту у QtCreator. По-третє,

натиснути кнопку «Збірка» і програма готова до роботи. Програма стабільно працює при використанні Desktop Qt 5.3 MinGW 32bit Kit, приклад налаштування збірки наведений на рис. 4.3.

de > core-planner-QtProject

Name	Date modified	Type	Size
debug	12/25/2018 10:25 ...	File folder	
img	12/25/2018 6:20 AM	File folder	
release	12/25/2018 6:21 AM	File folder	
aboutwindow.cpp	12/25/2018 6:20 AM	CPP File	1 KB
aboutwindow.h	12/25/2018 6:20 AM	H File	1 KB
aboutwindow.ui	12/25/2018 6:20 AM	Qt UI file	6 KB
Core_Planner_en.qm	12/25/2018 6:20 AM	QM File	6 KB
Core_Planner_en.ts	12/25/2018 6:20 AM	TS File	14 KB
Core_Planner_ru.qm	12/25/2018 6:20 AM	QM File	7 KB
Core_Planner_ru.ts	12/25/2018 6:20 AM	TS File	15 KB
core-cursors.qrc	12/25/2018 6:20 AM	QRC File	1 KB
core-images.qrc	12/25/2018 6:20 AM	QRC File	3 KB
core-planner-QtProject.pro	12/25/2018 6:20 AM	Qt Project file	2 KB
core-planner-QtProject.pro.user	12/25/2018 10:32 ...	Per-User Project O...	64 KB

Рисунок 4.2 – Каталог з файлами програми

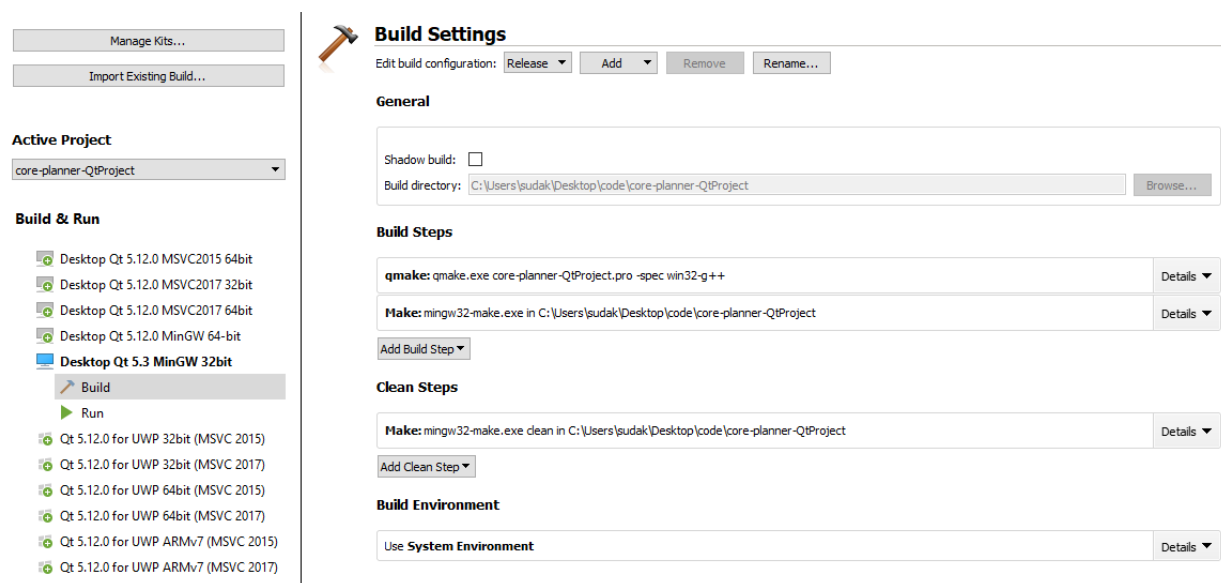


Рисунок 4.3 – Налаштування збірки проекту

4.4 Вхідні та вихідні дані

Вхідними даними виступають різні набори даних: розміри доступної площі для проектування; дані об'єктів у проекті (тип, найменування, розмір, положення на сцені, кут нахилу). Вихідними даними виступають *.json файли,

що зберігають впорядковану інформацію, або графічні (*.png, *.jpg, *.bmp) вже експортовані файли.

Також в програмі реалізована обробка виняткових ситуацій для окремих класів, що оброблюють помилки відкриття вхідних та вихідних файлів.

Для зберігання проектів використовуються файли формату *.json.

4.5 Повідомлення

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі оператор отримає інформаційні повідомлення щодо деталей помилки.

При роботі можуть бути отримані такі повідомлення:

– помилка відкриття файлу, така помилка може виникнути у випадку, якщо файл з інформацією знаходиться в невірній директорії або його пошкоджено (рис. 4.3). В такому разі необхідно перевірити правильність прописаного в коді шляху до текстового файлу, в якому зберігаються дані.

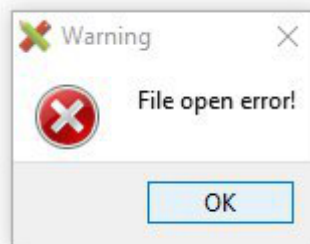


Рисунок 4.3 – Помилка відкриття файлу

У системі реалізовано функції винятків, тому усі непередбачені ситуації оброблюються та програма продовжує стабільно працювати.

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

В даному розділі розглянуто призначення програми, умови її виконання, процес виконання програми та повідомлення для користувача.

5.1 Призначення програми

Програма призначена для роботи з різними наборами інформації, їх оновлення, зміни та зберігання в окремих *.json файлах.

5.2 Умови використання програми

Програма має коректно працювати за умов:

- запуску останньої версії програми;
- наявність дисплею;
- наявність маніпулятора миші;
- наявність клавіатури;
- збереження наборів інформації у директорії, в яких є права на читання та запис файлів, що не є заблокованими.

До користувачів висуваються наступні вимоги:

- навички роботи з мишею, клавіатурою;
- навички роботи з графічним інтерфейсом операційної системи;
- володіння інформацією про призначення основних елементів керування більшої програм.

5.3 Як запустити програму

Для роботи у програмі необхідно розархівувати теку з програмою та розташувати її у каталог, до якого ви маєте доступ. Для запуску програми у папці core-planner-QtProject (рис.4.2) треба запустити файл core-planner-

QtProject.pro, дочекатися запуску проекту у QtCreator і натиснути кнопку «Збірка». Програма готова до роботи.

de > core-planner-QtProject				
Name	Date modified	Type	Size	
debug	12/25/2018 10:25 ...	File folder		
img	12/25/2018 6:20 AM	File folder		
release	12/25/2018 6:21 AM	File folder		
aboutwindow.cpp	12/25/2018 6:20 AM	CPP File	1 KB	
aboutwindow.h	12/25/2018 6:20 AM	H File	1 KB	
aboutwindow.ui	12/25/2018 6:20 AM	Qt UI file	6 KB	
Core_Planner_en.qm	12/25/2018 6:20 AM	QM File	6 KB	
Core_Planner_en.ts	12/25/2018 6:20 AM	TS File	14 KB	
Core_Planner_ru.qm	12/25/2018 6:20 AM	QM File	7 KB	
Core_Planner_ru.ts	12/25/2018 6:20 AM	TS File	15 KB	
core-cursors.qrc	12/25/2018 6:20 AM	QRC File	1 KB	
core-images.qrc	12/25/2018 6:20 AM	QRC File	3 KB	
core-planner-QtProject.pro	12/25/2018 6:20 AM	Qt Project file	2 KB	
core-planner-QtProject.pro.user	12/25/2018 10:32 ...	Per-User Project O...	64 KB	

Рисунок 5.1 – Каталог з файлами програми

5.4 Виконання програми

Після того, як було запущено програму, перед користувачем постає головна форма з вільною Mdi областю. Головна форма зображена на рис. 5.2.

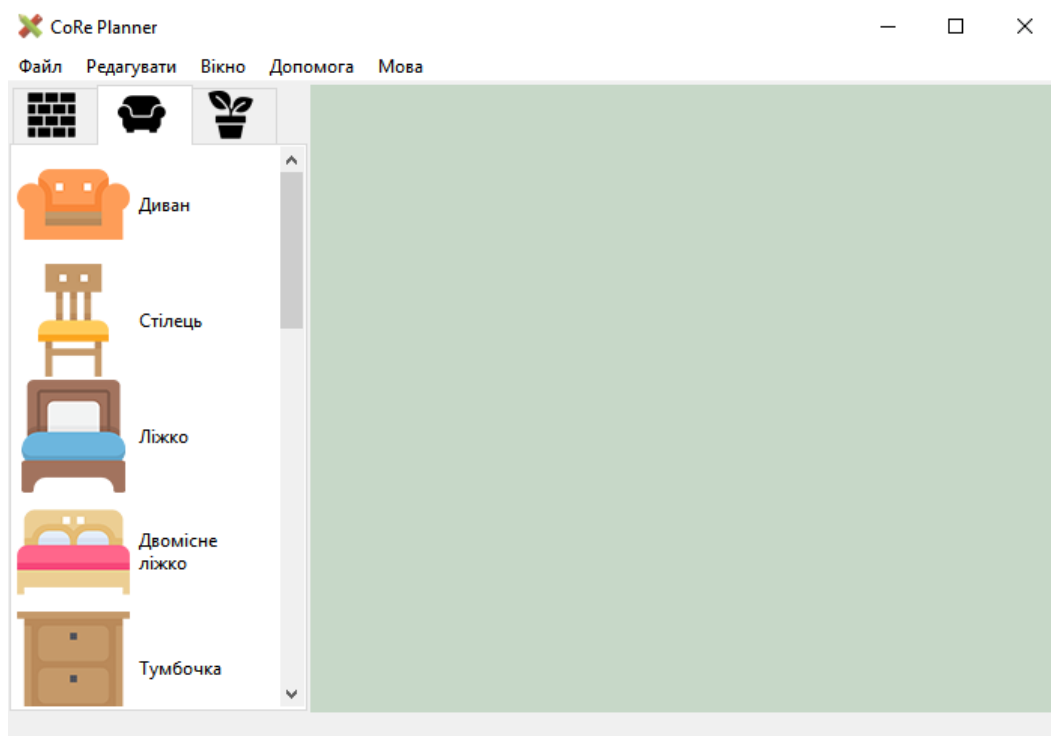


Рисунок 5.2 – Головна форма програми

Після створення нового проекту або відкриття вже існуючого, використовуючи пункти меню «Файл», у області відкритих проектів відобразиться сцена активного проекту планувальника (рис. 5.3). Для додавання нових об'єктів до проекту необхідно скористатися списком інструментів у лівій частині головної форми (натиснути лівою клав'яшею миші на необхідний об'єкт).

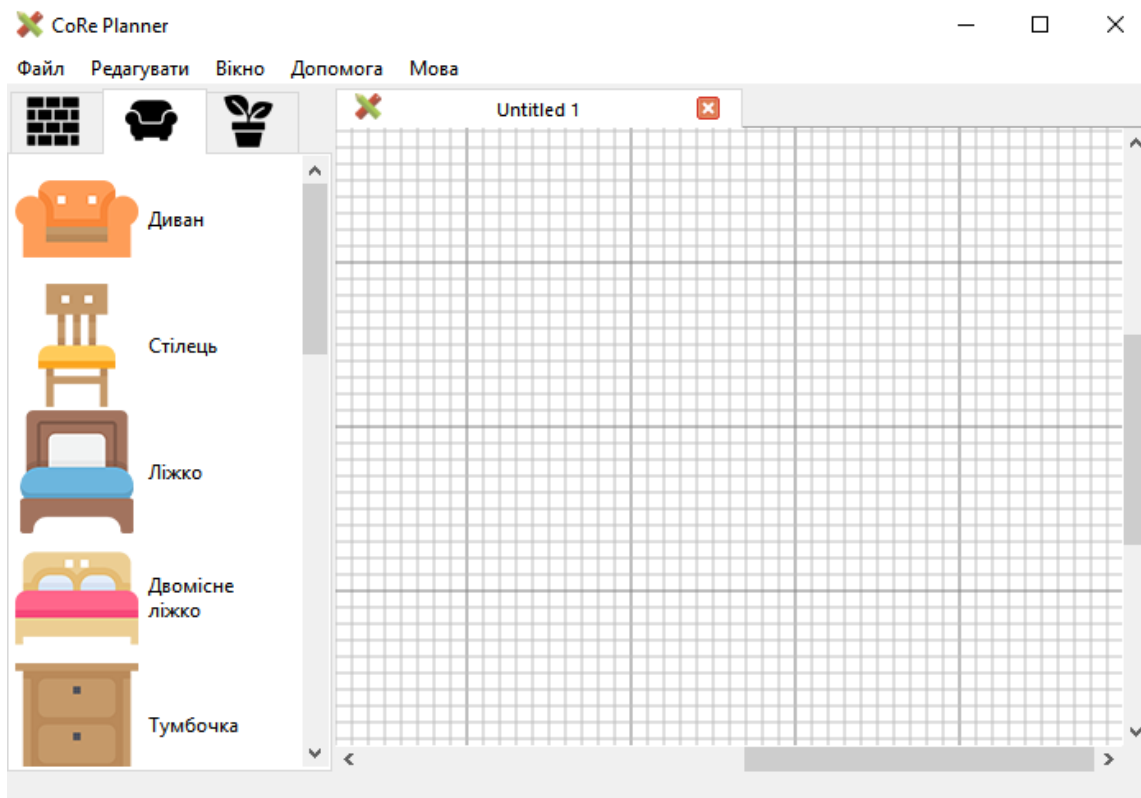


Рисунок 5.3 – Головна форма з відкритим проектом

Для створення нового проекту чи відкриття нещодавнього, користувач повинен натиснути пункт меню «Новий» на панелі головного меню. Після цього з'явиться форма створення проекту де користувач має змогу вказати розміри сцени проекту або відкрити нещодавній попередньо збережений проект.

Ім'я	Дата	Площа
qwerty.json	2018.12.26 04:34	100
Heroes.json	2018.12.26 04:35	100

Buttons: Створити, Відкрити

Рисунок 5.4 – Форма створення нового проекту або відкриття нещодавнього

Для точного вказання параметрів об'єкта необхідно обрати цей об'єкт на сцені та викликати «гарячу клавішу» Ctrl+E або обрати пункт головного меню «Змінити розмір». Це викликає форму, яка відображає поточні значення розміру, положення та куту нахилу об'єкта. У цій формі можна вказати нові значення параметрів та застосувати їх.

Buttons: Застосувати

Рисунок 5.5 – Форма редагування властивостей об'єкта

Для перегляду інформації про програмний продукт та отримання контактів розробників існує спеціальна форма. Для її виклику необхідно обрати пункт головного меню «Про CoRe Planner...».

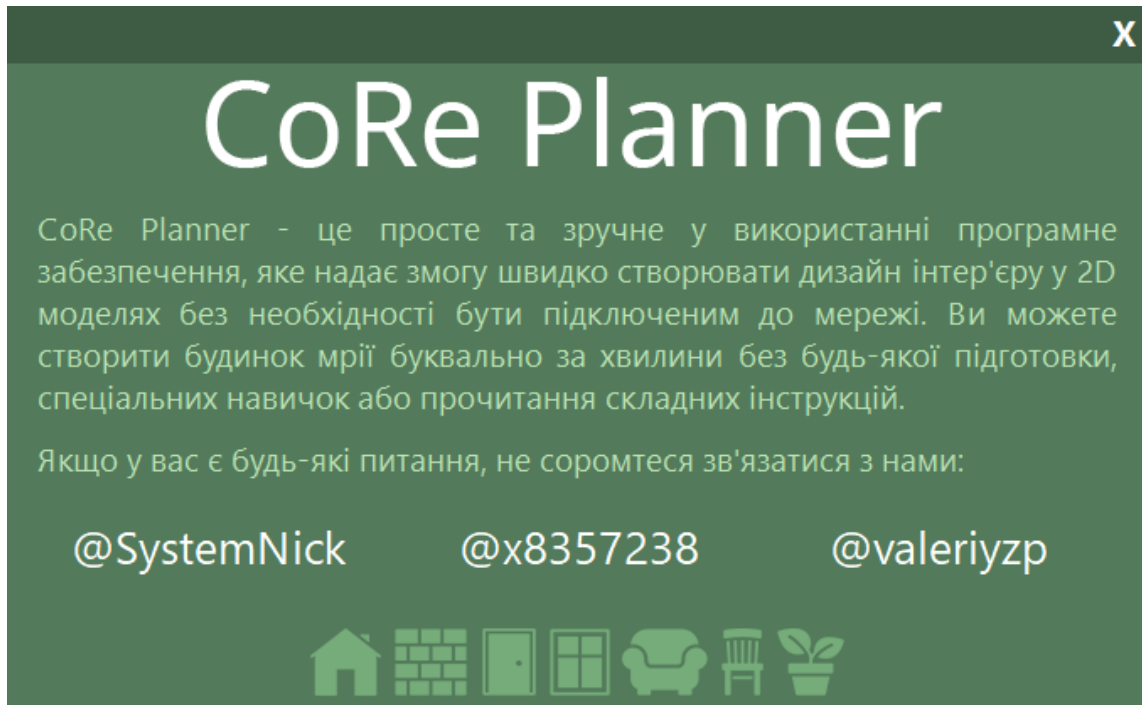


Рисунок 5.5 – Форма зворотного зв'язку

5.5 Повідомлення користувачу

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі користувач отримає інформаційні повідомлення щодо деталей помилки.

У системі реалізовано функції винятків, тому усі непередбачені ситуації оброблюються та програма продовжує стабільно працювати. Одне з можливих повідомлень, що можуть бути отримані при роботі – помилка відкриття файлу, така помилка може виникнути у випадку, якщо файл з інформацією знаходиться в невірній директорії або його пошкоджено. Ілюстрація повідомлення приведена на рис. 5.6.

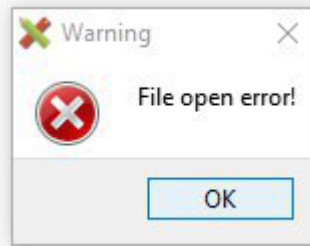


Рисунок 5.6 – Помилка відкриття файлу для читання

ВИСНОВКИ

Під час виконання курсового проекту було розроблено «Програмне забезпечення для проектування та редагування плану приміщень та внутрішнього інтер'єру».

Створена система організовує систему для забезпечення можливості проектувати нового плану приміщення або завантаження вже існуючого, додавання та видалення елементів інтер'єру, зберігання проекту у різних форматах. Основними характеристиками графічних об'єктів проектів є їх найменування, тип, розмір, розміщення та кут нахилу.

Також було виконано огляд та аналіз існуючих методів вирішення завдання та існуючих програмних засобів, таких як Smart Draw, Floor Planner, Planner 5D.

Вирішені наступні завдання курсового проекту:

- виконано огляд сучасних програмних засобів ведення нотаток;
- розглянуті можливості мови програмування C++, середовища розробки Qt Creator;
- розроблено програмне забезпечення для проектування та редагування плану приміщень та внутрішнього інтер'єру, яке повинне надавати користувачу можливість проектувати новий план приміщення або завантажувати вже існуючий, додавати та прибирати елементи інтер'єру, зберігати проект у різних форматах.

ПЕРЕЛІК ПОСИЛАНЬ

1. C++ – Вікіпедія: [Електрон. ресурс]. –
Режим доступу: <https://uk.wikipedia.org/wiki/C%2B%2B>.
2. Особливості C++: [Електрон. ресурс]. –
Режим доступу: <http://test.tanet.edu.te.ua/articles/category/61/message/4398/>
3. Qt Creator – Вікіпедія: [Електрон. ресурс]. –
Режим доступу: https://uk.wikipedia.org/wiki/Qt_Creator.
4. Табунщик, Г.В. Методичні вказівки до виконання курсової роботи з дисципліни “Об’єктно-орієнтоване програмування” для студентів напряму підготовки 6.050103 «Програмна інженерія» всіх форм навчання. Частина 2. Основні теоретичні відомості [Текст] /Укл.: Г.В. Табунщик, Г.В. Неласа, Н.О. Миронова – Запоріжжя: ЗНТУ, 2010. – 70 с.
5. Все классы Qt (главный указатель): [Електрон. ресурс]. –
Режим доступа: <http://www.doc.crossplatform.ru/qt/4.6.x/classes.html>.

ДОДАТОК А

Текст програми

A.1 Текст файлу aboutwindow.h

```
#ifndef ABOUTWINDOW_H
#define ABOUTWINDOW_H

#include <QtWidgets>
#include "ui_aboutwindow.h"

namespace Ui {
class AboutWindow;
}

class AboutWindow : public QDialog
{
    Q_OBJECT

public:
    explicit AboutWindow(QWidget *parent = 0);
    ~AboutWindow();

private slots:
    void on_pushButtonClose_clicked();
    void on_labelSystemNick_linkActivated(const QString &link);
    void on_labelx8357238_linkActivated(const QString &link);
    void on_labelvaleriyzp_linkActivated(const QString &link);

private:
    Ui::AboutWindow *ui;
    QPixmap backgroundImage;
    void setupOtherUi();
};

#endif // ABOUTWINDOW_H
```

A.2 Текст файлу aboutwindow.cpp

```
#include "aboutwindow.h"

AboutWindow::AboutWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::AboutWindow)
{
    ui->setupUi(this);
    setupOtherUi();
}

AboutWindow::~AboutWindow()
{
    delete ui;
}

void AboutWindow::setupOtherUi()
{
    setWindowFlags(Qt::Window | Qt::FramelessWindowHint);
    ui->pushButtonClose->setStyleSheet("QPushButton {color: white;}");
}
```

```

void AboutWindow::on_pushButtonClose_clicked()
{
    close();
}

void AboutWindow::on_labelSystemNick_linkActivated(const QString &link)
{
    QDesktopServices::openUrl(QUrl(link));
}

void AboutWindow::on_labelx8357238_linkActivated(const QString &link)
{
    QDesktopServices::openUrl(QUrl(link));
}

void AboutWindow::on_labelvaleriyzp_linkActivated(const QString &link)
{
    QDesktopServices::openUrl(QUrl(link));
}

```

A.3 Текст файла database.h

```

#ifndef DATABASE_H
#define DATABASE_H

#include <QtSql>

#define DATABASE_NAME        "FileHistory.db"
#define TABLE               "FileHistory"
#define TABLE_FNAME        "Name"
#define TABLE_FDATE        "Date"
#define TABLE_FWAY         "Way"
#define TABLE_FSQUARE      "Square"

struct InfoAboutFile {
public:
    QString FileName;
    QString Way;
    QString DateOfSave;
    double Square;
};

class DataBase
{
private:
    QSqlDatabase dataBase;
    int maxItems;
    int itemsLeft;
    int itemsLeftDataBase();
    void createTable();
    void openDataBase();
    void restoreDataBase();
    void removeRecord(const QString &fWay);

public:
    DataBase();
    ~DataBase();
    void connectToDataBase();
    void insertIntoTable(const QString &fName, const QString &fData, const
    QString &fWay, const double &fSquare);
    void WriteTableToArray(InfoAboutFile *Array, int MaximumCount, int
    *CurrentCount);
};

```

```
#endif // DATABASE_H
```

A.4 Текст файла database.cpp

```
#include "database.h"
DataBase::DataBase()
{
    maxItems = 5;
}

DataBase::~DataBase()
{
    dataBase.close();
}

void DataBase::connectToDataBase()
{
    if(!QFile(DATABASE_NAME).exists())
    {
        this->restoreDataBase();
    }
    else
    {
        this->openDataBase();
    }
}

void DataBase::restoreDataBase()
{
    this->openDataBase();
    this->createTable();
}

void DataBase::openDataBase()
{
    dataBase = QSqlDatabase::addDatabase("SQLITE");
    dataBase.setDatabaseName(DATABASE_NAME);
    try {
        if(!dataBase.open())
        {
            throw QString("NOT OPEN SQLITE: " DATABASE_NAME);
        }
    }
    catch (QString error) {
        qDebug() << error;
    }
}

void DataBase::createTable()
{
    QSqlQuery query;
    if(!query.exec( "CREATE TABLE " TABLE " ("
                    TABLE_FNAME      " BLOB      NOT NULL,"
                    TABLE_FDATE      " VARCHAR(50)  NOT NULL,"
                    TABLE_FWAY       " BLOB      NOT NULL,"
                    TABLE_FSQUARE    " REAL      NOT NULL"
                    " )"
                    ))
    {
        throw QString("NOT CREATE TABLE SQLITE: " TABLE);
    }
}
```

```

}

void DataBase::insertIntoTable(const QString&fName,const QString&fData,const
QString&fWay,const double&fSquare)
{
    removeRecord(fWay);
    itemsLeft = itemsLeftDataBase();
    QSqlQuery query;
    if(itemsLeft < 0)
    {
        query.exec("SELECT * FROM " TABLE );
        query.first();
        removeRecord(query.value(2).toString());
    }
    query.prepare("INSERT INTO " TABLE " ( " TABLE_FNAME ", "
                                                         TABLE_FDATE ", "
                                                         TABLE_FWAY ", "
                                                         TABLE_FSQUARE" ) "
                  "VALUES (:FName, :FData, :FWay, :FSquare)");
    query.bindValue(":FName", fName);
    query.bindValue(":FData", fData);
    query.bindValue(":FWay", fWay);
    query.bindValue(":FSquare", fSquare);
    if(!query.exec())
    {
        throw QString("ERROR INSERT INTO SQLITE TABLE:" TABLE);
    }
}

void DataBase::WriteTableToArray(InfoAboutFile *Array, int MaximumCount, int
*CurrentCount)
{
    QSqlQuery query;
    query.exec("SELECT * FROM " TABLE );
    try{
        if(!query.exec())
        {
            throw QString("ERROR WRITE TO ARRAY SQLITE: " TABLE);
        }

        while (query.next() && *CurrentCount < MaximumCount)
        {
            QFile file;
            file.setFileName(query.value(2).toString());
            if(file.open(QIODevice::ReadOnly|QFile::Text))
            {
                file.close();

                Array[*CurrentCount].FileName = query.value(0).toString();
                Array[*CurrentCount].Way = query.value(2).toString();
                Array[*CurrentCount].DateOfSave = query.value(1).toString();
                Array[*CurrentCount].Square = query.value(3).toDouble();
                *CurrentCount += 1;
            }
            else
            {
                removeRecord(query.value(2).toString());
            }
        }
    }
    catch (QString error) {
        qDebug() <<error;
    }
}

```

```

}

void DataBase::removeRecord(const QString&fWay)
{
    try{
        QSqlQuery query;
        query.prepare("DELETE FROM " TABLE " WHERE " TABLE_FWAY "= ' " +
fWay+"'"");
        query.bindValue(TABLE_FWAY, fWay);
        if(!query.exec())
        {
            throw QString("ERROR REMOVE FROM SQLITE: " TABLE);
        }
    }
    catch (QString error) {
        qDebug()<<error;
    }
}

int DataBase::itemsLeftDataBase() {
    int res = maxItems-1;
    QSqlQuery query;
    query.exec("SELECT * FROM " TABLE );
    while(query.next())
        res--;
    return res;
}

```

A.5 Текст файла itemeditwindow.h

```

#ifndef ITEMEDITWINDOW_H
#define ITEMEDITWINDOW_H

#include <QWidget>
#include "ui_itemeditwindow.h"
#include "mdichild.h"

namespace Ui {
class ItemEditWindow;
}

class ItemEditWindow : public QWidget
{
    Q_OBJECT

public:
    explicit ItemEditWindow(QWidget *parent = 0);
    ItemEditWindow(MainWindow *Main, MdiChild *Parent, QWidget *parent = 0);
    ~ItemEditWindow();

private slots:
    void on_pushButtonApply_clicked();

private:
    Ui::ItemEditWindow *ui;
    MdiChild *FormWithScene;
    MainWindow *MainWin;
};

#endif // ITEMEDITWINDOW_H

```

A.6 Текст файлу itemeditwindow.cpp

```
#include "itemeditwindow.h"

ItemEditWindow::ItemEditWindow(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ItemEditWindow)
{
    ui->setupUi(this);
}

ItemEditWindow::~ItemEditWindow()
{
    delete ui;
}

void ItemEditWindow::on_pushButtonApply_clicked()
{
    if(ui->doubleSpinBoxWidth->value()*100 >= 20)
        if(ui->doubleSpinBoxHeight->value()*100 >= 20)
            if(ui->doubleSpinBoxPosX->value()*100 >= 0 && ui->doubleSpinBoxPosY->value()*100 <= FormWithScene->sceneRect().toRect().width())
                if(ui->doubleSpinBoxAngle->value()*100 >= 0 && ui->doubleSpinBoxAngle->value()*100 <= FormWithScene->sceneRect().toRect().height())
                {
                    FormWithScene->LastActive->Width = ui->doubleSpinBoxWidth->value()*100;
                    FormWithScene->LastActive->Height = ui->doubleSpinBoxHeight->value()*100;
                    FormWithScene->LastActive->setPos(ui->doubleSpinBoxPosX->value()*100, ui->doubleSpinBoxPosY->value()*100);
                    FormWithScene->LastActive->setRotation(ui->doubleSpinBoxAngle->value());

                    if(FormWithScene->AfterUndo)
                    {
                        FormWithScene->SaveFileInfo.DeleteHighVersions(FormWithScene->VersionOfScene);
                        FormWithScene->AfterUndo = false;
                        FormWithScene->MaximumVersionOfScene = FormWithScene->VersionOfScene;
                    }
                    FormWithScene->VersionOfScene++;
                    FormWithScene->MaximumVersionOfScene++;
                    FormWithScene->SaveFileInfo.SaveToHistory();
                    FormWithScene->SaveFileInfo.LoadFromHistory(FormWithScene->VersionOfScene);

                    this->close();
                }
}

ItemEditWindow::ItemEditWindow(MainWindow *Main, MdiChild *Parent, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::ItemEditWindow)
{
    ui->setupUi(this);
    FormWithScene = Parent;
    MainWin = Main;
}
```

```

ui->doubleSpinBoxWidth->setValue(Parent->LastActive->GetWidth()/100);
ui->doubleSpinBoxHeight->setValue(Parent->LastActive->GetHeight()/100);
ui->doubleSpinBoxPosX->setValue(Parent->LastActive->pos().x()/100);
ui->doubleSpinBoxPosY->setValue(Parent->LastActive->pos().y()/100);
ui->doubleSpinBoxAngle->setValue(Parent->LastActive->rotation());

QPixmap NullPicture = QPixmap(1,1);
NullPicture.fill(QColor(0,0,0,0));
setWindowIcon(QIcon(NullPicture));
}

```

A.7 Текст файлу fileopen.h

```

#ifndef FILEOPEN_H
#define FILEOPEN_H

#include "fileway.h"
#include "jsondata.h"

class MainWindow;

class MdiChild;

class FileOpen : public FileWay
{
private:
    MainWindow *Main;

public:
    FileOpen(MainWindow *Parent);
    void SetWay();
    void OpenFile();
    void OpenFileByWay(QString &Way, QString &FileName);
};

#endif // FILEOPEN_H

```

A.8 Текст файлу fileopen.cpp

```

#include "fileopen.h"
#include "mdichild.h"
#include "mainwindow.h"

FileOpen::FileOpen(MainWindow *Parent) : FileWay()
{
    Main = Parent;
    FileName = "";
    Way = "";
}

void FileOpen::SetWay()
{
    Way = QFileDialog::getOpenFileName(nullptr, "Відкрити... | Open... |  
Открыть...", "D:\\", "*.json");
    FileName = GetFileNameFromWay(Way);
}

void FileOpen::OpenFile()
{
    try {

```



```

        SetWay();
        if (FileName == "" || Way == "")
        {
            throw QString("File open error!");
        }
        MdiChild *Child = new MdiChild(Main);
        Child->setWindowTitle(FileName);
        Child->setWindowIcon(QIcon(":/menu/cube"));
        Child->SaveFileInfo.Way = Way;
        Child->SaveFileInfo.FileName = FileName;

        JsonData JsonReader;
        JsonReader.inFromFile(Way, Child);
        Main->loadProject(Child);

        Child->SaveFileInfo.InitHistory();
        Child->VersionOfScene = 0;
        Child->MaximumVersionOfScene = 0;
    }
    catch (QString error) {
        QMessageBox::critical(Main, "Warning", error);
        qDebug() << error;
    }
}

void FileOpen::OpenFileByWay(QString &Way, QString &FileName)
{
    MdiChild *Child = new MdiChild(Main);
    Child->setWindowTitle(FileName);
    Child->setWindowIcon(QIcon(":/menu/cube"));
    Child->SaveFileInfo.Way = Way;
    Child->SaveFileInfo.FileName = FileName;

    JsonData JsonReader;
    JsonReader.inFromFile(Way, Child);
    Main->loadProject(Child);

    Child->SaveFileInfo.InitHistory();
    Child->VersionOfScene = 0;
    Child->MaximumVersionOfScene = 0;
}

```

A.9 Текст файла filesave.h

```

#ifndef FILESAVE_H
#define FILESAVE_H

#include "fileway.h"
#include "database.h"

class MdiChild;

class MainWindow;

class FileSave : public FileWay
{
public:
    MdiChild *ParenWithScene;
    FileSave();
    ~FileSave();
    void Save(MainWindow *Main);
    void SaveAs(MainWindow *Main);
    void SaveToHistory();
}

```

```

    void InitHistory();
    void LoadFromHistory(int Version);
    void DeleteHighVersions(int Version);

private:
    void SetWay();
    QString HistoryFileWay;
};

#endif // FILESAVE_H

```

A.10 Текст файла filesave.cpp

```

#include "filesave.h"
#include "mdichild.h"
#include "mainwindow.h"
#include "jsondata.h"

FileSave::FileSave() : FileWay()
{
    ParenWithScene = NULL;
    QDate DateNow = QDate::currentDate();
    QTime TimeNow = QTime::currentTime();
    HistoryFileWay = DateNow.toString("yyyyMMdd") +
    TimeNow.toString("hhmmss") + ".json";
}

void FileSave::SetWay()
{
    QString adress = QFileDialog::getSaveFileName(nullptr, "Зберіть в... |
Save to... | Сохранить в...", "D:\\", "Project files (*.json);;JPG file
(*.jpg);;BMP file (*.bmp);;PNG file (*.png)");
    try {
        if(adress == "")
        {
            throw "ERROR ADDRESS IS CLEAR";
        }
        Way = adress;
        FileName = GetFileNameFromWay(adress);
    }
    catch (QString error) {
        qDebug() << error;
    }
}

FileSave::~FileSave()
{
    QFile file;
    file.setFileName(HistoryFileWay);
    if(file.open(QIODevice::ReadWrite|QFile::Text))
    {
        file.remove();
    }
}

void FileSave::Save(MainWindow *Main)
{
    if(FileName != "" && Way != "")
    {
        JsonData JsonWriter;
        JsonWriter.outToFile(Way, ParenWithScene);

        QDate DateNow = QDate::currentDate();
    }
}

```

```

        QTime TimeNow = QTime::currentTime();
        QString Date = DateNow.toString("yyyy.MM.dd ") +
TimeNow.toString("hh:mm");
        double Square = (ParenWithScene->scene->sceneRect().toRect().width() *
ParenWithScene->scene->sceneRect().toRect().height()) / 10000.0;
        Main->database.insertIntoTable(FileName, Date, Way, Square);
    }
    else
    {
        SaveAs(Main);
    }
}

void FileSave::SaveAs(MainWindow *Main)
{
    SetWay();
    if(Way != "" && FileName != "")
    {
        ParenWithScene->setWindowTitle(FileName);
        if(FileName[FileName.length()-1] == 'n')
        {
            JsonData JsonWriter;
            JsonWriter.outToFile(Way, ParenWithScene);
            ParenWithScene->setWindowTitle(FileName);

            QDate DateNow = QDate::currentDate();
            QTime TimeNow = QTime::currentTime();
            QString Date = DateNow.toString("yyyy.MM.dd ") +
TimeNow.toString("hh:mm");
            double Square = (ParenWithScene->scene-
>sceneRect().toRect().width() * ParenWithScene->scene-
>sceneRect().toRect().height()) / 10000.0;
            Main->database.insertIntoTable(FileName, Date, Way, Square);
        }
        else
        {
            ParenWithScene->SaveAsImage(Way);
        }
    }
}

void FileSave::SaveToHistory()
{
    JsonData JsonWriter;
    JsonWriter.outHistoryToFile(HistoryFileWay, ParenWithScene);
}

void FileSave::InitHistory()
{
    JsonData JsonWriter;
    JsonWriter.outInitHistoryFile(HistoryFileWay, ParenWithScene);
}

void FileSave::LoadFromHistory(int Version)
{
    JsonData JsonWriter;
    JsonWriter.InitScenFromFile(HistoryFileWay, ParenWithScene, Version);
}

void FileSave::DeleteHighVersions(int Version)
{
    JsonData JsonWriter;
    JsonWriter.DeleteHighVersions(HistoryFileWay ,Version);
}

```

A.11 Текст файла fileway.h

```
#ifndef FILEWAY_H
#define FILEWAY_H

#include <QString>

class FileWay
{
public:
    QString Way;
    QString FileName;

    QString GetFileNameFromWay(QString &AdressFile);
    FileWay();
    FileWay(QString &WayToFile);
    virtual ~FileWay();
    virtual void SetWay() = 0;
};

#endif // FILEWAY_H
```

A.12 Текст файла fileway.cpp

```
#include "fileway.h"

FileWay::FileWay()
{
    Way = "";
    FileName = "";
}

FileWay::FileWay(QString &WayToFile)
{
    Way = WayToFile;
    FileName = GetFileNameFromWay(Way);
}

FileWay::~FileWay()
{
}

QString FileWay::GetFileNameFromWay(QString &AdressFile)
{
    QString NameFile = "";
    for(int i = AdressFile.length()-1; i >= 0; i--)
    {
        if(AdressFile[i] == '\\\\' || AdressFile[i] == '/')
        {
            for(int j = i+1; j < AdressFile.length(); j++)
                NameFile += AdressFile[j];
            break;
        }
    }
    return NameFile;
}
```

A.13 Текст файла formcreate.h

```

#ifndef FORMCREATE_H
#define FORMCREATE_H

#include <QWidget>
#include "ui_formcreate.h"
#include "mdichild.h"
#include "database.h"

namespace Ui {
class FormCreate;
}

class MainWindow;

class FormCreate : public QWidget
{
    Q_OBJECT

public:
    explicit FormCreate(MainWindow *Main, QWidget *parent = 0);
    ~FormCreate();

private slots:
    void on_pushButtonCreateNew_clicked();
    void on_pushButtonProjectName_clicked();
    void on_pushButtonProjectTime_clicked();
    void on_pushButtonProjectSquare_clicked();
    void SetFilesButtons();
    void SetButtonsTextAlign();
    void on_pushButton_1_clicked();
    void on_pushButton_2_clicked();
    void on_pushButton_3_clicked();
    void on_pushButton_4_clicked();
    void on_pushButton_5_clicked();
    void on_pushButtonOpen_clicked();

private:
    InfoAboutFile *Files;
    int CountOfFiles;
    int WayToOpen;
    MainWindow *MainWin;
    Ui::FormCreate *ui;
};

#endif // FORMCREATE_H

```

A.14 Текст файла formcreate.cpp

```

#include "formcreate.h"
#include "mainwindow.h"

FormCreate::FormCreate(MainWindow *Main, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::FormCreate)
{
    ui->setupUi(this);
    Files = new InfoAboutFile[5];
    CountOfFiles = 0;
    MainWin = Main;

    MainWin->database.WriteTableToArray(Files, 5, &CountOfFiles);
    SetFilesButtons();
    WayToOpen = -1;
}

```

```

        QPixmap NullPicture = QPixmap(1,1);
        NullPicture.fill(QColor(0,0,0,0));
        setWindowIcon(QIcon(NullPicture));
    }

FormCreate::~FormCreate()
{
    delete ui;
}

void FormCreate::on_pushButtonCreateNew_clicked()
{
    if(ui->doubleSpinBoxWidth->value() > 0)
        if(ui->doubleSpinBoxHeight->value() > 0)
        {
            MdiChild *child = new MdiChild(MainWin);
            child->setWindowTitle("Untitled " + QString::number(MainWin-
>uniqueProjectID++));
            child->SaveFileInfo.FileName = child->windowTitle();
            child->setWindowIcon(QIcon(":/menu/cube"));
            child->SetSceneParametrs(int(ui->doubleSpinBoxWidth-
>value()*100), int(ui->doubleSpinBoxHeight->value()*100));

            MainWin->loadProject(child);
            this->close();
        }
}

void SwapInfoAboutFile(InfoAboutFile *File1, InfoAboutFile *File2)
{
    InfoAboutFile temp;
    temp.DateOfSave = File1->DateOfSave;
    temp.FileName = File1->FileName;
    temp.Way = File1->Way;
    temp.Square = File1->Square;

    File1->DateOfSave = File2->DateOfSave;
    File1->FileName = File2->FileName;
    File1->Way = File2->Way;
    File1->Square = File2->Square;

    File2->DateOfSave = temp.DateOfSave;
    File2->FileName = temp.FileName;
    File2->Way = temp.Way;
    File2->Square = temp.Square;
}

void ShellSortByDate(InfoAboutFile *File, int Count)
{
    for (int gap = Count/2; gap > 0; gap /= 2)
    {
        for (int i = gap; i < Count; ++i)
        {
            for (int j = i-gap; j >= 0; j -= gap)
            {
                if (File[j+gap].DateOfSave <= File[j].DateOfSave) break;
                else
                {
                    SwapInfoAboutFile(&File[j], &File[j+gap]);
                }
            }
        }
    }
}

```

```

}

void BubbleSortBySquare(InfoAboutFile *File, int Count)
{
    for(int i = 0; i < Count-1; i++)
        for(int j = 0; j < Count-1; j++)
            if(File[j].Square > File[j+1].Square)
                SwapInfoAboutFile(&File[j], &File[j+1]);
}

void InsertionSortByFileName(InfoAboutFile *File, int Count)
{
    int i, j;
    InfoAboutFile key;

    for(i = 0; i < Count; i++)
    {
        j = i-1;
        while(j >= 0 && File[j+1].FileName < File[j].FileName)
        {
            SwapInfoAboutFile(&File[j], &File[j+1]);
            j--;
        }
    }
}

void FormCreate::on_pushButtonProjectName_clicked()
{
    InsertionSortByFileName(Files, CountOfFiles);
    SetFilesButtons();
}

void FormCreate::on_pushButtonProjectTime_clicked()
{
    ShellSortByDate(Files, CountOfFiles);
    SetFilesButtons();
}

void FormCreate::on_pushButtonProjectSquare_clicked()
{
    BubbleSortBySquare(Files, CountOfFiles);
    SetFilesButtons();
}

void FormCreate::SetFilesButtons()
{
    if(CountOfFiles >= 1)    ui->pushButton_1-
>setText((Files[0].FileName).leftJustified(16, ' ') + " " +
(Files[0].DateOfSave).leftJustified(16, ' ') + " " +
(QVariant(Files[0].Square).toString()).rightJustified(10, ' '));
    else ui->pushButton_1->setVisible(false);
    if(CountOfFiles >= 2)    ui->pushButton_2-
>setText((Files[1].FileName).leftJustified(16, ' ') + " " +
(Files[1].DateOfSave).leftJustified(16, ' ') + " " +
(QVariant(Files[1].Square).toString()).rightJustified(10, ' '));
    else ui->pushButton_2->setVisible(false);
    if(CountOfFiles >= 3)    ui->pushButton_3-
>setText((Files[2].FileName).leftJustified(16, ' ') + " " +
(Files[2].DateOfSave).leftJustified(16, ' ') + " " +
(QVariant(Files[2].Square).toString()).rightJustified(10, ' '));
    else ui->pushButton_3->setVisible(false);
    if(CountOfFiles >= 4)    ui->pushButton_4-
>setText((Files[3].FileName).leftJustified(16, ' ') + " " +

```

```

    (Files[3].DateOfSave).leftJustified(16, ' ') + " " +
    (QVariant(Files[3].Square).toString()).rightJustified(10, ' '));
    else ui->pushButton_4->setVisible(false);
    if(CountOfFiles >= 5) ui->pushButton_5-
>setText((Files[4].FileName).leftJustified(16, ' ') + " " +
(Files[4].DateOfSave).leftJustified(16, ' ') + " " +
(QVariant(Files[4].Square).toString()).rightJustified(10, ' '));
    else ui->pushButton_5->setVisible(false);
    SetButtonsTextAlign();
}

void FormCreate::SetButtonsTextAlign()
{
    ui->pushButton_1->setStyleSheet("text-align: left;");
    ui->pushButton_2->setStyleSheet("text-align: left;");
    ui->pushButton_3->setStyleSheet("text-align: left;");
    ui->pushButton_4->setStyleSheet("text-align: left;");
    ui->pushButton_5->setStyleSheet("text-align: left;");
}

void FormCreate::on_pushButton_1_clicked()
{
    WayToOpen = 0;
    SetButtonsTextAlign();
    ui->pushButton_1->setStyleSheet("background-color: rgb(173, 255, 153);
text-align: left;");
}

void FormCreate::on_pushButton_2_clicked()
{
    WayToOpen = 1;
    SetButtonsTextAlign();
    ui->pushButton_2->setStyleSheet("background-color: rgb(173, 255, 153);
text-align: left;");
}

void FormCreate::on_pushButton_3_clicked()
{
    WayToOpen = 2;
    SetButtonsTextAlign();
    ui->pushButton_3->setStyleSheet("background-color: rgb(173, 255, 153);
text-align: left;");
}

void FormCreate::on_pushButton_4_clicked()
{
    WayToOpen = 3;
    SetButtonsTextAlign();
    ui->pushButton_4->setStyleSheet("background-color: rgb(173, 255, 153);
text-align: left;");
}

void FormCreate::on_pushButton_5_clicked()
{
    WayToOpen = 4;
    SetButtonsTextAlign();
    ui->pushButton_5->setStyleSheet("background-color: rgb(173, 255, 153);
text-align: left;");
}

void FormCreate::on_pushButtonOpen_clicked()
{
    if(WayToOpen != -1)
    {

```



```

        FileOpen OpenFile(MainWin);
        OpenFile.OpenFileByWay(Files[WayToOpen].Way,
Files[WayToOpen].FileName);
        this->close();
    }
}

```

A.15 Текст файла graphicitem.h

```

#ifndef GRAPHICITEM_H
#define GRAPHICITEM_H

#include <QObject>
#include <QGraphicsItem>
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QCursor>
#include <QtMath>

#include "points.h"

struct GraphicItemInfo{
    QString Type;
    QString Value;
    double X;
    double Y;
    double Width;
    double Height;
    double Angle;
};

enum CornerFlags{
    Top = 1,
    Bottom = 2,
    Left = 4,
    Right = 8,
    TopLeft = Top|Left,
    TopRight = Top|Right,
    BottomLeft = Bottom|Left,
    BottomRight = Bottom|Right
};

class MdiChild;

class GraphicItem : public QObject, public QGraphicsItem
{
public:
    points MiddlePoints;
    double Width;
    double Height;
    int FlagCorner;
    bool IsEditNow;
    bool IsResizeModNow;
    int SizeOfPoint;
    double MinimumWidth;
    double MinimumHeight;
    double CurrentAngleOfRotation;
    QPointF ClickedPointOnScene;
    QPointF m_shiftMouseCoords;
    MdiChild *ParenWithScene;
    bool IsSceneEdited;

```

```

    GraphicItem(double setWidth, double setHeight, MdiChild *MdiChildParent,
QObject *parent = 0);
    ~GraphicItem();
    virtual QString GetType() = 0;
    double GetHeight() {return Height;}
    double GetWidth() {return Width;}
    double GetRotationAngle() {return this->rotation();}
    void SetRotationAngle(double RotationAngleToSet) {this-
>setRotation(RotationAngleToSet);}
    void SetHeight(double HeightToSet) {Height = HeightToSet > MinimumHeight
? HeightToSet : MinimumHeight;}
    void SetWidth(double WidthToSet) {Width = WidthToSet > MinimumWidth ?
WidthToSet : MinimumWidth;}
    void SetUnActive() {IsEditNow = false;}
    virtual GraphicItemInfo GetItemInfo() = 0;
    QRectF boundingRect() const;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget) = 0;
    void mouseMoveEvent(QGraphicsSceneMouseEvent *event);
    void mousePressEvent(QGraphicsSceneMouseEvent *event);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent *event);
    void mouseDoubleClickEvent(QGraphicsSceneMouseEvent *event);
    void hoverMoveEvent(QGraphicsSceneHoverEvent *event);
    void ResetCursor(double CursorX, double CursorY, bool MouseMove);
    void SetPoints();
};

#endif // GRAPHICITEM_H

```

A.16 Текст файла graphicitem.cpp

```

#include "graphicitem.h"
#include "mdichild.h"

double GetAngleABC( QPointF A, QPointF B, QPointF C)
{
    QPointF AB( B.x() - A.x(), B.y() - A.y() );
    QPointF CB ( B.x() - C.x(), B.y() - C.y() );
    double ScalarProduct = (AB.x() * CB.x() + AB.y() * CB.y());
    double Proection = (AB.x() * CB.y() - AB.y() * CB.x());
    double RadianAngle = qAtan2(Proection, ScalarProduct);
    return -(RadianAngle * 180. / M_PI + 0.5);
}

GraphicItem::GraphicItem(double setWidth, double setHeight, MdiChild
*MdiChildParent, QObject *parent) : QObject(parent), QGraphicsItem()
{
    IsEditNow = false;
    IsResizeModNow = false;
    SizeOfPoint = 6;
    MinimumWidth = 20;
    MinimumHeight = 20;
    CurrentAngleOfRotation = 0;
    SetHeight(setHeight);
    SetWidth(setWidth);
    ParenWithScene = MdiChildParent;
    setAcceptHoverEvents(true);
    IsSceneEdited = false;
    if(ParenWithScene->LastActive != NULL && ParenWithScene->LastActive !=
this) ParenWithScene->LastActive->SetUnActive();
    ParenWithScene->LastActive = this;
}

```

```

GraphicItem::~~GraphicItem()
{

}

QRectF GraphicItem::boundingRect() const
{
    return QRectF (-Width/2,-Height/2,Width,Height);
}

void GraphicItem::mouseDoubleClickEvent(QGraphicsSceneMouseEvent *event)
{
    IsEditNow = !IsEditNow;
    update();
}

void GraphicItem::hoverMoveEvent(QGraphicsSceneHoverEvent *event)
{
    ResetCursor(event->pos().x(), event->pos().y(), false);
}

void GraphicItem::mouseMoveEvent(QGraphicsSceneMouseEvent *event)
{
    IsSceneEdited = true;
    if(IsEditNow)
    {
        if(IsResizeModNow)
        {
            double dHeight = Height, dWidth = Width;
            switch(FlagCorner)
            {
                case Top:
                {
                    Height -= (event->pos().y() + Height/2);
                    if(Height >= MinimumHeight) this->setPos (mapToScene(0,
(event->pos().y() + Height/2)));
                    else {Height = MinimumHeight; this-
>setPos (MidlePoints.GetBottomMinimumCenter());}

                    break;
                }
                case Bottom:
                {
                    Height += (event->pos().y() - Height/2);
                    if(Height >= MinimumHeight) this->setPos (mapToScene(0,
(event->pos().y() - Height/2)));
                    else {Height = MinimumHeight; this-
>setPos (MidlePoints.GetTopMinimumCenter());}

                    break;
                }
                case Left:
                {
                    Width -= (event->pos().x() + Width/2);
                    if(Width >= MinimumWidth) this->setPos (mapToScene((event-
>pos().x() + Width/2), 0));
                    else {Width = MinimumWidth; this-
>setPos (MidlePoints.GetRightMinimumCenter());}

                    break;
                }
                case Right:
                {
                    Width += (event->pos().x() - Width/2);

```

```

        if (Width >= MinimumWidth) this->setPos (mapToScene ((event-
>pos().x() - Width/2), 0));
        else {Width = MinimumWidth; this-
>setPos (MiddlePoints.GetLeftMinimumCenter());}

        break;
    }
    case TopLeft:
    {
        Height -= (event->pos().y() + Height/2);
        Width -= (event->pos().x() + Width/2);
        Height = Height < MinimumHeight ? MinimumHeight : Height;
        Width = Width < MinimumWidth ? MinimumWidth : Width;

        dHeight -= Height;
        dWidth -= Width;

        this->setPos (mapToScene (dWidth/2, dHeight/2));

        break;
    }
    case TopRight:
    {
        Height -= (event->pos().y() + Height/2);
        Width += (event->pos().x() - Width/2);
        Height = Height < MinimumHeight ? MinimumHeight : Height;
        Width = Width < MinimumWidth ? MinimumWidth : Width;

        dHeight -= Height;
        dWidth -= Width;

        this->setPos (mapToScene (-dWidth/2, dHeight/2));

        break;
    }
    case BottomLeft:
    {
        Height += (event->pos().y() - Height/2);
        Width -= (event->pos().x() + Width/2);
        Height = Height < MinimumHeight ? MinimumHeight : Height;
        Width = Width < MinimumWidth ? MinimumWidth : Width;

        dHeight -= Height;
        dWidth -= Width;

        this->setPos (mapToScene (dWidth/2, -dHeight/2));

        break;
    }
    case BottomRight:
    {
        Height += (event->pos().y() - Height/2);
        Width += (event->pos().x() - Width/2);
        Height = Height < MinimumHeight ? MinimumHeight : Height;
        Width = Width < MinimumWidth ? MinimumWidth : Width;

        dHeight -= Height;
        dWidth -= Width;

        this->setPos (mapToScene (-dWidth/2, -dHeight/2));

        break;
    }
}
}

```

```

        update();
    }
    else
    {
        double Angle= GetAngleABC(mapToScene(event-
>pos()),mapToScene(0,0),ClickedPointOnScene);
        this->setRotation(CurrentAngleOfRotation+Angle);
    }
}
else
{
    this->setPos(mapToScene(event->pos()).x() + m_shiftMouseCoords.x(),
mapToScene(event->pos()).y()+m_shiftMouseCoords.y());
}

    if (this->pos().x() < 0) this->setPos(0, pos().y());
    else if (this->pos().x() > ParenWithScene->scene-
>sceneRect().toRect().width()) this->setPos(ParenWithScene->scene-
>sceneRect().toRect().width(), pos().y());

    if (this->pos().y() < 0) this->setPos(pos().x(), 0);
    else if (this->pos().y() > ParenWithScene->scene-
>sceneRect().toRect().height()) this->setPos(pos().x(), ParenWithScene-
>scene->sceneRect().toRect().height());
}

void GraphicItem::mousePressEvent(QGraphicsSceneMouseEvent *event)
{
    m_shiftMouseCoords = this->pos() - mapToScene(event->pos());
    CurrentAngleOfRotation=this->rotation();
    ClickedPointOnScene= mapToScene(event->pos());

    ResetCursor(event->pos().x(), event->pos().y(), true);
    SetPoints();

    if(ParenWithScene->LastActive != NULL && ParenWithScene->LastActive !=
this) ParenWithScene->LastActive->SetUnActive();
    ParenWithScene->LastActive = this;

    ParenWithScene->UpdateScene();
}

void GraphicItem::mouseReleaseEvent(QGraphicsSceneMouseEvent *event)
{
    ResetCursor(event->pos().x(), event->pos().y(), false);
    if(IsSceneEdited)
    {
        if(ParenWithScene->AfterUndo)
        {
            ParenWithScene->SaveFileInfo.DeleteHighVersions(ParenWithScene-
>VersionOfScene);
            ParenWithScene->AfterUndo = false;
            ParenWithScene->MaximumVersionOfScene = ParenWithScene-
>VersionOfScene;
        }
        ParenWithScene->VersionOfScene++;
        ParenWithScene->MaximumVersionOfScene++;
        ParenWithScene->SaveFileInfo.SaveToHistory();
        ParenWithScene->SaveFileInfo.LoadFromHistory(ParenWithScene-
>VersionOfScene);
        IsSceneEdited = false;
    }
}

```

```

void GraphicItem::ResetCursor(double CursorX, double CursorY, bool MouseMove)
{
    if(IsEditNow)
    {
        if(CursorX <= -Width/2+SizeOfPoint || CursorX >= Width/2-SizeOfPoint
|| CursorY <= -Height/2+SizeOfPoint || CursorY >= Height/2-SizeOfPoint)
        {
            QString NameOfCursor;
            IsResizeModNow = true;
            FlagCorner = 0;
            if(CursorX <= (-Width/2 + SizeOfPoint)) FlagCorner |= Left;
            if(CursorX >= (Width/2 - SizeOfPoint)) FlagCorner |= Right;
            if(CursorY <= (-Height/2 + SizeOfPoint)) FlagCorner |= Top;
            if(CursorY >= (Height/2 - SizeOfPoint)) FlagCorner |= Bottom;
            switch(FlagCorner)
            {
                case Top:
                case Bottom:
                {
                    NameOfCursor = ":/cursors/ResizeV";
                    break;
                }
                case Left:
                case Right:
                {
                    NameOfCursor = ":/cursors/ResizeH";
                    break;
                }
                case TopLeft:
                case BottomRight:
                {
                    NameOfCursor = ":/cursors/ResizeD1";
                    break;
                }
                case TopRight:
                case BottomLeft:
                {
                    NameOfCursor = ":/cursors/ResizeD2";
                    break;
                }
            }

            QPixmap Item(NameOfCursor);
            QTransform trans = transform();
            trans.rotate(this->rotation());
            Item.transformed(trans);
            this->
setCursor(Item.transformed(trans).scaled(24,24,Qt::KeepAspectRatio,
Qt::SmoothTransformation));
        }
        else
        {
            IsResizeModNow = false;
            this->
setCursor(QPixmap(":/cursors/Rotation").scaled(24,24,Qt::KeepAspectRatio,
Qt::SmoothTransformation));
        }
    }
    else
    {
        if(MouseMove) this->setCursor(Qt::ClosedHandCursor);
        else this->setCursor(Qt::ArrowCursor);
    }
}

```

```

void GraphicItem::SetPoints()
{
    MiddlePoints.SetTopMinimumCenter(mapToScene(0, - Height/2 +
MinimumHeight/2));
    MiddlePoints.SetLeftMinimumCenter(mapToScene(-Width/2 + MinimumWidth/2,
0));
    MiddlePoints.SetRightMinimumCenter(mapToScene(Width - Width/2 -
MinimumWidth/2, 0));
    MiddlePoints.SetBottomMinimumCenter(mapToScene(0, Height - Height/2 -
MinimumHeight/2));
}

```

A.17 Текст файла itembycolor.h

```

#ifndef ITEMBYCOLOR_H
#define ITEMBYCOLOR_H

#include "graphicitem.h"

class ItemByColor : public GraphicItem
{
public:
    ItemByColor(double setWidth, double setHeight, QColor setColor, MdiChild
*MdiChildParent, QObject *parent = 0);
    QString GetType() {return "Color";}
    GraphicItemInfo GetItemInfo();

private:
    QColor ColorOfItem;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

#endif // ITEMBYCOLOR_H

```

A.18 Текст файла itembycolor.cpp

```

#include "itembycolor.h"

ItemByColor::ItemByColor(double setWidth, double setHeight, QColor setColor,
MdiChild *MdiChildParent, QObject *parent)
: GraphicItem(setWidth, setHeight, MdiChildParent, parent)
{
    ColorOfItem = setColor;
}

GraphicItemInfo ItemByColor::GetItemInfo()
{
    GraphicItemInfo Information;
    Information.Angle = this->rotation();
    Information.Width = this->Width;
    Information.Height = this->Height;
    Information.X = this->pos().x();
    Information.Y = this->pos().y();
    Information.Type = "Color";
    Information.Value = ColorOfItem.name();
    return Information;
}

```

```

void ItemByColor::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    painter->save();
    QPen pointpen(Qt::black);
    painter->setPen(Qt::NoPen);
    painter->setBrush(ColorOfItem);
    painter->drawRect(int(-Width/2), int(-Height/2), int(Width),
int(Height));

    if(IsEditNow)
    {
        painter->setPen(Qt::DashLine);
        painter->setBrush(Qt::NoBrush);
        painter->drawRect(int(-Width/2), int(-Height/2), int(Width),
int(Height));
        pointpen.setWidth(SizeOfPoint);
        painter->setPen(pointpen);

        int LeftTopPointX = int(-Width/2+SizeOfPoint/2);
        int LeftTopPointY = int(-Height/2+SizeOfPoint/2);
        int RightBottomPointX = int(Width)%2 == 0 ? int(Width-Width/2-
SizeOfPoint/2) : int(Width-Width/2-SizeOfPoint/2+1);
        int RightBottomPointY = int(Height)%2 == 0 ? int(Height-Height/2-
SizeOfPoint/2) : int(Height-Height/2-SizeOfPoint/2+1);

        painter->drawPoint(LeftTopPointX, LeftTopPointY);
        painter->drawPoint(RightBottomPointX, LeftTopPointY);
        painter->drawPoint(LeftTopPointX, RightBottomPointY);
        painter->drawPoint(RightBottomPointX, RightBottomPointY);
    }
    painter->restore();
}

```

A.19 Текст файлу itembypicture.h

```

#ifndef ITEMBYPICTURE_H
#define ITEMBYPICTURE_H

#include "graphicitem.h"

class ItemByPicture : public GraphicItem
{
public:
    ItemByPicture(double setWidth, double setHeight, QString PictureName,
MdiChild *MdiChildParent, QObject *parent = 0);
    QString GetType() {return "Picture";}
    GraphicItemInfo GetItemInfo();

private:
    QString NameOfPicture;
    QImage PictureOfItem;
    void paint(QPainter *painter, const QStyleOptionGraphicsItem *option,
QWidget *widget);
};

#endif // ITEMBYPICTURE_H

```

A.20 Текст файлу itembypicture.cpp

```

#include "itembypicture.h"

```



```

ItemByPicture::ItemByPicture(double setWidth, double setHeight, QString
PictureName, MdiChild *MdiChildParent, QObject *parent)
    : GraphicItem(setWidth, setHeight, MdiChildParent, parent)
{
    NameOfPicture = PictureName;
    PictureOfItem.load(PictureName);
}

GraphicItemInfo ItemByPicture::GetItemInfo()
{
    GraphicItemInfo Information;
    Information.Angle = this->rotation();
    Information.Width = this->Width;
    Information.Height = this->Height;
    Information.X = this->pos().x();
    Information.Y = this->pos().y();
    Information.Type = "Picture";
    Information.Value = NameOfPicture;
    return Information;
}

void ItemByPicture::paint(QPainter *painter, const QStyleOptionGraphicsItem
*option, QWidget *widget)
{
    painter->save();
    QPen pointpen(Qt::black);
    QPainter PaintPicture;
    painter->drawImage(int(-Width/2), int(-Height/2),
PictureOfItem.scaled(int(Width),int(Height), Qt::IgnoreAspectRatio,
Qt::SmoothTransformation));

    if(IsEditNow)
    {
        painter->setPen(Qt::DashLine);
        painter->setBrush(Qt::NoBrush);
        painter->drawRect(int(-Width/2), int(-Height/2), int(Width),
int(Height));
        pointpen.setWidth(SizeOfPoint);
        painter->setPen(pointpen);

        int LeftTopPointX = int(-Width/2+SizeOfPoint/2);
        int LeftTopPointY = int(-Height/2+SizeOfPoint/2);
        int RightBottomPointX = int(Width)%2 == 0 ? int(Width-Width/2-
SizeOfPoint/2) : int(Width-Width/2-SizeOfPoint/2+1);
        int RightBottomPointY = int(Height)%2 == 0 ? int(Height-Height/2-
SizeOfPoint/2) : int(Height-Height/2-SizeOfPoint/2+1);

        painter->drawPoint(LeftTopPointX, LeftTopPointY);
        painter->drawPoint(RightBottomPointX, LeftTopPointY);
        painter->drawPoint(LeftTopPointX, RightBottomPointY);
        painter->drawPoint(RightBottomPointX, RightBottomPointY);
    }
    painter->restore();
}

```

A.21 Текст файла jsondata.h

```

#ifndef JSONDATA_H
#define JSONDATA_H

#include<QJsonObject>
#include<QJsonParseError>

```

```

#include<QFile>
#include<QJsonArray>

#include "graphicitem.h"
#include "mdichild.h"
#include "itembycolor.h"
#include "itembypicture.h"

class JsonData
{
private:
    QJsonDocument doc;
    QJsonParseError docError;

public:
    JsonData();
    void inFromFile(const QString &fWay, MdiChild *Parent);
    void outToFile(const QString &fWay, MdiChild *Parent);
    void outHistoryToFile(const QString &fWay, MdiChild *Parent);
    void outInitHistoryFile(const QString &fWay, MdiChild *Parent);
    void InitializeSceneFromFile(MdiChild *Parent, QJsonArray &docArr);
    void InitScenFromFile(const QString &fWay, MdiChild *Parent, int
Version);
    void DeleteHighVersions(const QString &fWay, int Version);
};

#endif // JSONDATA_H

```

A.22 Текст файла jsondata.cpp

```

#include "jsondata.h"

JsonData::JsonData()
{

}

void JsonData::inFromFile(const QString &fWay, MdiChild *Parent)
{
    try{
        QFile file;
        file.setFileName(fWay);
        if (!file.open(QIODevice::ReadOnly|QFile::Text))
        {
            throw QString("ERROR OPEN TO READ FILE: " + fWay);
        }
        doc = QJsonDocument::fromJson(QByteArray(file.readAll()), &docError);
        file.close();
        if(docError.errorString().toInt() != QJsonParseError::NoError)
        {
            throw QString("ERROR JSON PARSE");
        }
        QJsonValue sizeScene = QJsonValue(doc.object().value("Size scene"));
        int width = sizeScene.toObject().value("width").toInt();
        int height = sizeScene.toObject().value("height").toInt();
        Parent->SetSceneParametrs(width,height);

        QJsonArray docArr =
QJsonValue(doc.object().value("figures")).toArray();
        InitializeSceneFromFile(Parent,docArr);

    }
    catch (QString error) {

```

```

        qDebug() << error;
    }

}

void JsonData::outToFile(const QString &fWay, MdiChild *Parent)
{
    try {
        if (Parent == NULL)
        {
            throw QString("ERROR PARENT == NULL");
        }
        QFile file;
        file.setFileName(fWay);
        if (!file.open(QIODevice::WriteOnly))
        {
            throw QString("ERROR NOT OPEN FILE "+fWay);
        }
        QJsonObject mainStruct;
        QJsonObject sizeScene;
        sizeScene.insert("width", Parent->scene->sceneRect().toRect().width());
        sizeScene.insert("height", Parent->scene->sceneRect().toRect().height());
        mainStruct.insert("Size scene", sizeScene);
        QJsonArray recordsArray;

        Parent->DeleteGrid();
        QList<QGraphicsItem*> itm = Parent->scene->items();
        for (int i = 0; i < itm.size(); i++)
        {
            GraphicItem *item = qgraphicsitem_cast<GraphicItem*>(itm[i]);
            GraphicItemInfo Iinformation = item->GetItemInfo();

            QJsonObject recordObject;
            recordObject.insert("Type",
                QJsonValue::fromVariant(Iinformation.Type));
            recordObject.insert("Value",
                QJsonValue::fromVariant(Iinformation.Value));
            recordObject.insert("Angle",
                QJsonValue::fromVariant(Iinformation.Angle));

            QJsonObject position;
            position.insert("x", Iinformation.X);
            position.insert("y", Iinformation.Y);
            recordObject.insert("Position", position);
            QJsonObject size;
            size.insert("width", Iinformation.Width);
            size.insert("height", Iinformation.Height);
            recordObject.insert("Size", size);
            recordsArray.push_back(recordObject);
        }
        Parent->DrawGrid();

        mainStruct.insert("figures", recordsArray);
        QJsonDocument doc(mainStruct);
        file.write("\n" + doc.toJson() + "\n");
        file.close();
    }
    catch (QString error) {
        qDebug() << error;
    }
}

```

```

void JsonData::outInitHistoryFile(const QString &fWay, MdiChild *Parent)
{
    if(Parent == NULL)
    {
        return;
    }

    QFile file;
    file.setFileName(fWay);
    if(file.open(QIODevice::WriteOnly))
    {
        QJsonObject mainStruct;
        QJsonArray scenesArray;
        for(int i = 0; i < 1; i++)
        {
            QJsonObject scene;
            QJsonArray recordsArray;

            Parent->DeleteGrid();
            QList<QGraphicsItem *> itm = Parent->scene->items();
            for(int i = 0; i < itm.size(); i++)
            {
                GraphicItem *item = qgraphicsitem_cast<GraphicItem*>(itm[i]);
                GraphicItemInfo Iinformation = item->GetItemInfo();

                QJsonObject recordObject;
                recordObject.insert("Type",
QJsonValue::fromVariant(Iinformation.Type));
                recordObject.insert("Value",
QJsonValue::fromVariant(Iinformation.Value));
                recordObject.insert("Angle",
QJsonValue::fromVariant(Iinformation.Angle));

                QJsonObject position;
                position.insert("x", Iinformation.X);
                position.insert("y", Iinformation.Y);
                recordObject.insert("Position", position);
                QJsonObject size;
                size.insert("width", Iinformation.Width);
                size.insert("height", Iinformation.Height);
                recordObject.insert("Size", size);
                recordsArray.push_back(recordObject);
            }
            Parent->DrawGrid();
            scene.insert("figures", recordsArray);
            scenesArray.push_back(scene);
        }
        mainStruct.insert("Scenes", scenesArray);

        QJsonDocument doc(mainStruct);
        file.write("\n" + doc.toJson()+"\n");
        file.close();
    }
    else
    {
        qDebug() << "ERROR";
    }
}

void JsonData::outHistoryToFile(const QString &fWay, MdiChild *Parent)
{
    QFile file;
    file.setFileName(fWay);

```

```

    if (file.open(QIODevice::ReadOnly|QFile::Text))
    {
        doc = QJsonDocument::fromJson(QByteArray(file.readAll()), &docError);
    }
    file.close();
    if (docError.errorString().toInt() == QJsonParseError::NoError)
    {
        QJsonArray scenesArray =
QJsonValue(doc.object().value("Scenes")).toArray();

        QJsonObject scene;
        QJsonArray recordsArray;
        Parent->DeleteGrid();
        QList<QGraphicsItem*> itm = Parent->scene->items();
        for (int i = 0; i < itm.size(); i++)
        {
            GraphicItem *item = qgraphicsitem_cast<GraphicItem*>(itm[i]);
            GraphicItemInfo Iinformation = item->GetItemInfo();

            QJsonObject recordObject;
            recordObject.insert("Type",
QJsonValue::fromVariant(Iinformation.Type));
            recordObject.insert("Value",
QJsonValue::fromVariant(Iinformation.Value));
            recordObject.insert("Angle",
QJsonValue::fromVariant(Iinformation.Angle));

            QJsonObject position;
            position.insert("x", Iinformation.X);
            position.insert("y", Iinformation.Y);
            recordObject.insert("Position", position);
            QJsonObject size;
            size.insert("width", Iinformation.Width);
            size.insert("height", Iinformation.Height);
            recordObject.insert("Size", size);
            recordsArray.push_back(recordObject);
        }
        Parent->DrawGrid();
        scene.insert("figures", recordsArray);
        scenesArray.push_back(scene);

        QJsonObject mainStruct;
        mainStruct.insert("Scenes", scenesArray);

        QFile file;
        file.setFileName(fWay);
        if (file.open(QIODevice::WriteOnly))
        {
            QJsonDocument doc(mainStruct);
            file.write("\n" + doc.toJson() + "\n");
            file.close();
        }
        else
        {
            qDebug() << "ERROR";
        }
    }
    else
    {
        qDebug() << "ERROR";
    }
}

void JsonData::InitializeSceneFromFile(MdiChild *Parent, QJsonArray &docArr)

```

```

{
    for(int i = 0; i<docArr.count();i++)
    {
        GraphicItemInfo Information;
        Information.Type = docArr.at(i).toObject().value("Type").toString();
        Information.Value =
docArr.at(i).toObject().value("Value").toString();
        QJsonValue posVal =
QJsonValue(docArr.at(i).toObject().value("Position"));
        Information.X = posVal.toObject().value("x").toDouble();
        Information.Y = posVal.toObject().value("y").toDouble();
        QJsonValue sizeVal =
QJsonValue(docArr.at(i).toObject().value("Size"));
        Information.Width = sizeVal.toObject().value("width").toDouble();
        Information.Height = sizeVal.toObject().value("height").toDouble();
        Information.Angle =
docArr.at(i).toObject().value("Angle").toDouble();

        if(Information.Type == "Picture")
        {
            ItemByPicture *item = new ItemByPicture(Information.Width,
Information.Height, Information.Value, Parent);
            item->setPos(Information.X, Information.Y);
            item->setRotation(Information.Angle);
            Parent->scene->addItem(item);
        }
        else
        {
            ItemByColor *item = new ItemByColor(Information.Width,
Information.Height, QColor(Information.Value), Parent);
            item->setPos(Information.X, Information.Y);
            item->setRotation(Information.Angle);
            Parent->scene->addItem(item);
        }
    }
}

void JsonData::InitScenFromFile(const QString &fWay, MdiChild *Parent, int
Version)
{
    QFile file;
    file.setFileName(fWay);
    if(file.open(QIODevice::ReadOnly|QFile::Text))
    {
        doc = QJsonDocument::fromJson(QByteArray(file.readAll()),&docError);
    }
    file.close();
    if(docError.errorString().toInt() == QJsonParseError::NoError)
    {
        QJsonArray sceneArr =
QJsonValue(doc.object().value("Scenes")).toArray();
        QJsonArray docArr = QJsonValue(
sceneArr.at(Version).toObject().value("figures")).toArray();
        Parent->LastActive = NULL;
        Parent->DeleteGrid();
        Parent->scene->clear();
        Parent->DrawGrid();
        InitializeSceneFromFile(Parent,docArr);
    }
}

void JsonData::DeleteHighVersions(const QString &fWay, int Version)
{
    QFile file;

```

```

file.setFileName(fWay);
if (file.open(QIODevice::ReadWrite|QFile::Text))
{
    doc = QJsonDocument::fromJson(QByteArray(file.readAll()), &docError);
}
file.close();
if (docError.errorString().toInt() == QJsonParseError::NoError)
{
    QJsonArray sceneArr =
QJsonValue(doc.object().value("Scenes")).toArray();
    for (int j = 0; j < sceneArr.count(); j++)
    {
        while (Version+1 < sceneArr.size()) {
            sceneArr.removeLast();
        }
    }
    QJsonObject mainStruct;
    mainStruct.insert("Scenes", sceneArr);

    if (file.open(QIODevice::ReadWrite|QFile::Text))
    {
        QJsonDocument doc(mainStruct);
        file.resize(0);
        file.write("\n" + doc.toJson() + "\n");
    }
    file.close();
}
}

```

A.23 Текст файла mainwindow.h

```

#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QSettings>
#include <QTranslator>
#include <QtWidgets>
#include "ui_mainwindow.h"
#include "jsondata.h"
#include "aboutwindow.h"
#include "mdichild.h"
#include "graphicitem.h"
#include "itembycolor.h"
#include "itembypicture.h"
#include "fileopen.h"
#include "itemeditwindow.h"
#include "formcreate.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    MdiChild *activeMdiChild() const;
    void loadProject(QWidget *widget);
    DataBase database;
    int uniqueProjectID = 1;

```

```

private slots:
    void on_actionNew_triggered();
    void on_actionOpen_triggered();
    void on_actionSave_triggered();
    void on_actionSaveAs_triggered();
    void on_actionClose_triggered();
    void on_actionHelp_triggered();
    void on_actionAbout_triggered();
    void on_listViewBuildTools_clicked(const QModelIndex &index);
    void on_listViewFurnitureTools_clicked(const QModelIndex &index);
    void on_listViewDecorTools_clicked(const QModelIndex &index);
    void on_actionUk_triggered();
    void on_actionEn_triggered();
    void on_actionRu_triggered();
    void on_actionUndo_triggered();
    void on_actionRedo_triggered();
    void on_actionDelete_triggered();
    void on_actionClear_triggered();
    void on_actionZoomIn_triggered();
    void on_actionZoomOut_triggered();
    void on_actionSetItemSize_triggered();

private:
    void saveSettings();
    QString loadSettings();
    Ui::MainWindow *ui;

    QStandardItemModel *modelBuildTools;
    QSortFilterProxyModel *proxyModelBuildTools;
    QStandardItemModel *modelFurnitureTools;
    QSortFilterProxyModel *proxyModelFurnitureTools;
    QStandardItemModel *modelDecorTools;
    QSortFilterProxyModel *proxyModelDecorTools;

    int FilesCount = 0;
    void setupOtherUi();
    void setupBuildTools();
    void setupFurnitureTools();
    void setupDecorTools();

    QSettings* settings;
    QTranslator* translator;
    void changeTranslator(QString postfix);
    void changeEvent(QEvent *event);
};

#endif // MAINWINDOW_H

```

A.24 Текст файла mainwindow.cpp

```

#include "mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    setupOtherUi();
    database.connectToDataBase();
    settings = new QSettings("settings.ini", QSettings::IniFormat, this);
    changeTranslator(loadSettings());
}

```



```

MainWindow::~MainWindow()
{
    saveSettings();
    delete settings;
    delete ui;
}

void MainWindow::setupOtherUi()
{
    setupBuildTools();
    setupFurnitureTools();
    setupDecorTools();
}

void MainWindow::saveSettings()
{
    if(ui->actionRu->isChecked())
    {
        settings->setValue("language", "ru");
    }
    else if(ui->actionEn->isChecked())
    {
        settings->setValue("language", "en");
    }
    else
    {
        settings->setValue("language", "uk");
    }
    settings->setValue("geometry", geometry());
}

QString MainWindow::loadSettings()
{
    setGeometry(settings->value("geometry", QRect(50, 50, 700, 460)).toRect());
    return settings->value("language", "uk").toString();
}

void MainWindow::setupBuildTools()
{
    modelBuildTools = new QStandardItemModel(this);
    proxyModelBuildTools = new QSortFilterProxyModel(this);

    modelBuildTools->insertColumn(0);
    int numRows = 4;
    modelBuildTools->insertRows(0, numRows);

    modelBuildTools->setData(modelBuildTools->index(0, 0),
    QPixmap(":/list/build/wall"), Qt::DecorationRole);
    modelBuildTools->setData(modelBuildTools->index(0, 0), "Стіна",
    Qt::EditRole);
    modelBuildTools->setData(modelBuildTools->index(1, 0),
    QPixmap(":/list/build/window"), Qt::DecorationRole);
    modelBuildTools->setData(modelBuildTools->index(1, 0), "Вікно",
    Qt::EditRole);
    modelBuildTools->setData(modelBuildTools->index(2, 0),
    QPixmap(":/list/build/door"), Qt::DecorationRole);
    modelBuildTools->setData(modelBuildTools->index(2, 0), "Двері",
    Qt::EditRole);
    modelBuildTools->setData(modelBuildTools->index(3, 0),
    QPixmap(":/list/build/door-garage"), Qt::DecorationRole);
    modelBuildTools->setData(modelBuildTools->index(3, 0), "Гаражні\ndвері",
    Qt::EditRole);

    proxyModelBuildTools->setSourceModel(modelBuildTools);
    proxyModelBuildTools->setFilterKeyColumn(0);
}

```

```

        ui->listViewBuildTools->setModel(modelBuildTools);
    }

void MainWindow::setupFurnitureTools()
{
    modelFurnitureTools = new QStandardItemModel(this);
    proxyModelFurnitureTools = new QSortFilterProxyModel(this);

    modelFurnitureTools->insertColumn(0);
    int numRows = 13;
    modelFurnitureTools->insertRows(0, numRows);

    modelFurnitureTools->setData(modelFurnitureTools->index(0, 0),
    QPixmap(":/list/furniture/sofa"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(0, 0), "Диван",
    Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(1, 0),
    QPixmap(":/list/furniture/chair"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(1, 0), "Стілець",
    Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(2, 0),
    QPixmap(":/list/furniture/bed-single"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(2, 0), "Ліжко",
    Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(3, 0),
    QPixmap(":/list/furniture/bed-double"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(3, 0),
    "Двомісне\нліжко", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(4, 0),
    QPixmap(":/list/furniture/bedside"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(4, 0),
    "Тумбочка", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(5, 0),
    QPixmap(":/list/furniture/table-dining"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(5, 0),
    "Кухонний\нстіл", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(6, 0),
    QPixmap(":/list/furniture/table-rect"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(6, 0),
    "Квадратний\нстіл", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(7, 0),
    QPixmap(":/list/furniture/table-work"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(7, 0),
    "Робочий\нстіл", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(8, 0),
    QPixmap(":/list/furniture/table-glass"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(8, 0),
    "Скляний\нстіл", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(9, 0),
    QPixmap(":/list/furniture/table-wood"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(9, 0),
    "Диванний\нстолик", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(10, 0),
    QPixmap(":/list/furniture/oven"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(10, 0),
    "Духовка", Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(11, 0),
    QPixmap(":/list/furniture/bath"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(11, 0), "Ванна",
    Qt::EditRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(12, 0),
    QPixmap(":/list/furniture/toilet"), Qt::DecorationRole);
    modelFurnitureTools->setData(modelFurnitureTools->index(12, 0), "Туалет",
    Qt::EditRole);
}

```

```

        proxyModelFurnitureTools->setSourceModel(modelFurnitureTools);
        proxyModelFurnitureTools->setFilterKeyColumn(0);
        ui->listViewFurnitureTools->setModel(modelFurnitureTools);
    }

void MainWindow::setupDecorTools()
{
    modelDecorTools = new QStandardItemModel(this);
    proxyModelDecorTools = new QSortFilterProxyModel(this);

    modelDecorTools->insertColumn(0);
    int numRows = 2;
    modelDecorTools->insertRows(0, numRows);

    modelDecorTools->setData(modelDecorTools->index(0, 0),
        QPixmap(":/list/decor/plant"), Qt::DecorationRole);
    modelDecorTools->setData(modelDecorTools->index(0, 0), "Рослина",
        Qt::EditRole);
    modelDecorTools->setData(modelDecorTools->index(1, 0),
        QPixmap(":/list/decor/car"), Qt::DecorationRole);
    modelDecorTools->setData(modelDecorTools->index(1, 0), "Автомобіль",
        Qt::EditRole);

    proxyModelDecorTools->setSourceModel(modelDecorTools);
    proxyModelDecorTools->setFilterKeyColumn(0);
    ui->listViewDecorTools->setModel(modelDecorTools);
}

MdiChild *MainWindow::activeMdiChild() const
{
    if (QMdiSubWindow *activeSubWindow = ui->mdiArea->activeSubWindow())
        return qobject_cast<MdiChild *>(activeSubWindow->widget());
    return 0;
}

void MainWindow::on_actionNew_triggered()
{
    FormCreate *Window = new FormCreate(this);
    Window->show();
}

void MainWindow::on_actionOpen_triggered()
{
    FileOpen OpenFile(this);
    OpenFile.OpenFile();
}

void MainWindow::on_actionSave_triggered()
{
    if(activeMdiChild() != NULL)
    {
        activeMdiChild()->SaveFileInfo.Save(this);
    }
}

void MainWindow::on_actionSaveAs_triggered()
{
    if(activeMdiChild() != NULL)
    {
        activeMdiChild()->SaveFileInfo.SaveAs(this);
    }
}

```

```

void MainWindow::on_actionClose_triggered()
{
    close();
}

void MainWindow::on_actionHelp_triggered()
{
}

void MainWindow::on_actionAbout_triggered()
{
    AboutWindow *about = new AboutWindow();
    about->show();
}

void MainWindow::loadProject(QWidget *widget)
{
    auto window = ui->mdiArea->addSubWindow(widget);
    if (ui->mdiArea->subWindowList().size() == 1) window->showMaximized();
    else window->show();
}

void MainWindow::on_listViewBuildTools_clicked(const QModelIndex &index)
{
    MdiChild *activeChild = activeMdiChild();
    if (!activeChild)
        return;

    int i = index.row();

    switch (i) {
        case 0:
            activeChild->addItem(Qt::darkGray, 500, 30);
            break;
        case 1:
            activeChild->addItem(QColor("#87CEFA"), 100, 30);
            break;
        case 2:
            activeChild->addItem(QColor("#CB6843"), 80, 30);
            break;
        case 3:
            activeChild->addItem(QColor("#202020"), 250, 30);
            break;
    }
}

void MainWindow::on_listViewFurnitureTools_clicked(const QModelIndex &index)
{
    MdiChild *activeChild = activeMdiChild();
    if (!activeChild)
        return;

    int i = index.row();

    switch (i) {
        case 0:
            activeChild->addItem(QString(":scene/sofa"), 200, 70);
            break;
        case 1:
            activeChild->addItem(QString(":scene/chair"), 50, 60);
            break;
        case 2:
            activeChild->addItem(QString(":scene/bed-single"), 90, 200);

```

```

        break;
    case 3:
        activeChild->addItem(QString(":scene/bed-double"), 150, 200);
        break;
    case 4:
        activeChild->addItem(QString(":scene/bedside"), 50, 50);
        break;
    case 5:
        activeChild->addItem(QString(":scene/table-dining"), 180, 180);
        break;
    case 6:
        activeChild->addItem(QString(":scene/table-rect"), 140, 140);
        break;
    case 7:
        activeChild->addItem(QString(":scene/table-work"), 150, 90);
        break;
    case 8:
        activeChild->addItem(QString(":scene/table-glass"), 150, 90);
        break;
    case 9:
        activeChild->addItem(QString(":scene/table-wood"), 150, 70);
        break;
    case 10:
        activeChild->addItem(QString(":scene/oven"), 70, 70);
        break;
    case 11:
        activeChild->addItem(QString(":scene/bath"), 190, 80);
        break;
    case 12:
        activeChild->addItem(QString(":scene/toilet"), 50, 80);
        break;
    }
}

void MainWindow::on_listViewDecorTools_clicked(const QModelIndex &index)
{
    MdiChild *activeChild = activeMdiChild();
    if (!activeChild)
        return;

    int i = index.row();

    switch (i) {
        case 0:
            activeChild->addItem(QString(":/scene/plant"), 50, 50);
            break;
        case 1:
            activeChild->addItem(QString(":/scene/car"), 150, 350);
            break;
    }
}

void MainWindow::on_actionUk_triggered()
{
    changeTranslator("uk");
}

void MainWindow::on_actionEn_triggered()
{
    changeTranslator("en");
}

```

```

void MainWindow::on_actionRu_triggered()
{
    changeTranslator("ru");
}

void MainWindow::changeTranslator(QString postfix)
{
    if(postfix=="uk")
    {
        ui->actionEn->setChecked(false);
        ui->actionRu->setChecked(false);
        ui->actionUk->setChecked(true);
    }
    else if(postfix=="en")
    {
        ui->actionEn->setChecked(true);
        ui->actionRu->setChecked(false);
        ui->actionUk->setChecked(false);
    }
    else
    {
        ui->actionEn->setChecked(false);
        ui->actionRu->setChecked(true);
        ui->actionUk->setChecked(false);
    }
    QApplication::removeTranslator(translator);
    translator = new QTranslator(this);
    translator->load("Core_Planner_" + postfix);
    QApplication::installTranslator(translator);
}

void MainWindow::changeEvent(QEvent *event)
{
    if (event->type() == QEvent::LanguageChange)
    {
        ui->retranslateUi(this);
        modelBuildTools->setData(modelBuildTools->index(0, 0), tr("Стіна"),
Qt::EditRole);
        modelBuildTools->setData(modelBuildTools->index(1, 0), tr("Вікно"),
Qt::EditRole);
        modelBuildTools->setData(modelBuildTools->index(2, 0), tr("Двері"),
Qt::EditRole);
        modelBuildTools->setData(modelBuildTools->index(3, 0),
tr("Гаражні\ндвері"), Qt::EditRole);

        modelFurnitureTools->setData(modelFurnitureTools->index(0, 0),
tr("Диван"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(1, 0),
tr("Стілець"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(2, 0),
tr("Ліжко"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(3, 0),
tr("Двомісне\нліжко"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(4, 0),
tr("Тумбочка"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(5, 0),
tr("Кухонний\нстіл"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(6, 0),
tr("Квадратний\нстіл"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(7, 0),
tr("Робочий\нстіл"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(8, 0),
tr("Скляний\нстіл"), Qt::EditRole);
    }
}

```

```

        modelFurnitureTools->setData(modelFurnitureTools->index(9, 0),
tr("Диванний\нстолик"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(10, 0),
tr("Духовка"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(11, 0),
tr("Ванна"), Qt::EditRole);
        modelFurnitureTools->setData(modelFurnitureTools->index(12, 0),
tr("Туалет"), Qt::EditRole);

        modelDecorTools->setData(modelDecorTools->index(0, 0), tr("Рослина"),
Qt::EditRole);
        modelDecorTools->setData(modelDecorTools->index(1, 0),
tr("Автомобіль"), Qt::EditRole);
    }
    else
    {
        QMainWindow::changeEvent(event);
    }
}

void MainWindow::on_actionUndo_triggered()
{
    if(activeMdiChild() != NULL)
    {
        if(activeMdiChild()->VersionOfScene > 0)
        {
            activeMdiChild()->AfterUndo = true;
            activeMdiChild()->VersionOfScene--;
            activeMdiChild()->SaveFileInfo.LoadFromHistory(activeMdiChild()-
>VersionOfScene);
        }
    }
}

void MainWindow::on_actionRedo_triggered()
{
    if(activeMdiChild() != NULL)
    {
        if(activeMdiChild()->VersionOfScene < activeMdiChild()-
>MaximumVersionOfScene)
        {
            activeMdiChild()->VersionOfScene++;
            activeMdiChild()->SaveFileInfo.LoadFromHistory(activeMdiChild()-
>VersionOfScene);
        }
    }
}

void MainWindow::on_actionDelete_triggered()
{
    if(activeMdiChild() != NULL && activeMdiChild()->LastActive != NULL)
    {
        delete activeMdiChild()->LastActive;
        activeMdiChild()->LastActive = NULL;
        if(activeMdiChild()->AfterUndo)
        {
            activeMdiChild()-
>SaveFileInfo.DeleteHighVersions(activeMdiChild()->VersionOfScene);
            activeMdiChild()->AfterUndo = false;
            activeMdiChild()->MaximumVersionOfScene = activeMdiChild()-
>VersionOfScene;
        }
        activeMdiChild()->VersionOfScene++;
    }
}

```

```

        activeMdiChild()->MaximumVersionOfScene++;
        activeMdiChild()->SaveFileInfo.SaveToHistory();
        activeMdiChild()->SaveFileInfo.LoadFromHistory(activeMdiChild()-
>VersionOfScene);
    }

}

void MainWindow::on_actionClear_triggered()
{
    if(activeMdiChild() != NULL)
    {
        activeMdiChild()->DeleteGrid();
        activeMdiChild()->scene->clear();
        activeMdiChild()->DrawGrid();
        if(activeMdiChild()->AfterUndo)
        {
            activeMdiChild()-
>SaveFileInfo.DeleteHighVersions(activeMdiChild()->VersionOfScene);
            activeMdiChild()->AfterUndo = false;
            activeMdiChild()->MaximumVersionOfScene = activeMdiChild()-
>VersionOfScene;
        }
        activeMdiChild()->VersionOfScene++;
        activeMdiChild()->MaximumVersionOfScene++;
        activeMdiChild()->SaveFileInfo.SaveToHistory();
        activeMdiChild()->SaveFileInfo.LoadFromHistory(activeMdiChild()-
>VersionOfScene);
    }
}

void MainWindow::on_actionZoomIn_triggered()
{
    if(activeMdiChild() != NULL) activeMdiChild()->setScaleToView(1.5, 1.5);
}

void MainWindow::on_actionZoomOut_triggered()
{
    if(activeMdiChild() != NULL) activeMdiChild()->setScaleToView(0.75,
0.75);
}

void MainWindow::on_actionSetItemSize_triggered()
{
    if(activeMdiChild() != NULL && activeMdiChild()->LastActive != NULL)
    {
        ItemEditWindow *window = new ItemEditWindow(this, activeMdiChild());
        window->show();
    }
}

```

A.25 Текст файла mdichild.h

```

#ifndef MDICHILD_H
#define MDICHILD_H

#include <QtWidgets>
#include "ui_mdichild.h"
#include "graphicitem.h"
#include "itembypicture.h"
#include "itembycolor.h"
#include "filesave.h"

```



```

namespace Ui {
class MdiChild;
}

class FileSave;

class MdiChild : public QWidget
{
    Q_OBJECT

public:
    explicit MdiChild(MainWindow *Main, QWidget *parent = 0);
    ~MdiChild();
    GraphicItem *LastActive = 0;
    QGraphicsScene *scene;
    void addItem(QString Name, double setWidth, double setHeight);
    void addItem(QColor ItemColor, double setWidth, double setHeight);
    void UpdateScene();
    void SaveAsImage(QString NameOfFile);
    void setScaleToView(double X, double Y);
    void DrawGrid();
    void DeleteGrid();
    void SetSceneParametrs(int setWidth, int setHeight);
    FileSave SaveFileInfo;
    int VersionOfScene;
    int MaximumVersionOfScene;
    bool AfterUndo;

private:
    Ui::MdiChild *ui;
    MainWindow *MainWind;
    int Width, Height;
    QVector<QGraphicsLineItem*> HorizontalLines;
    QVector<QGraphicsLineItem*> VerticalLines;
    void DrawHorizontalLines();
    void DrawVerticalLines();
};

#endif // MDICHILD_H

```

A.26 Текст файла mdichild.cpp

```

#include "mdichild.h"

MdiChild::MdiChild(MainWindow *Main, QWidget *parent) :
    QWidget(parent),
    ui(new Ui::MdiChild)
{
    ui->setupUi(this);
    MainWind = Main;
    Width = 2000; Height = 1000;
    scene = new QGraphicsScene(this);
    scene->setItemIndexMethod(QGraphicsScene::NoIndex);
    scene->setSceneRect(0,0,Width,Height);
    ui->graphicsView->setScene(scene);

    SaveFileInfo.ParenWithScene = this;
    SaveFileInfo.FileName = this->windowTitle();
    DrawGrid();
    SaveFileInfo.InitHistory();
    VersionOfScene = 0;
    MaximumVersionOfScene = 0;
    AfterUndo = false;
}

```

```

}

void MdiChild::SetSceneParameters(int setWidth, int setHeight)
{
    DeleteGrid();
    Width = setWidth;
    Height = setHeight;
    scene->setSceneRect(0,0,Width,Height);
    DrawGrid();
}

MdiChild::~MdiChild()
{
    QString Message = tr("Зберегти проект ");
    Message += SaveFileInfo.FileName;
    Message += tr(" перед закриттям?");
    if(QMessageBox::Yes == QMessageBox::question(this, tr("Увага"), Message))
    {
        SaveFileInfo.Save(MainWind);
    }
    delete ui;
}

void MdiChild::addItem(QString Name, double setWidth, double setHeight)
{
    if(AfterUndo)
    {
        SaveFileInfo.DeleteHighVersions(VersionOfScene);
        AfterUndo = false;
        MaximumVersionOfScene = VersionOfScene;
    }

    ItemByPicture *item = new ItemByPicture(setWidth, setHeight, Name, this);
    item->setPos(scene->sceneRect().toRect().width()/2, scene-
>sceneRect().toRect().height()/2);
    scene->addItem(item);

    VersionOfScene++;
    MaximumVersionOfScene++;
    SaveFileInfo.SaveToHistory();
    SaveFileInfo.LoadFromHistory(VersionOfScene);
}

void MdiChild::addItem(QColor ItemColor, double setWidth, double setHeight)
{
    if(AfterUndo)
    {
        SaveFileInfo.DeleteHighVersions(VersionOfScene);
        AfterUndo = false;
        MaximumVersionOfScene = VersionOfScene;
    }

    ItemByColor *item = new ItemByColor(setWidth, setHeight, ItemColor,
this);
    item->setPos(scene->sceneRect().toRect().width()/2, scene-
>sceneRect().toRect().height()/2);
    scene->addItem(item);

    VersionOfScene++;
    MaximumVersionOfScene++;
    SaveFileInfo.SaveToHistory();
    SaveFileInfo.LoadFromHistory(VersionOfScene);
}

```

```

void MdiChild::UpdateScene()
{
    scene->update();
}

void MdiChild::DrawHorizontalLines()
{
    for(int i = 0; i <= Height; i+=10)
    {
        QGraphicsLineItem *HLine = new QGraphicsLineItem();
        HLine->setLine(0,i,Width,i);
        if(i%100 == 0) HLine->setPen(QPen(Qt::darkGray));else HLine-
>setPen(QPen(Qt::lightGray));
        HorizontalLines.push_back(HLine);
        scene->addItem(HLine);
    }
}

void MdiChild::DrawVerticalLines()
{
    for(int i = 0; i <= Width; i+=10)
    {
        QGraphicsLineItem *VLine = new QGraphicsLineItem();
        VLine->setLine(i,0,i,Height);
        if(i%100 == 0) VLine->setPen(QPen(Qt::darkGray));else VLine-
>setPen(QPen(Qt::lightGray));
        VerticalLines.push_back(VLine);
        scene->addItem(VLine);
    }
}

void MdiChild::DrawGrid()
{
    if(VerticalLines.size() == 0) DrawVerticalLines();
    if(HorizontalLines.size() == 0) DrawHorizontalLines();
}

void MdiChild::DeleteGrid()
{
    while(HorizontalLines.size() > 0)
    {
        delete HorizontalLines.back();
        HorizontalLines.pop_back();
    }
    while(VerticalLines.size() > 0)
    {
        delete VerticalLines.back();
        VerticalLines.pop_back();
    }
}

void MdiChild::SaveAsImage(QString NameOfFile)
{
    DeleteGrid();

    QImage image(int(scene->sceneRect().toRect().width()), int(scene-
>sceneRect().toRect().height()), QImage::Format_ARGB32);
    QPainter painter(&image);
    painter.setRenderHint(QPainter::Antialiasing);
    scene->render(&painter);
    image.save(NameOfFile);

    DrawGrid();
}

```

```
void MdiChild::setScaleToView(double X, double Y)
{
    ui->graphicsView->scale(X, Y);
}
```

A.27 Текст файла points.h

```
#ifndef POINTS_H
#define POINTS_H

#include <QPointF>

class points
{
private:
    QPointF TopMinimumCenter;
    QPointF LeftMinimumCenter;
    QPointF RightMinimumCenter;
    QPointF BottomMinimumCenter;

public:
    points();
    void SetTopMinimumCenter(QPointF point) {TopMinimumCenter = point;}
    void SetLeftMinimumCenter(QPointF point) {LeftMinimumCenter = point;}
    void SetRightMinimumCenter(QPointF point) {RightMinimumCenter = point;}
    void SetBottomMinimumCenter(QPointF point) {BottomMinimumCenter = point;}
    QPointF GetTopMinimumCenter() {return TopMinimumCenter;}
    QPointF GetLeftMinimumCenter() {return LeftMinimumCenter;}
    QPointF GetRightMinimumCenter() {return RightMinimumCenter;}
    QPointF GetBottomMinimumCenter() {return BottomMinimumCenter;}
};

#endif // POINTS_H
```

ДОДАТОК Б

Інтерфейс програми

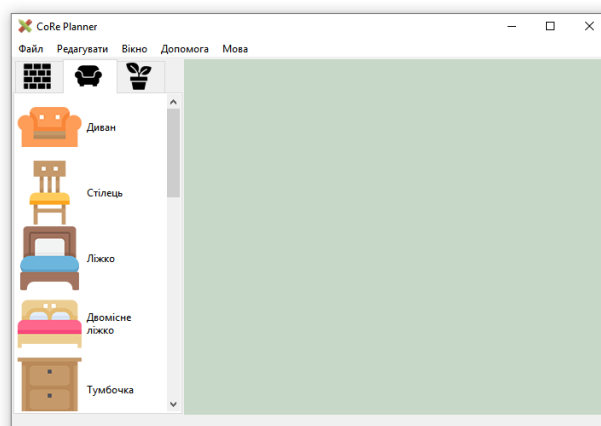


Рисунок Б.1 – Головне вікно програми

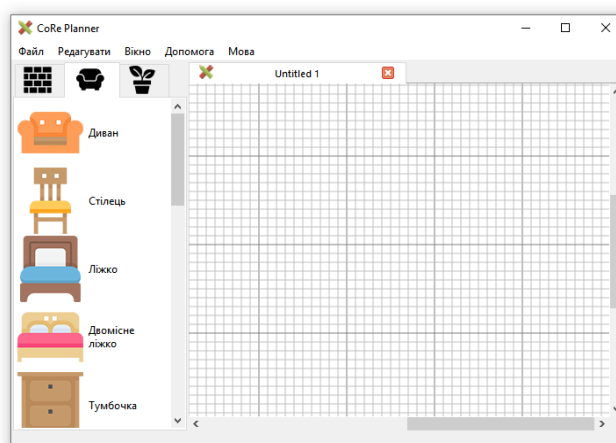


Рисунок Б.2 – Головне вікно програми з відкритим пустим проектом

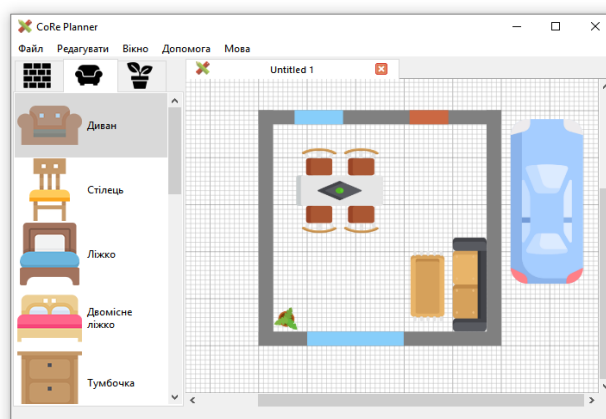


Рисунок Б.3 – Головне вікно програми з проектом у роботі

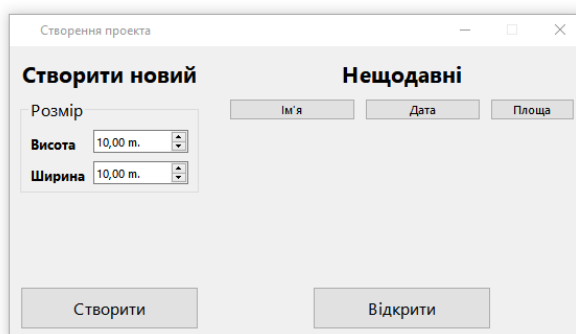


Рисунок Б.4 – Вікно створення нового проекту або відкриття нещодавнього

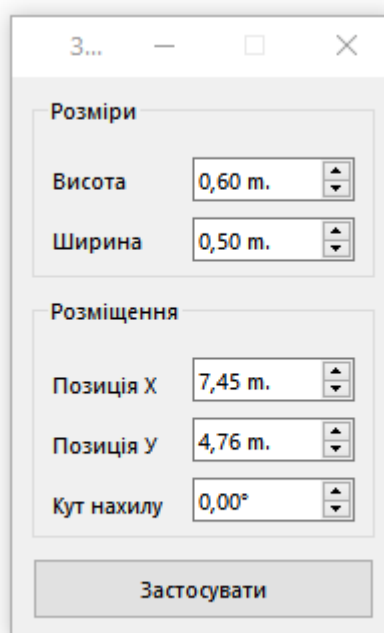


Рисунок Б.5 – Вікно редагування властивостей об'єктів

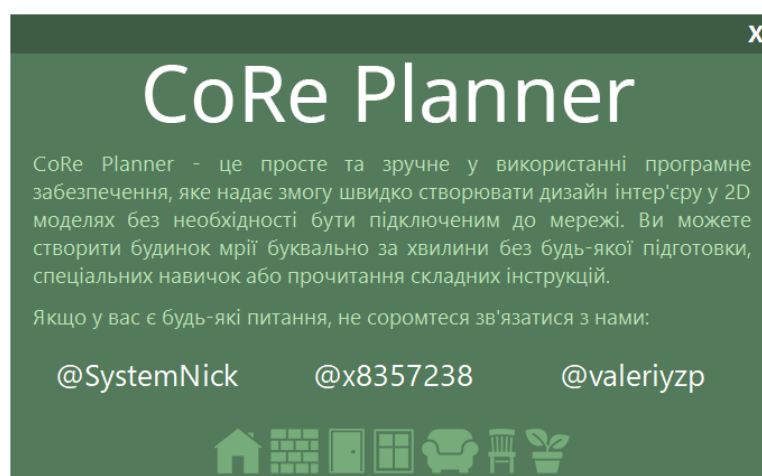


Рисунок Б.6 – Вікно з інформацією про програму та розробників

ДОДАТОК В

Слайди презентації



Рисунок В.1 – Тема курсового проекту

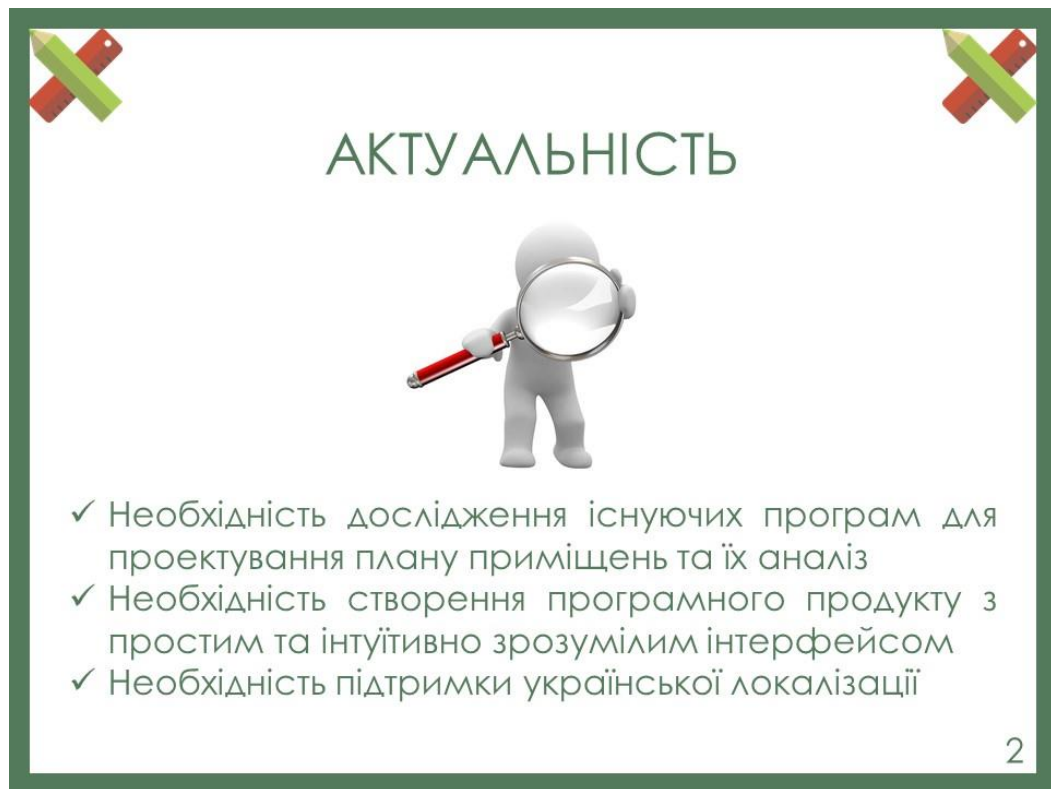


Рисунок В.2 – Актуальність



Рисунок В.3 – Мета та завдання курсової роботи

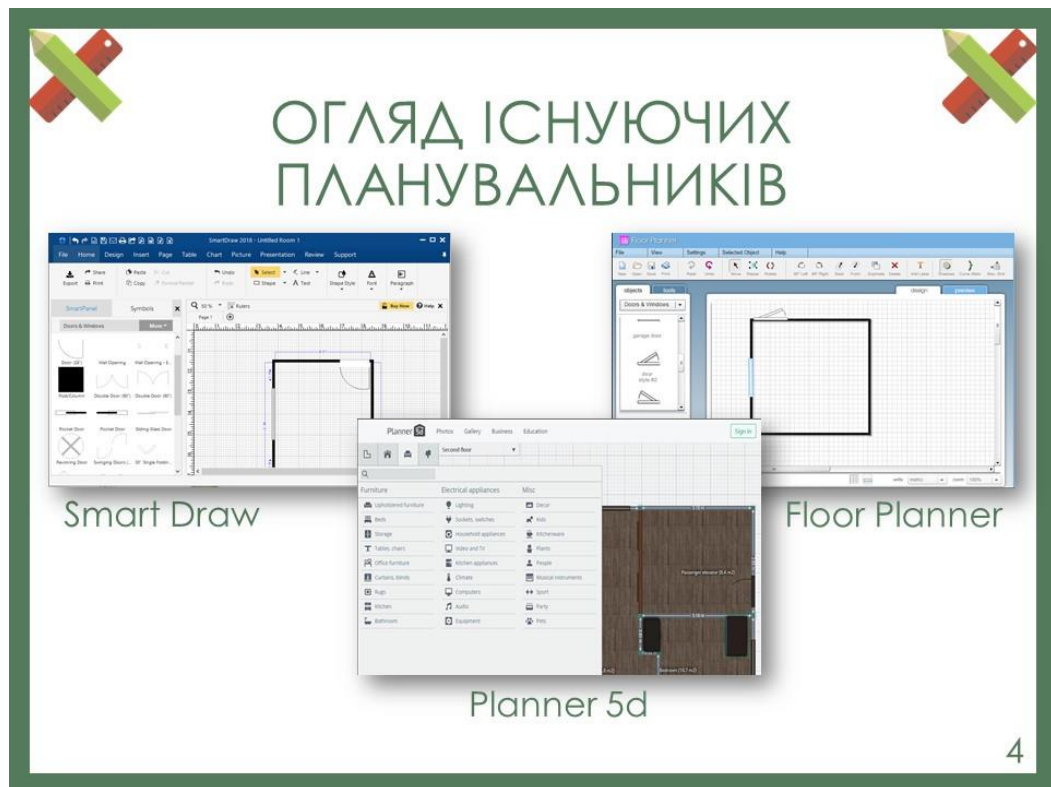
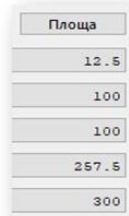


Рисунок В.4 – Огляд існуючих аналогів



Рисунок В.5 – Розроблені алгоритми

СОРТУВАННЯ ДАНИХ БУЛЬБАШКОВИМ МЕТОДОМ



Площа
12.5
100
100
257.5
300

Застосування у програмі

START	6	2	9	11	9	3	7	12
	2	6	9	9	3	7	11	12
	2	6	9	3	7	9	11	12
	2	9	3	7	9	9	11	12
END	2	3	7	9	9	9	11	12

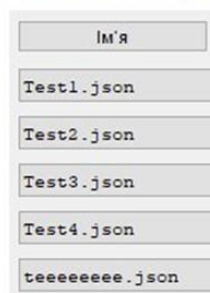
Приклад роботи алгоритму

Складність алгоритму у найгіршому у випадку
рівна $O(n^2)$, де n — кількість елементів для
сортування. Даний алгоритм має низьку
ефективність у випадках, коли N є досить великим.

6

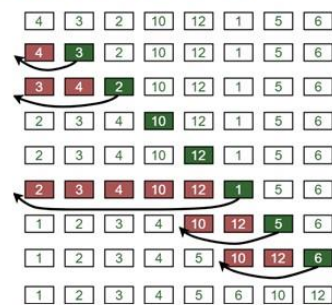
Рисунок В.6 – Сортування бульбашкою

СОРТУВАННЯ ДАНИХ МЕТОДОМ ВСТАВОК



Ім'я
Test1.json
Test2.json
Test3.json
Test4.json
teeeeeeeee.json

Застосування у програмі



4	3	2	10	12	1	5	6
4	3	2	10	12	1	5	6
3	4	2	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
2	3	4	10	12	1	5	6
1	2	3	4	10	12	5	6
1	2	3	4	5	10	12	6
1	2	3	4	5	6	10	12

Приклад роботи алгоритму

Алгоритм сортування базується на основі порівнянь.

Простий у реалізації, ефективний на маленьких
масивах, на практиці ефективніший за більшість
інших квадратичних алгоритмів, є більш стабільним.

7

Рисунок В.7 – Сортування вставками

СОРТУВАННЯ ДАНИХ МЕТОДОМ ШЕЛЛА

Дата
2018.12.26 12:37
2018.12.26 07:39
2018.12.26 04:41
2018.12.26 04:40
2018.12.26 04:39

Застосування у програмі

START	44	55	12	42	94	18	6	67
	44	18	6	42	94	55	12	67
	2	18	12	42	44	55	94	67
END	2	3	7	9	9	9	11	12

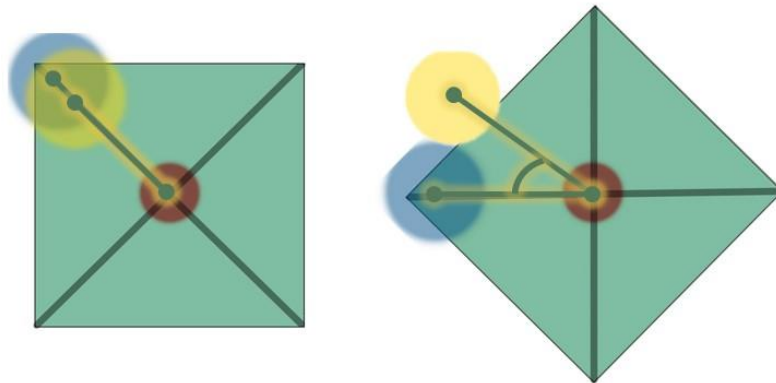
Приклад роботи алгоритму

Алгоритм є вдосконалим варіантом сортування вставками. Відбувається порівняння елементів, що стоять не тільки поруч, але і на певній відстані один від одного - алгоритм сортування вставками з попередніми «грубими» проходами.

8

Рисунок В.8 – Сортування методом Шелла

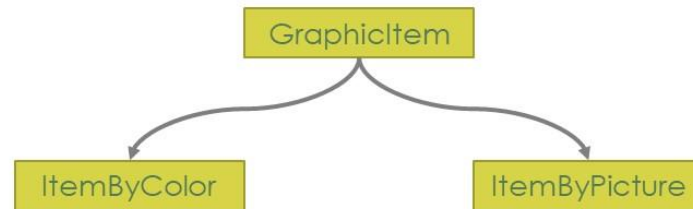
РОЗРАХУНОК КУТА ЗА ТРЬОМА ТОЧКАМИ



10

Рисунок В.9 – Розрахування кута за трьома точками

РОБОТА З КЛАСАМИ



Ієрархія класів призначених для обробки графічних об'єктів на сцені.

10

Рисунок В.10 – Робота з класами

РОБОТА З ДАНИМИ JSON

```

1  {
2  "Size scene": {
3    "height": 1000,
4    "width": 500
5  },
6  "figures": [
7    {
8      "Angle": 90,
9      "Position": {
10       "x": 100,
11       "y": 500
12     },
13     "Size": {
14       "height": 90,
15       "width": 150
16     },
17     "Type": "Picture",
18     "Value": ":scene/table-glass"
19   },
20   {
21     "Angle": 0,
22     "Position": {
23       "x": 100,
24       "y": 500
25     },
26     "Size": {
27       "height": 90,
28       "width": 150
29     },
30     "Type": "Picture",
31     "Value": ":scene/table-glass"
32   }
33 ]
  
```

Файл збереження проекту

```

1  {
2  "Scenes": [
3    {
4      "figures": [
5        {
6          "Angle": 0,
7          "Position": {
8            "x": 500,
9            "y": 500
10         },
11         "Size": {
12           "height": 200,
13           "width": 90
14         },
15         "Type": "Picture",
16         "Value": ":scene/table-glass"
17       },
18       "Angle": 0,
19       "Position": {
20         "x": 500,
21         "y": 500
22       },
23       "Size": {
24         "height": 200,
25         "width": 90
26       },
27       "Type": "Picture",
28       "Value": ":scene/table-glass"
29     }
30 ]
  
```

Файл історії змін проекту

11

Рисунок В.11 – Робота з даними json

РОБОТА З БАЗОЮ ДАНИХ

Структура бази даних

i	Name	Date	Way	Square
Test1.json	2018.12.26 04:39	D:/Test1.json	300	
Test2.json	2018.12.26 04:40	D:/Test2.json	257.5	
Test3.json	2018.12.26 04:41	D:/Test3.json	12.5	

Запит на створення таблиці

```
"CREATE TABLE " TABLE " ("
  TABLE_FNAME " BLOB NOT NULL,"
  TABLE_FDATE " VARCHAR(50) NOT NULL,"
  TABLE_FWAY " BLOB NOT NULL,"
  TABLE_FSQUARE " REAL NOT NULL"
)"
```

Запит на вставку в таблицю

```
"INSERT INTO " TABLE " ( " TABLE_FNAME ", "
  TABLE_FDATE ", "
  TABLE_FWAY ", "
  TABLE_FSQUARE" ) "
"VALUES (:FName, :FData, :FWay, :FSquare)"
```

Запит на видалення з таблиці

```
"DELETE FROM " TABLE " WHERE " TABLE_FWAY "= " + fWay+""
```

Запит на отримання значень з усіх полів таблиці

```
"SELECT * FROM " TABLE
```

12

Рисунок В.12 – Робота з базою даних

ВІКНО ЗАВАНТАЖЕННЯ ПРОГРАМИ



CoRe Planner v1.0



2018

... +  +  +  + ...

13

Рисунок В.13 – Вікно завантаження програми

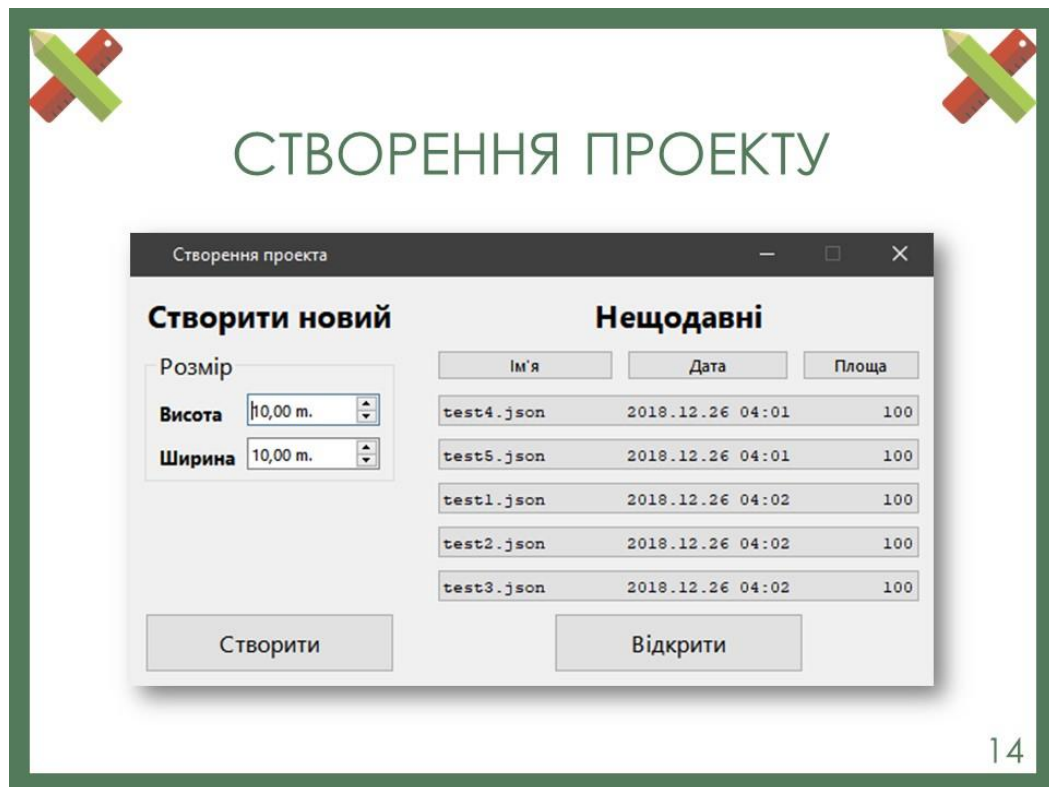


Рисунок В.14 – Створення проекту

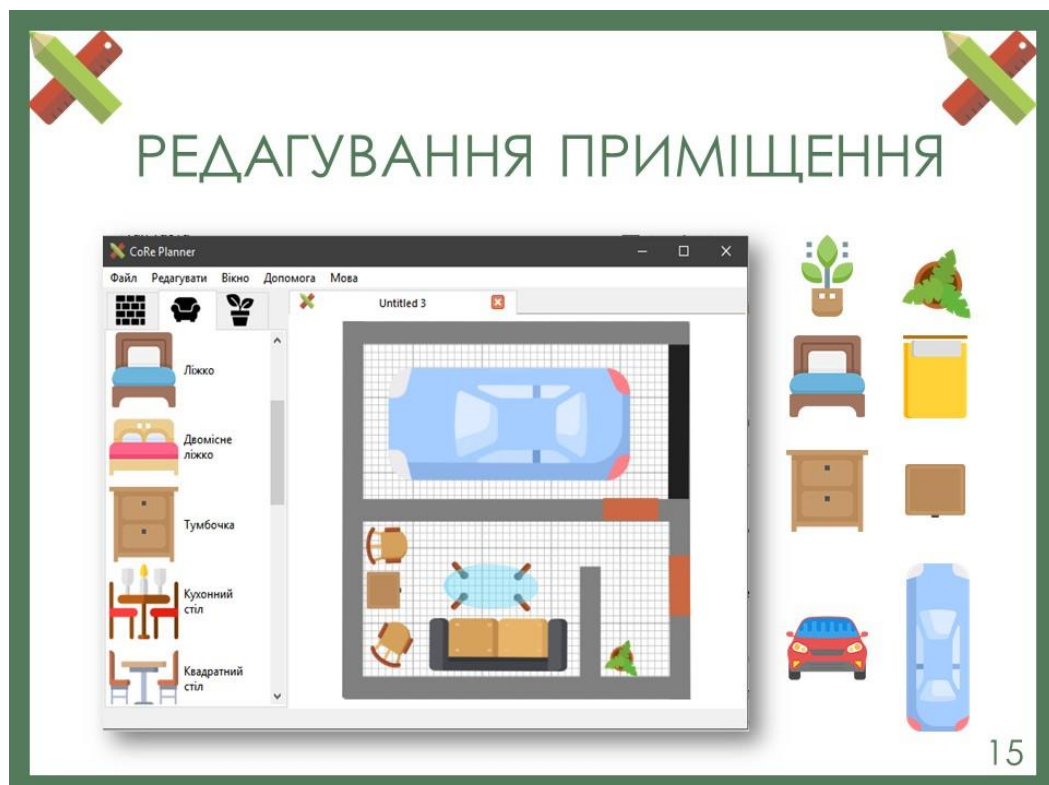


Рисунок В.15 – Редагування приміщення

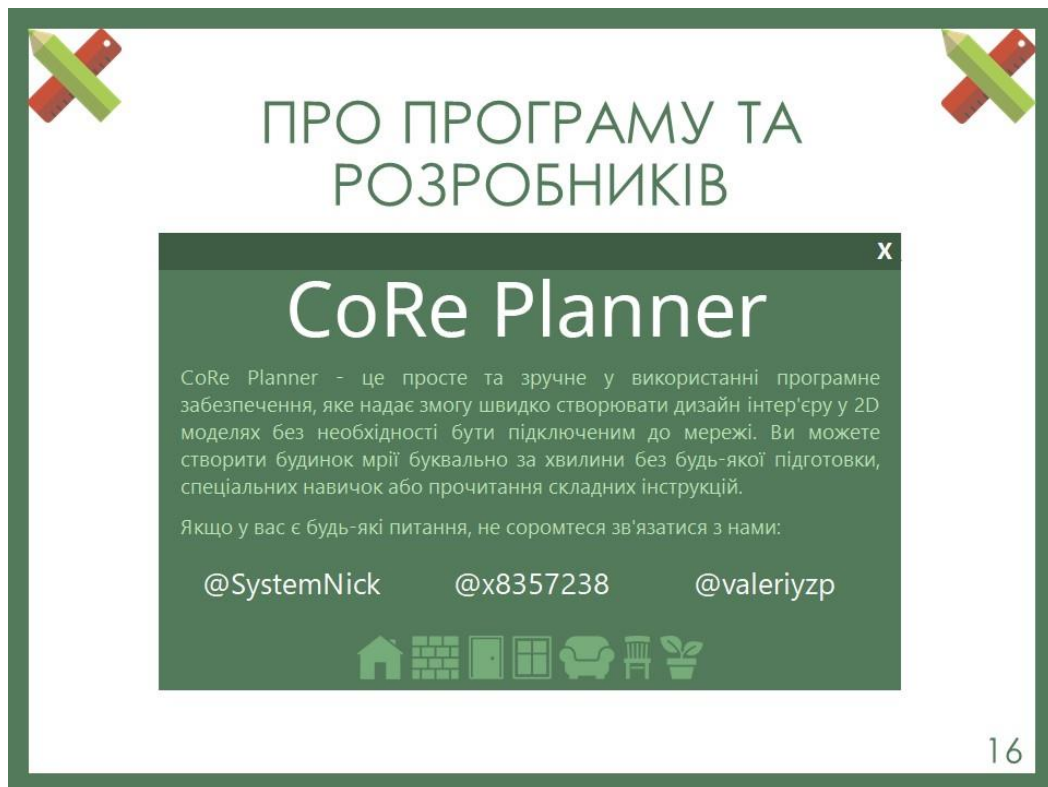


Рисунок В.16 – Про програму та розробників

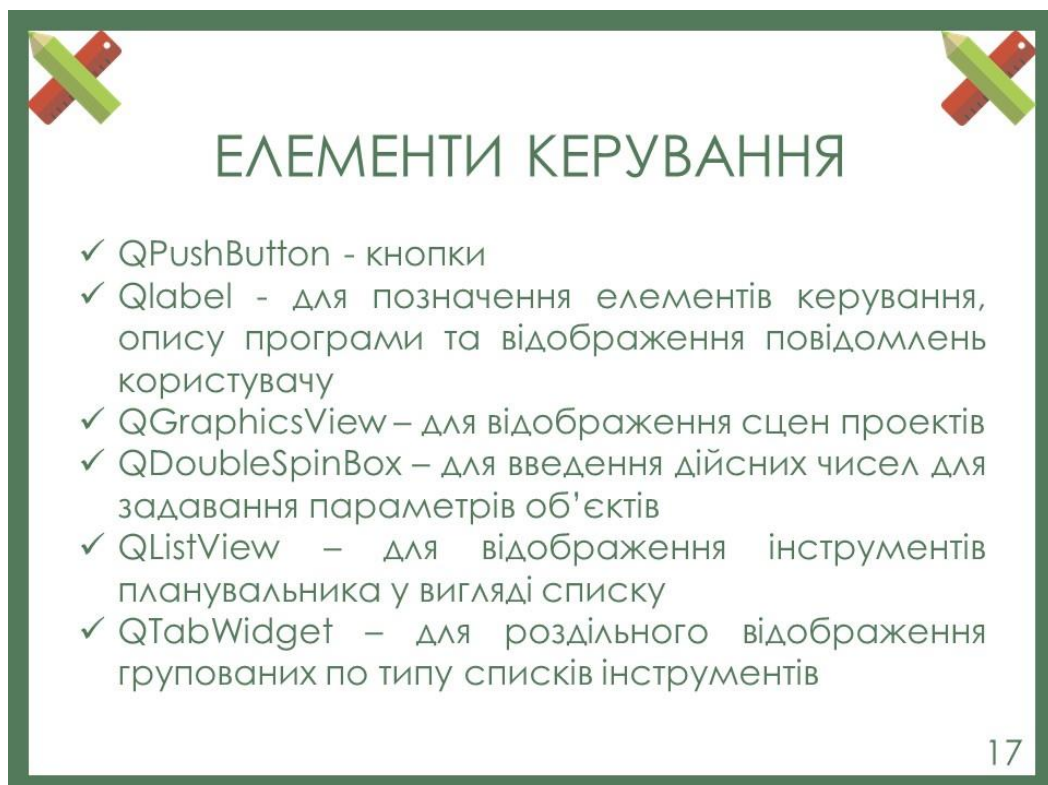


Рисунок В.17 – Елементи керування





РЕЗУЛЬТАТИ РОБОТИ



- ✓ Виконано розробку проєктувальника приміщень «CoRe Planner»
- ✓ Посилені знання та навички щодо роботи у середовищі «Qt Creator» з мовою C++
- ✓ Засвоєно основні принципи роботи з базами даних
- ✓ Вивчено алгоритми створення інтерфейсу користувача

18

Рисунок В.18 – Результати роботи



ДЯКУЄМО ЗА УВАГУ!

19

Рисунок В.19 – Фінальний слайд