МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРЕДОВИЩА ДЛЯ ФРЕЙМОВОГО ПРЕДСТАВЛЕННЯ НАВЧАЛЬНОГО МАТЕРІАЛУ»**

Виконала: студентка 2 курсу, групи 8.1228
спеціальності 122 комп'ютерні науки
освітньої програми комп'ютерні науки
(шифр і назва спеціальності)
Є. Л. Сорочинська
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к.п.н., Пшенична О.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент д.пед.н., професор кафедри фізичної культури і спорту, Клопов Р.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Напрям підготовки 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук, к.т.н., доцент

_____ Борю С. Ю.
(підпис)

« 29 » травня 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ (СТУДЕНТЦІ)

Сорочинській Єлизаветі Леонідівні
(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Програмна реалізація середовища для фреймового представлення навчального матеріалу
керівник роботи (проекту) Пшенична Олена Станіславівна, к.п.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » травня 2019 року № 811-с

2. Строк подання студентом роботи 16.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розроблений програмний продукт відповідно до теми кваліфікаційної роботи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	10.06.19	
2.	Збір вихідних даних.	05.07.19	
3.	Обробка методичних та теоретичних	28.08.19	
	джерел.		
4.	Розробка першого та другого розділу.	25.09.19	
5.	Розробка третього розділу.	16.10.19	
6.	Оформлення та нормоконтроль	16.12.19	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	14.01.20	

Студент _____ Є. Л. Сорочинська
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.С. Пшенична
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Програмна реалізація середовища для фреймового представлення навчального матеріалу»: 71 с., 24 рис., 3 табл., 8 джерел, 2 додатки.

ВИКЛАДАННЯ, МЕТОД, МОВА ПРОГРАМУВАННЯ, НАВЧАЛЬНИЙ МАТЕРІАЛ, ПІДХІД, РОЗРОБКА, СЕРЕДОВИЩЕ, СТРУКТУРА, ФРЕЙМ.

Об’єкт дослідження – фреймовий підхід та його використання в викладанні.

Мета роботи: розробити програмне середовище для використання фреймового представлення матеріалу для навчання.

Метод дослідження – аналітичний, оглядовий.

Для реалізації поставленої мети було вирішено такі завдання:

− визначено сучасні методи подачі навчального матеріалу;

− проаналізовано актуальність проблем розробки середовища;

− розглянуто існуючі програми для застосування фреймового методу;

− проведено аналіз і вибір технологій для розробки середовища;

− розроблено середовище для використання фреймового методу у навчанні .

При послідовному вирішенні поставлених завдань було розроблене середовище для застосування фреймового методу в викладанні.

При виконанні роботи застосовано мову програмування «Delphi».

SUMMARY

Master's Qualification Thesis « Software Implementation of Environment for Frame Presentation of Educational Material»: 71 pages, 24 figures, 3 table, 8 references, 2 supplements.

APPROACH, DEVELOPMENT, ENVIRONMENT, FRAME, METHOD, PROGRAMMING LANGUAGE, STRUCTURE, TEACHING, TRAINING MATERIAL.

The object of the study is frame approach and its use in teaching.

The aim of the study is to develop a software environment for the use of a framework approach in teaching.

The methods of research are analytical and survey.

In order to achieve the goal, the following tasks were solved:

– defines modern methods of information submission;

– analyzes the relevance of problems of environment development;

– existing programs for application of the frame method are considered;

– analysis and selection of technologies for environment development;

– environment for using the frame method in teaching has been developed.

In the sequential solution of the tasks, the environment was developed for the use of the frame method in teaching.

Delphi programming language was used when doing the job.

ЗМІСТ

ВСТУП

Фреймові засоби подання знань в педагогіці – об'єктивна інновація, викликана необхідністю інтенсифікувати навчальний процес, оскільки з прискоренням науково-технічного прогресу збільшуються темпи накопичення знань і зростання інформації. З психології відомо, що знання засвоюються, укладаються і зберігаються в довгостроковій пам'яті в стислому вигляді – у вигляді когнітивних ментальних структур – фреймів.

Ідея застосування фреймового підходу в навчанні полягає в тому, що якщо знання засвоюються і зберігаються в пам'яті у вигляді фреймів, то і представляти знання в процесі навчання треба теж у вигляді фреймів. Таким чином, якщо представляти навчальну інформацію учням в структурованому, згорнутому вигляді – у вигляді фреймових опор, можна істотно інтенсифікувати навчальний процес.

У ході проведених досліджень були розглянуті програми, які дають можливість створення фреймів. Але всі програми не спеціалізовані на фреймовій структурі. Такім чином, можна зробити висновок, що існуючі програми не є достатніми, а саме з цього питання, створення середовища для фреймового подання навчального матеріалу є досить актуальним.

Головною метою роботи є створення простої та зручної програми виключно під використання фреймів. Яка буде вельми комфортна у використанні як для студентів, так і для викладачів.

Однак середовище вимагає оптимізації і доопрацювань для створення повноцінного інструменту для користувача, який він зможе застосовувати для коректного відображення фреймового методу.

1 ФРЕЙМИ ЯК ЗАСОБИ ЕФЕКТИВНОГО ПРЕДСТАВЛЕННЯ НАВЧАЛЬНОГО МАТЕРІАЛУ

Основна ознака розвитку технології – збільшення обсягів досліджуваних знань без збільшення навчального часу. Сьогодні практично всі розвинуті країни усвідомили необхідність реформування систем освіти з тим, щоб студент дійсно став центральною фігурою навчального процесу.

Гуманістичний характер освіти в системі вищої школи передбачає, що в центрі освітнього процесу знаходиться особистість – студент. Пізнавальна діяльність і надання допомоги студентам з боку педагога є провідними в тандемі «викладач студент». Саме така система освіти відображає гуманістичний напрямок в інноваційній педагогіці.

Будь-яка зміна наукових парадигм в сферах знань має неминуче впливати і на методику. Друга половина XX століття ознаменувалася вступом науки в нову «еру» – когнітивну, що характеризується інтересом до вивчення пізнавальних процесів, проблем пам'яті і мислення, співвідношення останнього з мовою [1].

Процес життєдіяльності індивіда розглядається в когнітології як процес отримання, обробки, зберігання, породження і використання ним знань. При цьому наявні в індивіда знання не тільки грають важливу роль в розумінні навколишнього світу, а й активно використовуються в процесі вербальної комунікації, а також детермінують поведінку людини [2]. Саме під цим впливом з’явилося поняття фрейму.

Тому фреймові способи подання знань в педагогіці – об'єктивна інновація, викликана необхідністю інтенсифікувати навчальний процес, так як з прискоренням науково-технічного прогресу збільшуються темпи накопичення знань і зростання інформації.

1.1 Фрейм в освітньому процесі

Аналіз джерел, пов’язаних з дослідженням ефективності різних способів запам’ятовування інформації, дозволив становити, що увага, сприйняття, мислення, пам'ять та мовлення суттєво впливають на збереження знань її «стискання» і візуалізації, які технічно можуть досягатися різними способами.

До найбільш уживаних у практиці навчання природничо-математичних дисциплін відносять схемно-знакові моделі подання навчальної інформації (див. рис 1.1).

Кожна із зазначених моделей має свою специфіку:

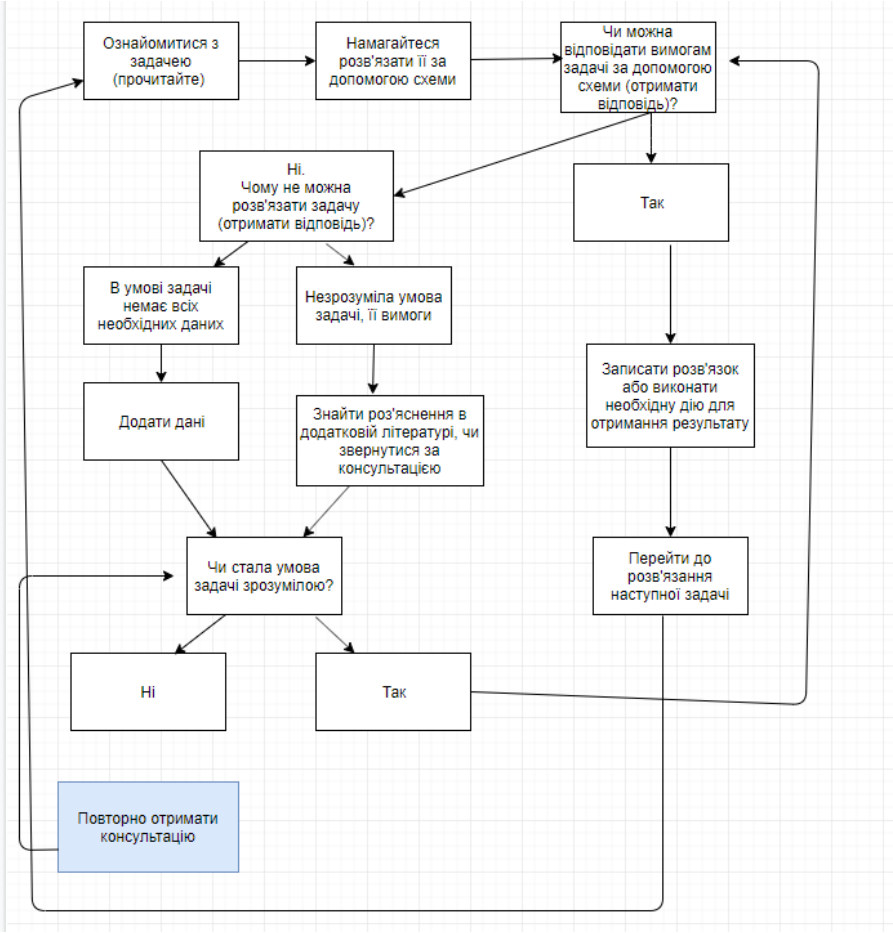

Рисунок 1.1 – Зображення онлайн редактора Draw.io

− логічна структура навчальної інформації – модель, яка найчастіше використовується для запису математичних аксіом і теорем із використанням

логіки предикатів, що дозволяє скоротити кількість записуваних «знаків» у кілька разів;

− продукційна модель являє собою набір правил або алгоритмічних приписів для виконання певної навчальної дії. Якщо звичайна інструкція складається з декількох, а іноді й великої кількості правил (продукцій), то продукційна модель зводить їх в одну візуальну композицію з усіма зв'язками й розгалуженнями;

− модель семантичної мережі, яка використовується для розкриття обсягу поняття, тобто тих, його різновидів, котрі характеризують даний предмет. Прикладами можуть служити формально-логічні відображення блоків інформації (графи, блок-схеми, дерева);

− фреймова модель, якою в дидактиці називають спосіб організації повторення навчального матеріалу (концепт) і навчального часу (сценарій), що застосовується до дисциплін, у яких є повторюване змістове ядро.

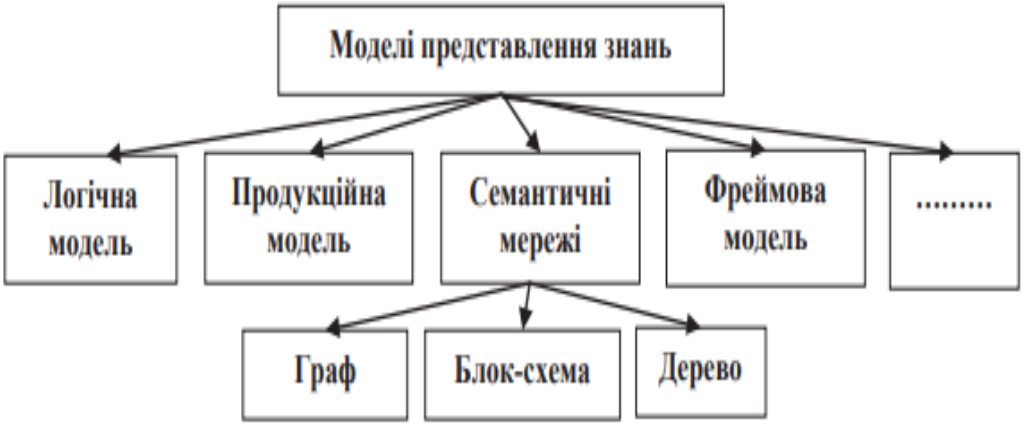

Рисунок 1.2 – Моделі представлення знань

Остання технологія максимально ефективна у вивченні тих дисциплін, у яких можна виділили змістове ядро (функції, процеси, властивості, характеристики тощо), яке розглядається і повторюється в усіх темах, розділах, курсах.

Якщо говорити про фрейм як технологія навчання, то це одиниця подання знань, що має однакову структуру, деталі якої при необхідності

можуть змінюватися відповідно до ситуації. За допомогою фреймової моделі можна «стискати», структурувати і систематизувати інформацію у вигляді схем, таблиць, матриць.

Структура фрейму передбачає наявність в якості елементів порожніх комірок, вікон, рядків (слотів), що повинні заповнюватися і можуть багаторазово перезавантажуватися новою інформацією (на відміну від опорних конспектів і структурних схем).

Слоти, що заповнюються інформацією, утворюють варіативну частину фрейму, постійні ключові слова, які входять до каркасної схеми – інваріатну.

Ідея застосування фреймового підходу в навчанні полягає в тому, що якщо знання засвоюються і зберігаються в пам'яті у вигляді фреймів, то і представляти знання в процесі навчання треба теж у вигляді фреймів. У цьому полягає основний сенс фреймового підходу в навчанні будь-якої дисципліни.

При цьому фреймова опора розглядається в контексті теорії поетапного формування розумових дій як інструкція для орієнтовної основи дій. Таким чином, якщо представляти навчальну інформацію учням в структурованому, згорнутому вигляді – у вигляді фреймових опор, можна істотно інтенсифікувати навчальний процес.

Окремі автори користуються декількома поняттями для позначення фрейму, поділяючи їх за статичністю (каркас будови) і динамічністю (сценарієм).

Якщо говорити про ознаки фреймів, то науковці до них відносять: стереотипність, повторюваність, наявність рамки, можливості візуалізації, наявність ключових слів, ментальність, універсальність, скелетну форму (наявність каркасу з порожніми вікнами), асоціативні зв’язки, фіксацію аналогій, узагальнень, правил і принципів.

Отже, ґрунтуючись на визначенні фрейму, можна визначити поняття фреймування. Фреймуванням називають високоефективний спосіб ущільнення інформації у вигляді схем, моделей, алгоритмів-сценаріїв, який

дозволяє розміщувати і зберігати її в довготривалій пам’яті. Це один із методів, що забезпечує якісне навчання в стислий термін за рахунок ущільнення навчального матеріалу зі збереженням у ньому кількості одиниць інформації, необхідної для засвоєння студентами.

Фреймовий підхід ґрунтується на ідеї застосування фреймів у процесі навчання, яка полягає у тому, що оскільки знання засвоюються у вигляді фреймів, то й подавати їх треба теж у вигляді фреймів. При цьому дотримуватися розуміння фрейму як каркасної структури подання стереотипної навчальної інформації, що містить інваріантну та варіативну складові, які включають слоти – пусті вікна або строки, котрі заповнюють студенти, і ключові слова як зв’язки між слотами, а також правила, що задають методику (когнітивна методика навчання).

В основі когнітивної методики навчання лежить три категорії: «знання, розуміння, вміння», на відміну від традиційної «знання, уміння, навички».

Фрейми – це змістові структури, які подаються у графічному вигляді (схеми або таблиці), а фрейми-сценарії – у текстовому вигляді.

Фреймову схему можна відрізнити від інших видів опор візуального сприйняття, за такими критеріями:

− за каркасом, що відображає стереотипні характеристики змісту;

− за системами слот і системами ключових словосполучень, що утворюють каркас. Та їх місцезнаходження не змінюється;

− наявність постійного сценарію (узагальненого плану) відповіді;

− багаторазове використання фреймових схем-опор при вивченні нових ситуацій.

При роботі з фреймами студенти бачать не тільки, те що треба говорити, а й те, в якій послідовності і як говорити. Саме в цьому полягає цінність фреймових схем.

1.2 Існуючі програми для конструювання фреймів

На сьогоднішній день існує безліч програм конструювання діаграм, фреймів, які допомагають студентам краще засвоїти матеріал. Існують як онлайн, так і офлайн програми. Проте деякі мають як і недоліки, так і переваги. Перевагу мають онлайн ресурси, так як додаток не потрібно встановлювати на пристрій, не важливо які параметри він має, необхідно мати тільки доступ до мережі Інтернет. Загалом всі вони безкоштовні, проте деякі інструменти можна використовувати купуючи преміум-послуги. Найвідоміші з них: Draw.io, UMLet, yEd, DiagramDesigner.

Але дані додатки мають свої мінуси:

- великі затрати трафіку Інтернету;
- для використання інструментів необхідна реєстрація;
- постійна реклама;
- складний інтерфейс;
- при завантаженні готового фрейму, додається емблема ресурсу.

Також нижче представлено, зображення онлайн-редактора Draw.io (див. рис. 1.3).

Draw.io – це сервіс, призначений для формування діаграм, фреймів і схем. Сервіс розділений на три частини – меню, панель об'єктів і сам документ.

За допомогою веб-сервісу Draw.io можна створювати:

- діаграми;
- фрейми;
- UML-моделі;
- вставка в діаграму зображень;
- графіки;
- блок-схеми;
- форми.

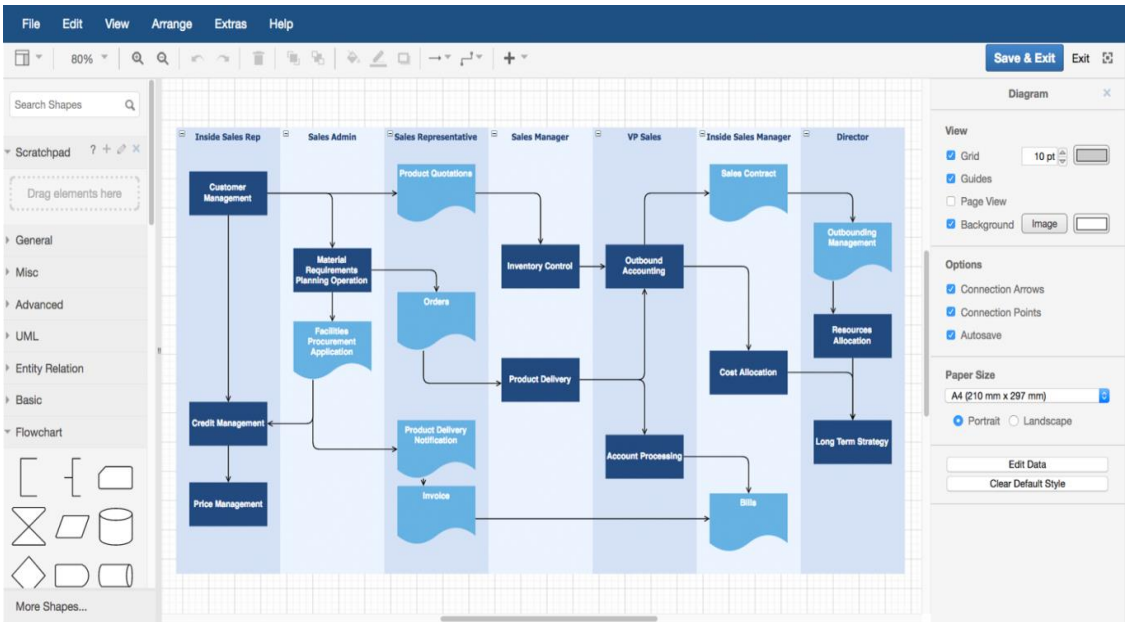

Рисунок 1.3 – Зображення редактора Draw.io онлайн

Для початку користувач може вибрати об'єкт з панелі, переглянувши категорії, і перенести мишею об'єкт в документ. Для з'єднання об'єктів блок-схеми необхідно виділити другий об'єкт і навести покажчиком на перший, далі з'явиться зелений прапорець і за допомогою нього виконується перетягування.

В меню сервісу діаграму або фрейм можна відформатувати наступними настройками:

- стиль шрифту;
- колір фону документа або об'єктів;
- тіні і ступінь прозорості;
- колір і товщина ліній;
- заливка і градієнт.

Також доступний експорт готових схем в зображення (PNG, GIF, JPG, PDF), синхронізація отриманих документів з Google Диском.

UMLet – безкоштовна програма для створення фреймів, діаграм. Програма підтримує всі типи діаграм. UMLet дозволяє швидко створити фрейм, тому що редагування властивостей об'єктів відбуваються в текстовому вигляді (див. рис. 1.4).

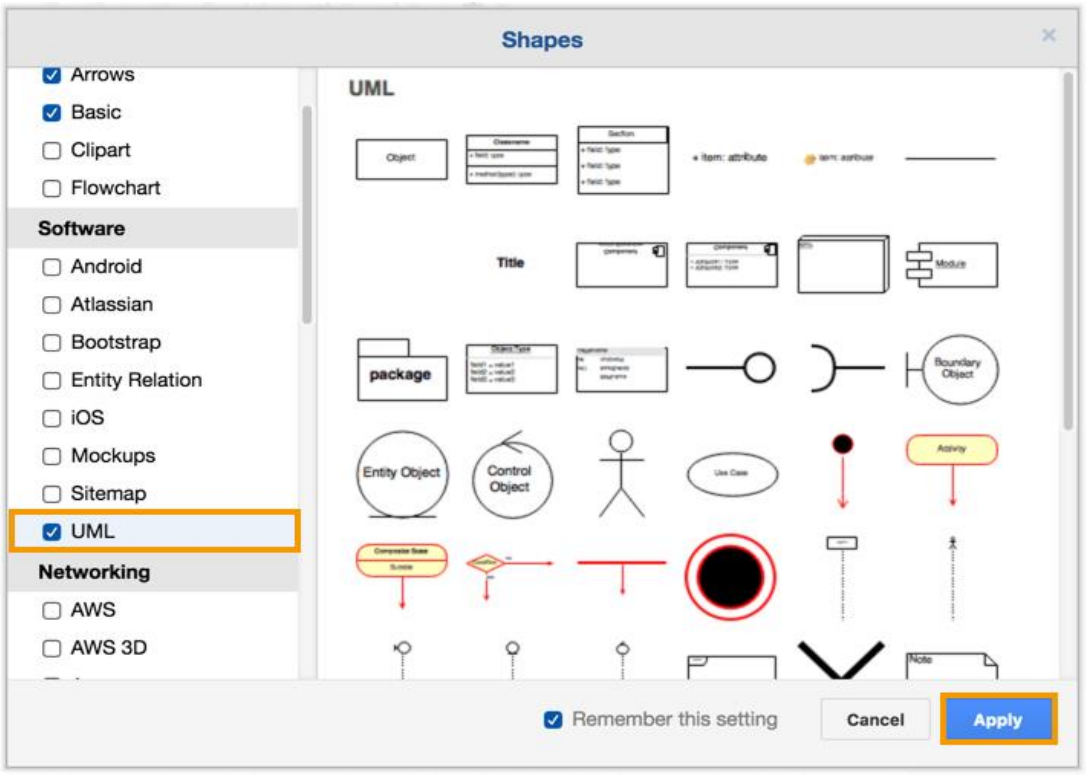

Рисунок 1.4 – Програма для створення фреймів UMLet

Діаграму можна зберегти в графічному файлі або роздрукувати на принтері. Панель компонентів в програмі незвичайна, вона є невеликим полем, в якому видно як компоненти виглядають (див рис 1.5). Варто відзначити, що UMLet також поставляється у вигляді плагіна для Eclipse.

yEd – безкоштовний редактор діаграм, який можна представити у вигляді графа. Програма підтримує велику кількість різних діаграм, та фреймів: UML, діаграми мережі, блок-схеми, діаграми процесу, схеми (див рис 1.6). yEd має зрозумілий інтерфейс: робоча зона, панель елементів, властивості об'єктів, панель інструментів. Варто відзначити кілька особливостей даної програми.

Наприклад, yEd може проаналізувати граф і розрахувати деякі з їхніх властивостей або розмістити елементи фрейму, діаграми за заданим алгоритмом.

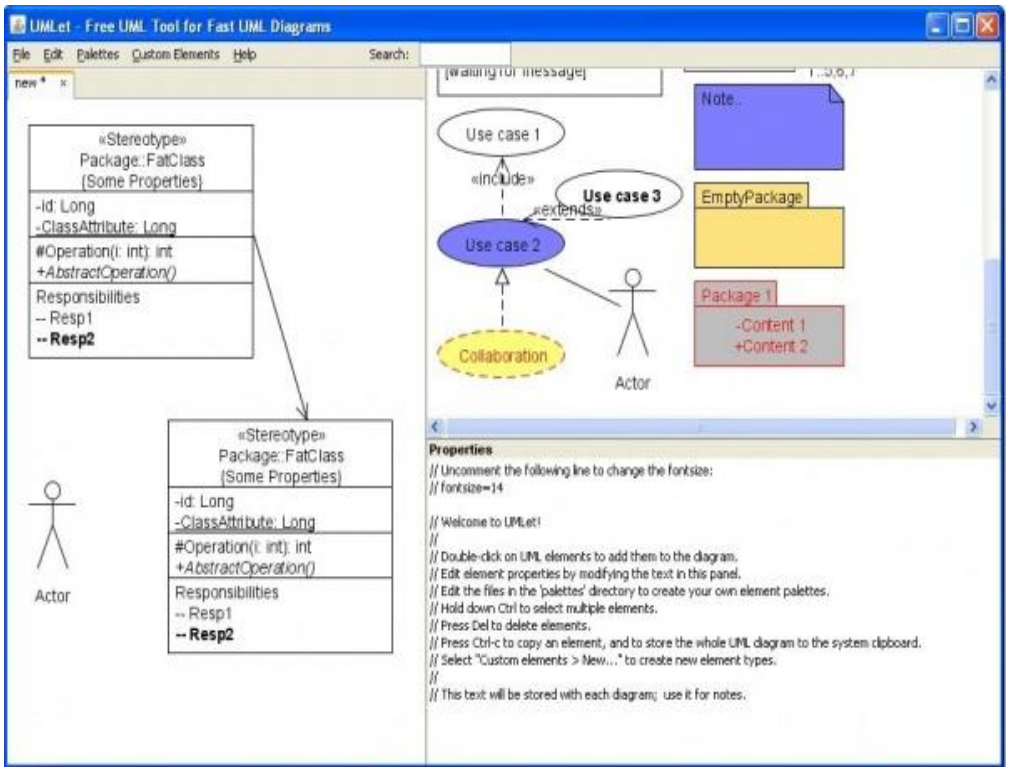

Рисунок 1.5 – Програма для створення фреймів UMLet

Ця функція може бути зручна, коли фрейм неструктурований. Додаток підтримує такі формати для збереження діаграми: GraphML, стиснений GraphML, ygf, gml, xgml і tgf.

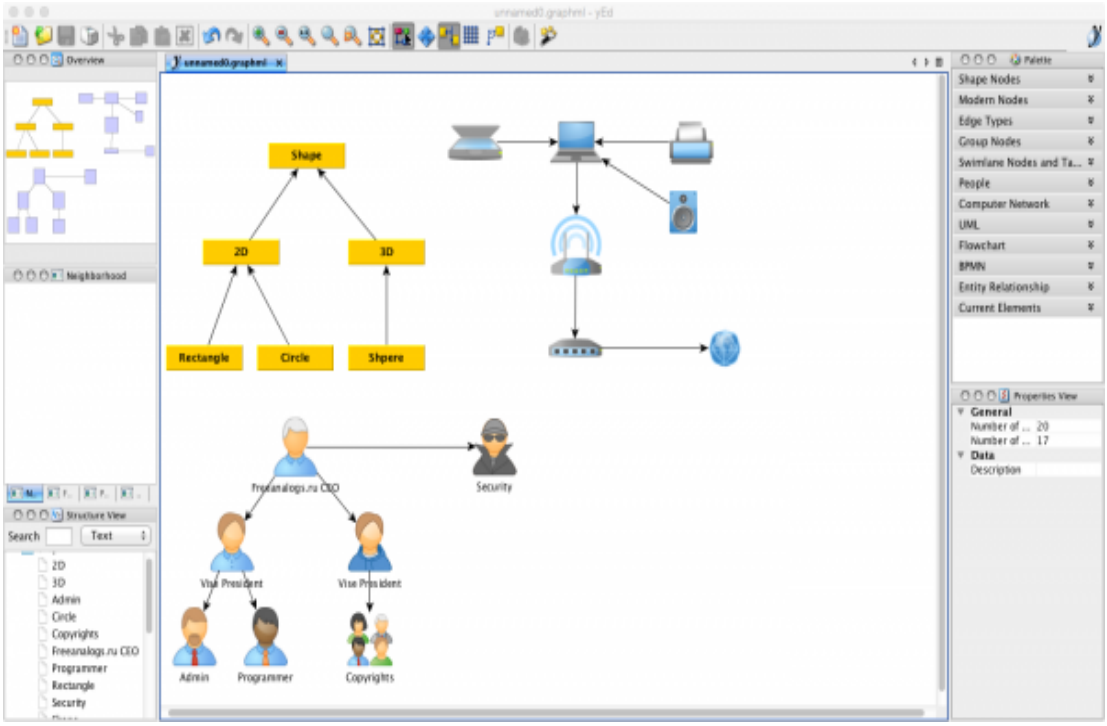

Рисунок 1.6 – Редактор діаграм yEd

DiagramDesigner – невелика програма для створення фрейму та схем. Інтерфейс програми представляє вікно для діаграми в центрі, панель елементів праворуч і поточні елементи зліва (див. рис. 1.7).

Для введення тексту для всіх елементів використовується один і той же діалог.

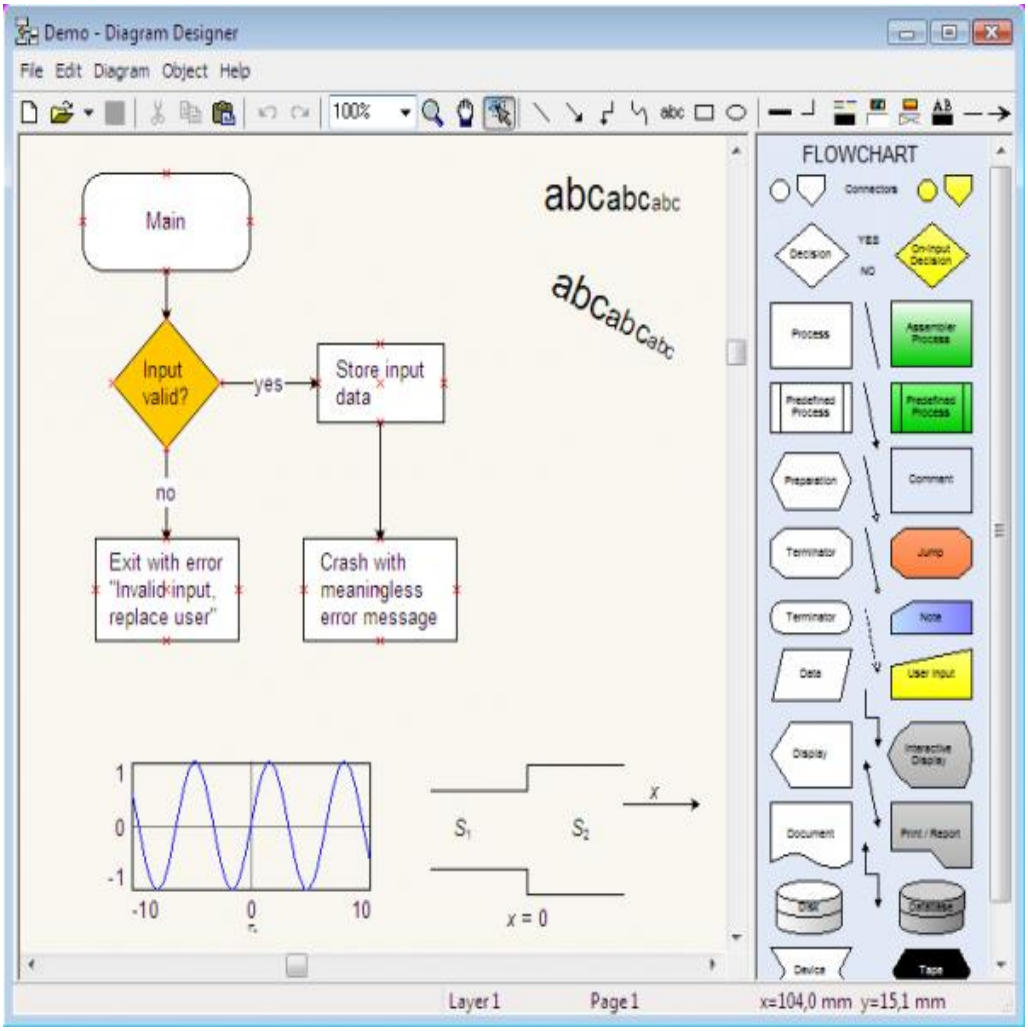

Рисунок 1.7 – Програма для створення фрейму DiagramDesigner

З його допомогою можна задати не тільки текст, а й форматування. Правда форматування задається командами, що може бути не завжди зручно.

Сучасні редактори не позбавлені недоліків, однак грамотне їх використання дозволяє вирішити більшість завдань, що виникають при створенні фреймів. Вони дозволяють в якійсь мірі створювати блок-схеми, та мають майже весь набір інструментів.

1.3 Актуальність розробки середовища

Серед актуальних проблем сучасної методичної науки важливе місце посідає проблема підвищення якості знань учнів і студентів. Її розв’язання пов’язане з розвитком когнітивної сфери суб’єктів навчання, до складу якої входять п’ять когнітивних процесів: увага, сприйняття, мислення, пам'ять та мовлення. Кожен із них відіграє свою роль у творення знання. Пам'ять забезпечує збереження сприйнятої й осмисленої інформації у свідомості людини.

Крім пам’яті невід'ємною частиною пізнання є абстрактне мислення (до нього можна віднести: поняття, судження, умовивід) і діяльнісний самостійний підхід до вирішення різних завдань. Тому актуальною залишається проблема цілісного формування знань студентів. Її головна задача полягає у відсутності розірваності в групах знань (сформованих в пам'яті), чуттєвому пізнанні, абстрактному мисленні, а також в вольовому прагненні до пізнання. Це можливо здійснити лише при цілісному впливі на всі складові людської природи.

Фреймова технологія, маючи в своєму розпорядженні широкі просторово-часові можливості, може вирішувати проблему цілісного формування особистості, включаючи рішення трьох найважливіших завдань:

- формування цілісних знань студента;
- формування абстрактного мислення;
- формування вольового прагнення до пізнання.

Ряд педагогічних ідей навчання, які не задовольняють сучасні вимоги і реалізуються в рамках класно-урочної системи, вважається перехідним від традиційного навчання до навчання із застосуванням фреймів які називаються ідеями, які передують фреймовим технологіям.

Можливість вирішувати за допомогою педагогічної фреймовой технології названі проблеми робить необхідним і важливим осмислення

практичного застосування даної технології і наповнення цього досвіду цілісним теоретичним змістом.

Тому актуальність роботи заключається в тому, що в ході роботи було створено програмне забезпечення для створення фреймових схем, розроблені підходи до систематизації і структурування навчального матеріалу і запропоновано методичні рекомендації реалізації даної технології.

У даному розділі розглянуто поняття фрейму, модель фреймового подання навчальних знань (концепт), яка є основою структурування навчального матеріалу в інтенсивності педагогічної технології, а також фреймову організацію навчального часу (сценарій). За підсумком розділу зроблено висновок, що існуючі редактори створення фрейму мають свої недоліки, та незручні в використанні.

Онлайн програми не потребують встановлення програмного додатку на носій, проте використовують багато трафіку Інтернету та повільно передають результат роботи. Якщо говорити про офлайн ресурси, то вони займають багато пам’яті на жорсткому пристрої та більшість інструментів обробки зображень мають платну основу.

Виникає проблема в розробленні додатку створення фрейму з всіма наявними інструментами.

2 ВИБІР РЕАЛІЗАЦІЇ ТА РОЗРОБКА СТРУКТУРИ ТА ІНТЕРФЕЙСУ ПРОГРАМИ МЕТОДІВ

2.1 Вибір мови програмування та середовища розробки

У якості мови програмування в дипломному проекті було обрано ObjectPascal,а відповідно середа розробки Delphi.

Система програмування Delphi версії 7 фірми Enterprise (Borland) надає найбільш широкі можливості для програмування додатків ОС Windows.

Delphi – це продукт Borland International для швидкого створення додатків. Процес створення інтерфейсу майбутньої програми нагадує забаву з ігровим комп'ютерним конструктором. Тому RAD-середовища ще називають візуальними середовищами розробки: якими ми бачимо робочі і діалогові вікна програми при проектуванні, такими вони і будуть, коли програма запрацює.

Високопродуктивний інструмент візуального побудови додатків включає в себе справжній компілятор коду і надає засоби візуального програмування, кілька схожі на ті, що можна виявити в Microsoft Visual Basic (вона не є RAD-системою) або в інших інструментах візуального проектування. В основі Delphi лежить мова Object Pascal, який є розширенням об'єктно-орієнтованої мови Pascal. У Delphi також входять локальний SQL-сервер, генератори звітів, бібліотеки візуальних компонентів, і інше, необхідне для того, щоб відчувати себе абсолютно впевненим при професійній розробці інформаційних систем або просто програм для Windows-середовища.

Перш за все, Delphi призначений для професійних розробників, бажаючих дуже швидко розробляти програми в архітектурі клієнт-сервер. Delphi виробляє невеликі за розмірами високоефективні виконавчі модулі (.exe і .dll).

Переваги Delphi в порівнянні з аналогічними програмними продуктами:

− швидкість розробки програми (RAD);

− висока продуктивність розробленого додатка;

− низькі вимоги розробленого додатка до ресурсів комп'ютера;

− нарощюваємість за рахунок вбудовування нових компонентів і інструментів в середу Delphi;

− можливість розробки нових компонентів і інструментів власними засобами Delphi (існуючі компоненти і інструменти доступні в початкових кодах);

− вдала опрацювання ієрархії об'єктів.

Система програмування Delphi розрахована на програмування різних додатків і надає велику кількість компонентів для цього. До того ж роботодавців цікавить, перш за все, швидкість і якість створення програм, а ці характеристики може забезпечити тільки середовище візуального проектування, здатна взяти на себе значні обсяги рутинної роботи по підготовці додатків, а також узгодити діяльність групи постановників, кодувальників, тестерів і технічних письменників . Можливості Delphi повністю відповідають цим вимогам і підходять для створення систем будь-якої складності.

Для того щоб обґрунтувати, чому наш вибір зупинився на Borland Delphi 7, досить просто перерахувати деякі недоліки мови C ++ в порівнянні з ObjectPascal:

− Треба робити багато ініціалізації (реєструвати клас вікна, організовувати цикл обробки повідомлень, створювати віконну функцію, піктограму та інше) і частково бути системним програмістом. На Delphi, системне програмування вже вбудовано і ініціалізація працює за замовчуванням, тому програміст головний акцент робить на своїх алгоритмах, а не на організації допоміжних робіт;

− Значно більша, в порівнянні з Object Pascal, складність мови, навіть, незважаючи на компактність коду, виникають складності в його сприйнятті.

− Одна особливість, на мій погляд, мови C ++ дуже псує цю мову – він чутливий до регістру символів, тобто змінна A і змінна a – це різні змінні.

− У Delphi класи (об'єкти) можуть розташовуватися тільки в динамічної пам'яті, а в C ++ в будь-який пам'яті (статична, стек, динамічна). Це додає безпеки програмування в Delphi.

2.2 Розробка структури програми

Перед створенням програми треба передбачити її структуру та основні елементи. Було розроблено структуру, яку наведено на малюнку (див. рис. 2.1.)

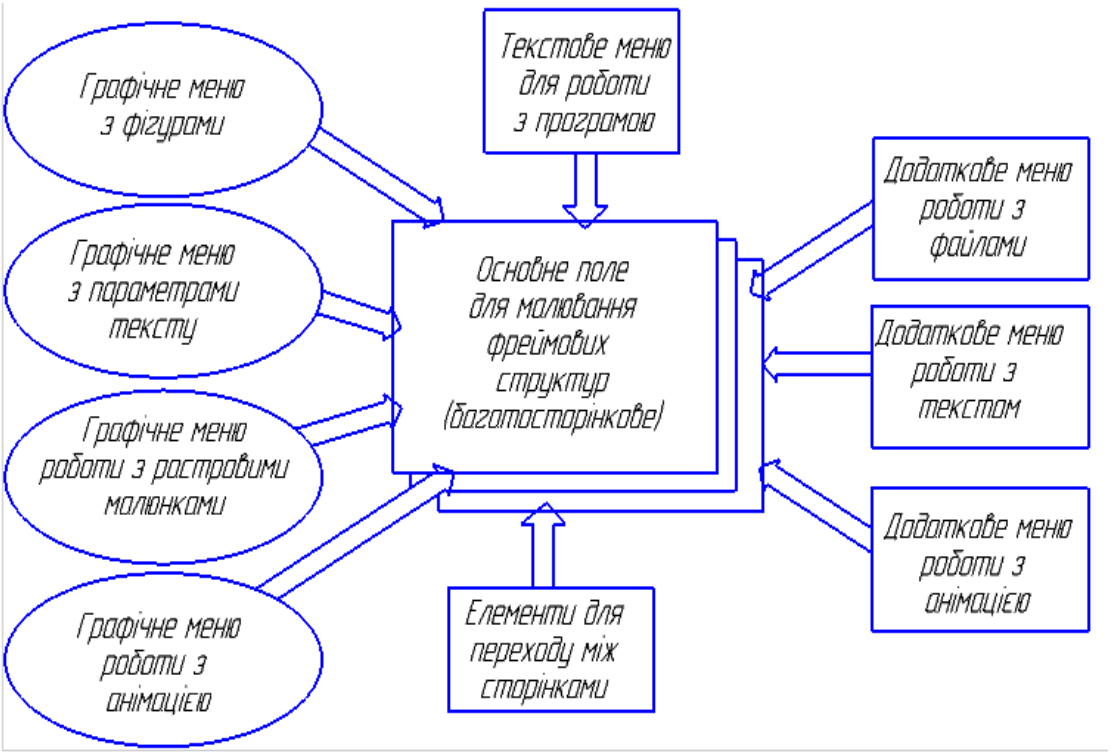

Рисунок 2.1 – Структура програми, що розробляється

Основним елементом структури є поле для малювання елементів фреймових структур, на якому будуть розташовуватися всі векторні та растрові елементи фреймової структури з можливою анімацією.

Оскільки створення фреймових структур передбачає можливість промальовування різного типу фігур, в програмі присутнє меню для малювання, яке складається з наступних елементів:

− малювання прямої лінії, як самого простого елементу, що може створювати різні контури та з'єднувати інші елементи;

− малювання лінії зі стрілочкою (стрілочка може бути як в одну сторону, так і в обидві сторони);

− малювання трикутників;

− малювання чотирикутників (ромбів, прямокутників, квадратів, трапецій – в залежності від параметрів сторін та кутів);

− малювання кіл та еліпсів;

− малювання секторів кіл та еліпсів;

− малювання кривої лінії по базовим точкам;

− малювання багатокутників.

Наступним елементом структури програми є меню з введення тексту, що може розміщуватися як всередині замкнених фігур (трикутників, чотирикутників, многокутників, кіл, еліпсів), так і в будь якому місці полотна для малювання. Це меню складається з наступних елементів:

− вставка тексту;

− вибір кольору тексту

− вибір кута нахилу;

− вибір направлення тексту.

− вибір ефектів тексту (тінь, контур, градієнт кольору та ін.);

Третім елементом структури програми є меню для вставки растрових зображень. Це меню складається з наступних елементів:

− вставка зображення з файлу на диску;

− масштаб зображення;

− обрізання зображення по певному контуру.

− налаштування яскравості та контрасту зображення.

Четвертим елементом програми є меню, що дозволяє працювати з анімаційними ефектами. Це меню включає в себе різні види ефектів, такі як:

− масштабування;

− рух по заданій траєкторії;

− обертання;

− поява через заданий час;

− зникнення через заданий час.

Крім основних елементів структури запропоновано додаткові елементи, що дозволять швидко налагоджувати та перевіряти роботу окремих елементів, виконанні графічними кнопками.

Додатковим меню є меню для роботи з файлами, яке дозволяє швидко працювати з файлами фреймових структур, що складається з наступних елементів:

− створити новий файл;

− відкрити файл;

− зберігти файл.

Другим додатковим меню швидкого доступу до параметрів є меню по роботі з текстом, що включає наступні елементи:

− тип шрифту;

− розмір шрифту;

− вирівнювання тексту (по лівому краю, по центру, по правому краю, по ширині);

− стиль тексту (жирний, курсив, підкреслений текст).

Третім додатковим графічним меню є меню по роботі з запуском анімації, яке включає наступні елементи:

− запуск анімації поточної сторінки;

− пауза в анімації сторінки;

− зупинити та скинути на початок анімацію поточної сторінки;

− налаштування параметрів анімації (часу відображення, порядок слідування анімації та ін.).

Крім графічних меню в програмі присутнє текстове меню, що дублює деякі елементи графічних меню, а також додатково має можливість загальних налаштувань програми, опис її роботи.

Окремою кнопкою пропонується винести запуск анімації всіх сторінок починаючи з першою та закінчуючи останньою.

Також для навігації між сторінками окремо пропонується створити елементи, що вказують номер поточної сторінки та можливість переходу на наступну та попередню сторінку, а також на сторінку з будь яким номером.

2.3 Розробка інтерфейсу користувача

Першим етапом при розробці програмного забезпечення під операційну систему Windows є створення інтерфейсу користувача, з розробкою елементів для керування роботою. Інструменти Delphi дозволяють створювати будь який елемент управління доступний в операційній системі.

Оскільки, згідно з розробленою структурою у нас наявні 4 графічних меню, 1 текстове меню та 3 додаткових меню з додатковими параметрами у вигляді графічних елементів, вони створювались наступним чином.

Для створення текстового меню було використано компонент MainMenuна вкладціStandard. Цей компонент дозволяє інтуїтивно складати меню додавання мишкою пункти та прописуючи їх назви. Для інтуїтивно розуміння приналежності елементу окремому пункту текстового меню всі елементи меню не лише в полі Caption, як вміщає напис, що виводиться користувачеві, було прописано зміст пунктів, а й у значенні параметруName було обрано відповідні імена кожному пункту. Це дозволить при написанні коду швидко розуміти з яким пунктом ми працюємо, оскільки

використовується об’єктно-орієнтований підхід, та кожен елемент інтерфейсу є об’єктом зі своїми параметрами, методами та подіями.

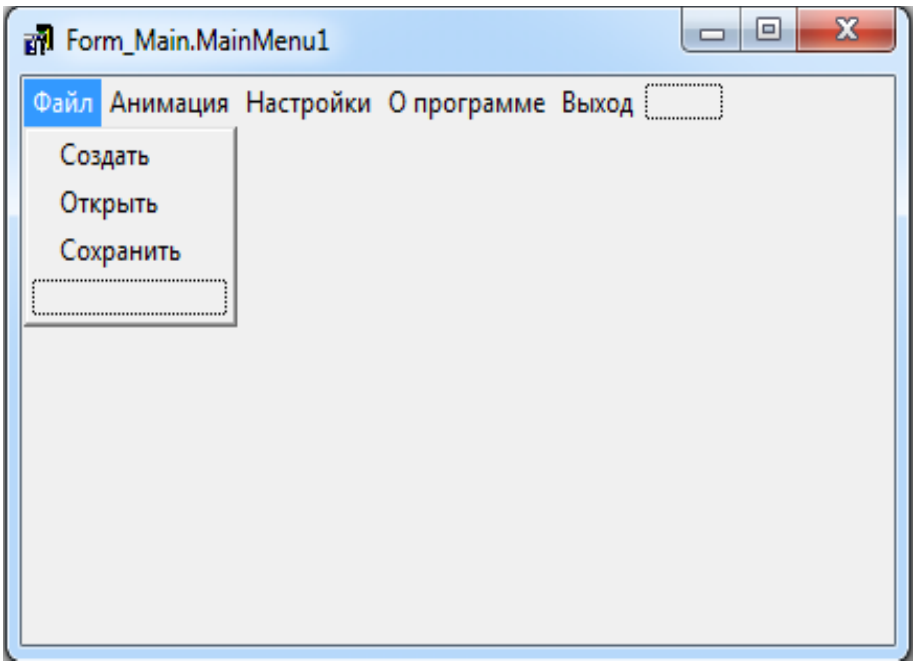

Рисунок 2.2 – Створення текстового меню в шапці програми

В таблиці 2.1 наведено перелік пунктів меню з їх підписами для відображення користувачеві (Caption) та назвами для програміста (Name).

Основні графічні меню було прийнято рішення зробити у якості великих кнопок з малюнком при натисканні на яких буде з’являтися контекстне меню з малюнками, що відображають підпункти.

Для кнопок з малюнками існує компонент BitBtn (див. рис.2.3) у вкладці Additional, який якості одного з основних параметрів має параметр Glypth, що дозволяє виводити на кнопці рисунок завантажений з файлу.

Як і з текстовим меню для простоти розуміння відношення елементів при програмуванні їх роботи всі їх назви були зроблені інтуїтивно зрозумілими.

Основні графічні меню, у якості елементів виклику яких на інтерфейсі користувача були використані компоненти BitBtn названі відповідно:

- BitBtn_Figures – меню малювання різних фігур;
- BitBtn_Text – меню вставки тексту та редагування його параметрів;

Таблиця 2.1 – Текстове меню з вказаними параметрами

<table>
<tr><td>Пункт «Файл» (N_File)</td><td>Пункт«Анімація» (N_Animation)</td><td rowspan="5">Пункт «Налаштування» (N_Settings)</td><td>Пункт «О програмі» (N_About)</td><td rowspan="5">Пункт «Вихід» (N_Exit)</td></tr>
<tr><td>Підпункт «Створити» (N_CreateNew)</td><td>Підпункт «Запустити» (N_Play)</td><td>Підпункт «Довідка» (N_Help)</td></tr>
<tr><td>Підпункт «Відкрити» (N_Open)</td><td>Підпункт «Зупинити» (N_Stop)</td><td>Підпункт «Автори» (N_Authors)</td></tr>
<tr><td>– Підпункт «Зберігти» – (N_Save)</td><td>– Підпункт «Пауза» – (N_Pause)</td><td rowspan="2"></td></tr>
<tr><td></td><td>Підпункт «Параметри затримок» (N_DelaySetup)</td></tr>
<tr><td></td><td>Підпункт «Анімація всіх сторінок» (N_AnimateAll)</td><td></td><td></td></tr>
</table>

– BitBtn_InsertImages – меню вставки растрових малюнків;

– BitBtn_Animation – меню додавання анімації.

Рисунок 2.3 – Компонент BitBtn на панелі інструментів Delphi

Для інтуїтивного розуміння користувачем функціоналу цих меню було намальовано відповідні малюнки та розміщено їх на відповідних кнопках (див. рис. 2.4). Всі 4 основних графічних меню були розміщенні зліва на основній формі програми та об’єднані елементом GroupBoxпанелі Standardз назвою «Основні функції» (GroupBox_MainBlocks).

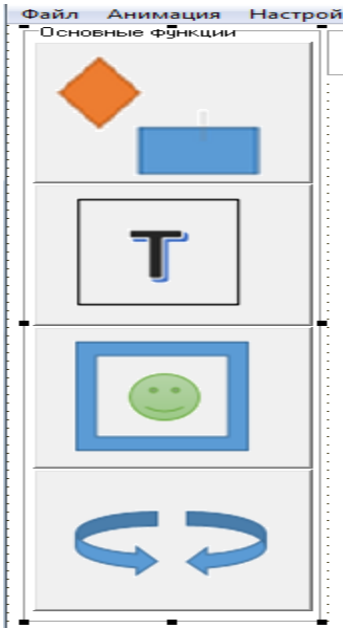

Рисунок 2.4 – Графічні меню основних функцій

Загалом засоби Delphi дозволяютьдуже просто розміщати елементи інтерфейсу на формах програми та змінювати їх розміри за допомогою миші (див. рис.2.5)

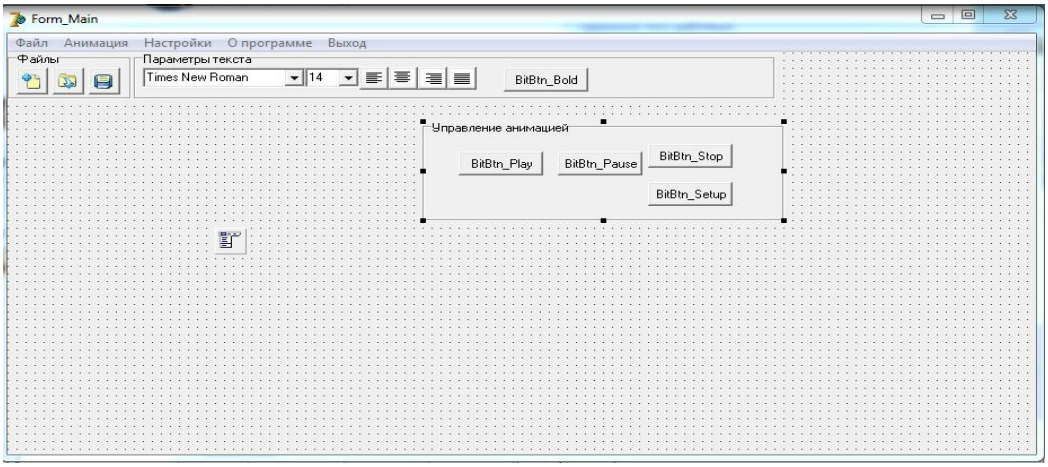

Рисунок 2.5 – Розміщення та опис параметрів елементів інтерфейсу користувача

Для створення додаткових графічних меню, що допомагають працювати з основними елементами було також створено 3 GroupBox з назвами:

– «Файли» (GroupBox_Files);

– «Параметри тексту» (GroupBox_Text);

– «Керування анімацією» (GroupBox_Animation).

Ці групи елементів були розміщенні в верхній частині програми під текстовим меню.

Для створення цих меню додатково, крім компоненту BitBtn були застосовані наступні елементи:

– OpenDialog – компонент для відкривання файлів;

– SaveDialog – компонент для зберігання файлів;

– ConboBox – компонент, що надає список, що вивалюється при натисненні на трикутник справа.

Оскільки основним елементом програми є поле для малювання, в центральній зоні програми було розміщення компонент Image мета якого забезпечити границі для малювання та у разі необхідності надати необхідний колір фону, що має встановлюватись у налаштуваннях та за замовчуванням є білим.

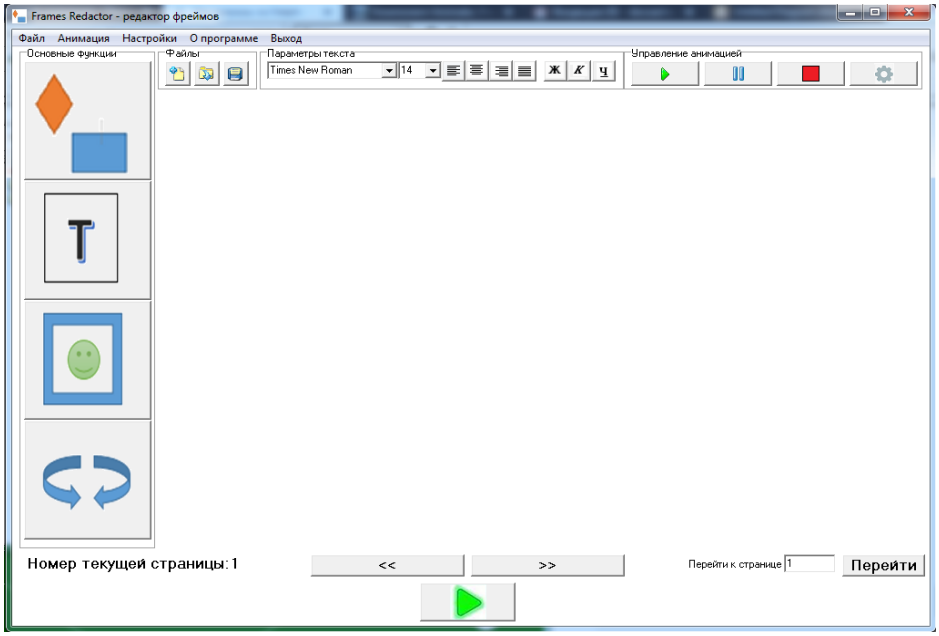

Рисунок 2.6 – Розроблений інтерфейс основного вікна програми

3 РОЗРОБКА СТРУКТУР ДАНИХ ТА НАПИСАННЯ ТЕКСТІВ ПРОГРАМНИХ МОДУЛІВ

3.1 Розробка структури зберігання даних

Оскільки після створення графічних анімованих об’єктів на полі для малювання нам необхідно передбачити формат їх зберігання для подальшого використання, редагування, демонстрації – важливим етапом розробки програми є розробка формату зберігання, що дозволить повністю передавати всі дії всіх об’єктів, промальованих на кожній сторінці анімованого фреймового середовища.

Насамперед треба визначитися з форматом зберігання намальованих фреймів, яких може буде дуже багато навіть на одній сторінці.

Концепція програмної роботи з фреймами передбачає використання в першу чергу векторної графіки оскільки саме вона дозволить чітко та стисло описувати фреймові структури. А отже треба провести аналіз існуючих форматів зберігання векторних зображень та можливість їх використання у нашій програмі.

Найпопулярніші форматифайлів векторної графики:

а) . AI (Adobe Illustrator);

б) .CDR (Computer Graphics Metafile);

в) .CGM (Computer Graphics Metafile);

г) .DXF (Drawing eXchange Format);

д) .EPS (Encapsulated PostScript);

е) .FH (FreeHand);

ж).PDF (Portable Document Format);

з) .TIFF (Tag Image File Format);

и) .WMF (Windows MetaFile);

- AI (Adobe Illustrator).

Векторний формат AI належить фірмі Adobe і є внутрішнім форматом векторного редактора Adobe Illustrator.

- CDR (Computer Graphics Metafile)

Векторний формат CDR належить фірмі Corel і є внутрішнім форматом векторного редактора CorelDRAW.

- CGM (Computer Graphics Metafile)

CGM популярний векторний формат (хоча застосований тут алгоритм дозволяє записувати і растрові картинки), прийнятий ANSI і використовуваний для перенесення даний на інші платформи. Теоретично основна перевага CGM – його незалежність від апаратної і програмної платформ. Однак постачальники настільки "поліпшили" цей формат, що, незважаючи на існування цілого зводу правил і схем шифрування / дешифрування, він вже не є універсальним. У будь-якому випадку CGM більше підходить для обміну даними між програмами і платформами, ніж для зберігання креслень і малюнків.

- DXF (Drawing eXchange Format)

Векторний формат DXF підтримують всі програми автоматизованого проектування: починаючи з пакета AutoCAD компанії Autodesk. Однак через його складність деякі додатки "вміють" тільки читати DXF-файли і не здатні зберігати дані в цьому форматі. У DXF реалізовані багато можливостей, які відсутні в більшості інших форматів, наприклад зберігання тривимірних об'єктів. Необхідно відзначити наявність прекрасного вбудованого кодувальника тексту.

- EPS (Encapsulated PostScript)

У форматі (EPS) використовуються як векторний, так і растровий способи запису інформації. EPS виник в результаті співпраці компаній Adobe Systems і Altsys з метою створення технології, яка дозволяє програмам працювати з PostScript-зображеннями (PostScript – універсальний, що не залежить від платформи мова опису сторінки, розроблений фірмою Adobe

Systems). Файл PostScript насправді являє собою набір команд, що виконується інтерпретатором мови PostScript при виведенні зображення, причому цей файл дійсно не залежить ні від апаратної платформи, ні від операційної системи. Оскільки PostScript є мовою програмування, то за допомогою інтерпретатора можна перекодувати PostScript-файл для його виведення і на принтері, що не підтримує PostScript. Команди PostScript дозволяють навіть управляти екраном монітора (як це реалізовано в ОС NeXTstep).

Однак між файлами в форматі EPS і PostScript існують відмінності. По суті, EPS дає можливість програмам працювати з графікою, не вдаючись безпосередньо до PostScript – коду. Цей формат дозволяє переносити дані на мові PostScript в стандартні для даної ОС формати представлення інформації, наприклад на екрані монітора. Для платформи Macintosh – це PICT-файли, для Windows – файли в форматах Windows MetaFile (WMF) або TIFF. На жаль, хоча потужна форма представлення даних у вигляді EPS дуже популярна, програми, що підтримують цей формат, не забезпечують повної сумісності. EPS-файли, створені одним додатком, часом не можуть бути відкриті або імпортовані іншим.

Зображення в файлі зберігається в двох варіантах (як в форматі TIF): основний варіант – це власне векторне зображення, збережене як опис на мові PostScript, і додатковий варіант – це піксельний зображення з зменшеним дозволом, що використовується (так само, як і в форматі TIF) для попереднього перегляду. Крім того, в програмах верстки піксельний зображення відображається на екрані і друкується на принтерах, які не підтримують мову PostScript.

Формат передбачає запис пиксельного зображення в форматах TIF або WMF. Якщо передбачається необхідність друку файлу EPS на PCL-принтері, слід зберегти документ з варіантом TIFF і достатнім дозволом.

Завдяки своїй надійності, сумісності з багатьма програмами і платформами і великий сукупності параметрів, що настроюються формат

EPS вибирають більшість розробників програмного і апаратного забезпечення.

− FH (FreeHand)

Векторний формат FH з порядковим номером версії належить фірмі Macromedia і є внутрішнім форматом векторного редактора FreeHand.

− PDF (Portable Document Format)

Векторний формат PDF (Portable Document Format - "стерпний формат документів") – це ще одна можливість мови PostScript, а саме: його оптимізована версія, орієнтована як міжплатформений формат, інтегруючий макет сторінки з ілюстраціями, як векторними, так і пиксельними, шрифтами, гіпертекстовими посиланнями, звуками і анімаційними фрагментами.

Формат PDF розроблявся Adobe Systems для електронного поширення публікацій. До його появи не існувало універсального вирішення цього завдання: щоб переглянути зверстаний документ, потрібно встановити додаток, в якому він був створений, всі використані шрифти, переписати все зовнішні файли (файли з зображеннями). Завдяки PDF всі ці кроки звелися до установки єдиної програми перегляду.

У форматі PDF можна записати будь-який документ, створений в будь-якій програмі. Файли в цьому форматі створюються програмою Distiller на основі файлу друку для будь-якого PostScript-принтера. Програма Distiller входить до складу пакету Adobe Acrobat, призначеного спеціально для створення, редагування та каталогізації PDF-документів. Отриманий PDF-документ можна переглянути за допомогою безкоштовно розповсюджуваної програми перегляду Acrobat Reader. При цьому документ повністю збереже свій зовнішній вигляд, включаючи векторні і растрові зображення, шрифти, розбиття на сторінки.

З появою третьої версії пакету Adobe Acrobat (поточна версія - 6.0) формат PDF став завойовувати сферу не тільки електронного поширення документів, але і додрукарської підготовки. Використовуваний на фотоскладальних автоматах формат .PS (PostScript) має істотний недолік:

неможливість побачити вміст зображення і отримати гарантію його правильності до виведення плівок. При використанні формату PDF ця проблема повністю вирішується: то, що оператор побачить в Acrobat Reader, то і вийде при друку. Зручна і можливість обмеженого редагування готових до друку PDF-документів, яка немислима для файлів друку PostScript. Використовуючи програму Adobe Acrobat (Acrobat Exchange), можна швидко змінити формат і порядок проходження сторінок, їх орієнтацію, вставити додаткові сторінки або видалити зайві і виправити дрібні помилки.

Для забезпечення невеликого розміру використовуються різні способи компресії:

− TIFF (Tag Image File Format)

Результатом об'єднаних зусиль компаній Aldus і Microsoft, спрямованих на подолання труднощів, які виникають при перенесенні графічних файлів з IBM-сумісних комп'ютерів на Macintosh і назад, став растровий формат Tag Image File Format (TIFF). Поява TIFF поклало також кінець деякого безладдя, що панувало в області програмного забезпечення для сканерів. Назва розшифровується як "тегів образотворчий файл". Цей графічний формат є досить складним, зате його структура передбачає як гнучкість записи даних, так і широкі можливості для розширення. Причина цього криється в принципі побудови файлу. Вся інформація, яка описувала цифрові дані зображення (геометричний розмір, дозвіл, глибину кольору) міститься не в заголовку файлу, як у багатьох інших форматів, а в спеціальних блоках ("тегах", що перекладається з англійської мови як "бирка"), які містять внутрішні визначення параметрів зображення. Наприклад, п'ята версія формату включає 45 різних тегів, хоча практично використовується набагато менше. Застосування тегів також дає можливість формулювати численні додаткові функції.

Існує п'ять типів TIFF-файлів:

− B – чорно-білі ілюстрації;

− F – зображення для факсів;

− G – напівтонові зображення (в цьому випадку кожна точка може бути будь-якого ступеня сірого, від 0% - білий колір, до 100% – чорний колір);

− P – кольорові зображення, що використовують власну кольорову палітру;

− R – фотореалістичні зображення, що записують для кожної точки червону, зелену і блакитну складові кольору.

В останніх версіях програми Adobe Photoshop цей формат дозволяє зберігати документи з шарами, що раніше було можливо тільки у внутрішньому форматі програми PSD. Також з'явилася можливість вибирати метод стиснення (LZW, Deflate або JPEG) і зберігати копії зображення з різним дозволом (функція "image pyramid").

Сукупність тегів утворює область IFD, що означає Image File Directory - "покажчик зображень файлу". Формат дозволяє включати в документ кілька таких покажчиків, а, отже, і кілька зображень. Зокрема, самим звичайним є розміщення в файлі формату TIFF невеликого контрольного зображення, яке називається thumbnail image - "мініатюра". Така мініатюра представляється в діалогових вікнах при виборі імені файлу. Ця функція надзвичайно корисна в роботі художників і дизайнерів, які створюють безліч зображень і зберігають їх в файлах з близькими назвами, наприклад "Портрет1.tif," Портрет2.tif і т. Д., А потім, після деякого часу, важко згадати не тільки відмінність між цими файлами, але навіть і об'єкт цих портретів.

У цьому форматі підтримуються відсічні контури, альфа-канали, а також інформація про автора, категорії зображення і ключових словах. Формат переноситься між платформами і легко імпортується в усі програми верстки, що робить його незамінним при підготовці документів для друку.

Формат TIFF дуже зручний, але розмір одержуваних файлів, великий (файл формату A4 в колірній моделі CMYK з роздільною здатністю 300 dpi, має розмір близько 40 Мбайт). Крім того, як і в разі CGM, існує кілька "діалектів" формату, які не кожна програма, яка підтримує TIFF, "розуміє".

− WMF (Windows MetaFile)

Аналогом формату PICT в світі Windows є WMF (Windows MetaFile), розроблений корпорацією Windows. Він підходить для зберігання векторних і растрових файлів і їх подальшого виведення, як на екрани моніторів, так і на друкують устрою. В цей формат конвертуються векторні зображення при перенесенні через буфер обміну Clipboard, тому для редагування формату ніякого спеціального додатку не існує.

Незважаючи на те, що формат WMF розроблений для середовища Windows, він підтримується графічними програмами на багатьох інших платформах і часто використовується для обміну даними з Windows-додатками. У разі векторної ілюстрації формат Encapsulated PostScript підтримує використання графічних елементів, збережених як WMF.

Формат WMF виявляється більш зручним, ніж, наприклад, EPS, коли необхідно вставити малюнки в документ, створений в текстовому редакторі або програмі верстки, а потім вивести його на екран монітора або роздрукувати на принтері, що не підтримує мову PostScript. Що стосується сверхпопулярної в Росії програми Word for Windows компанії Microsoft, то, за рідкісним винятком, будь-яку графіку, створену в інших додатках (наприклад CorelDraw), зручніше зберігати в форматі WMF і після цього поміщати в Word-документ за допомогою команди меню Insert-Picture.

Проводячи аналіз найпоширеніших форматів зберігання векторних зображень треба зробити висновок, що значна частина форматів є приватними та не описані, а частина використовується не зовсім для нашого випадку. Деякі формати складні в зберігання, деякі не дозволяють зберігати растрові об’єкти у якості елементів файлу. Але найголовніше те, що жоден з форматів не дозволяє зберігати анімаційні ефекти, а отже не підходить для наших цілей. Тому було прийнято рішення розробити власний формат зберігання векторних зображень з елементами анімації.

Перш за все наше фреймове середовище розділено на сторінки, а отже файл має мати роздільник сторінок. Другим важливим фактором є роздільник

векторних елементів. Третім важливим роздільником є початок блоку анімації.

Крім того необхідно на початку файлу вказувати кількість сторінок та розширення сторінки у точках, а на початку кожної сторінки вказувати який використовується фон.

Тепер більш детально розглянемо запропонований формат зберігання.

Файл представляє собою текстовий файл в якому з нового рядка позначається новий параметр або елемент. Розширення файлу пропонується зробити з назвою «frr»скорочено від назви розробленої програми FramesRedactor.

Файл починається з вказання приналежності файлу програмі: «FramesRedactorv.1.0»

Наступним параметром є вказання кількості сторінок з фреймами – пропонується у якості признаку цього параметру обрати слово в переводі з англійської Pages=X, де X кількість сторінок з фреймами.

Наступним параметром має бути розмір кожного зображення в точках, в якості опису надамо параметри у вигляді X=xxx||Y=yyy, де xxx – кількість пікселів по горизонталі, yyy – кількість пікселів по вертикалі з роздільником «||» оскільки зображення можуть бути з розміром по вертикалі та горизонталі на одиниці, десятки, сотні і навить тисячі пікселей, а отже ми не знаємо яку саме кількість знаків нам буде треба задіяти для зберігання розширення по кожній x вісей.

Далі переходимо до опису кожної окремої сторінки.

Першим параметром сторінки є її номер – вкажемо з англійського перекладу PageNumber=XXX, де XXX номер сторінки. Треба зазначити, що номер сторінки не може перевищувати загальну кількість сторінок, інакше файл вважається пошкодженим.

Наступний параметр фон сторінки – його пропонується зробити з переводом з англійської BackgroundType=BType при цьому параметр BType може приймати наступні значення:

- None;
- Color;
- Gradiet;
- Rastr.

Другим параметром після роздільника «||»вказуються параметри обраного типу.

Для None – роздільник та другий параметр відсутній.

Для Color вказується ColorCode=RRGGBB, де RRGGBB код колыру у 24бітному форматі RGB.

Для Gradiet вказується початкова точка з початковим кольором та кінцева точка з кінцевим кольором:

- BeginX=XX – початкова координата по горизонталі;
- BeginY=YY – початкова координата по вертикалі;
- BeginColor=RRGGBB – початковий колір;
- EndX=XX – кінцева координата по горизонталі;
- EndY=YY – кінцева координата по вертикалі;
- EndColor=RRGGBB – кінцевий колір.

Всі ці параметри розділяються роздільником «||».

Для Rastr вказуємо розміщення растрового зображенняна диску комп’ютера, його яскравість та контраст відносно початкового файлу:

- ImgLocation = “розташування файлу”;
- Brightness = 90%;
- Contrast = 120%.

Далі в якості параметрів записуються векторні елементи за параметрами для сторінки поки програма не знайде параметр «PageNumber», що буде позначати наступну сторінку.

Для простоти розуміння структури файлу було прийнято рішення вказувати назви елементів та їх параметри так, як вони вказуються у середі розробки Delphi у тих випадках, коли це можливо. Таблиця векторних елементів з параметрами вказана (див. табл.3.1).

Таблиця 3.1 – Перелік векторних елементів з параметрами, що зберігаються у розробленій структурі файлу.

Назва	Позначення	Параметри	Ім’я
Пряма лінія	Line	(X1, Y1, X2, Y2)	LN_P_N
Стрілочка	Arrow	(Type, X1, Y1, X2, Y2)	AR_P_N
Трикутник	Triangle	(X1, Y1, X2, Y2, X3, Y3)	TR_P_N
Чотирикутник	quadrangle	(X1, Y1, X2, Y2, X3, Y3, X4, Y4)	QU_P_N
Еліпс	Ellipse	(Xc, Yc, Xa, Ya, Xb, Yb)	EL_P_N
Сектор	Sector	(Xc, Yc, Xa, Ya, Xb, Yb, Ang1, Ang2)	SC_P_N
Крива лінія	CurveLine	N, array [1..N] of Sector	CL_P_N
Багатокутник	Polygon	N, array [1..N] of Point	PL_P_N
Текст	TextXY	(X, Y, Font, Size, Style)	TX_P_N

Крім того кожному елементу присвоюється власне ім’я, що складається з двох літер, номеру сторінки та після підкреслення номеру такого елементу на сторінці.

Додатково для кожного векторного елементу додається параметр, що характеризує його колір та товщину лінії:

- FigureColor=RRGGBB – колір фігури;
- Thikness=3 – товщина лінії фігури в точках.

Якщо на сторінці присутня анімація після вказання всіх елементів вказується ключове слово Animation.

Далі для кожного елемента, що має анімацію вказується його ім’я, тип анімації, час програвання анімації, номер анімації по порядкуна листі.

Типи анімації та їх позначення у файлі:

- масштабування – Scaling;
- рух по заданій траєкторії –Traectory;
- обертання – Rotation;
- поява через заданий час – Appereance;

− зникнення через заданий час – Disappereance.

Наприклад: EL_2_1||Scaling||2.3||5

3.2 Написання коду програми по розміщенню елементів в робочому полі програми

Коли програму слід вивести будь-яку інформацію на екран дисплея, операційна система Windows надає їй в своє розпорядження віконно-орієнтовану графіком. Це означає, що кожна форма (як, втім, і ряд інших об'єктів) розглядається як поверхня для малювання – полотно (canvas).

Коли в формі проводиться побудова малюнка, то виведене зображення обрізається по її краях, тобто виводяться тільки ті його частини, які поміщаються на полотно вікна. Такий підхід гарантує, що висновок однієї програми не зачепить клієнтської області інших додатків, на чому, власне і ґрунтується віконна система Windows.

За точку відліку координат у вікнах приймається лівий верхній кут його клієнтської області, обмеженою рамкою (тобто якраз ті габарити, що відображаються властивостями ClientHeight і ClientWidth). Для інших елементів, що мають полотно для малювання, подібної рамки може і не бути – в такому випадку рамками такого віртуального полотна виступають межі об'єкта.

Для багатьох елементів інтерфейсу в Windows додатках використовується компоненти, засновані на класі TWinControl, що є прямим спадкоємцем вже розглянутого нами класу TControl. Але безліч інших візуальних компонент походять від іншого його спадкоємця – класу TGraphicControl, що є більш простим і мають таку властивість, як Canvas, за допомогою якого можна в буквальному сенсі малювати в Windows. Об'єкт Canvas інкапсулює в собі взаємодію з Windows GDI (Graphic Device Interface – інтерфейс графічного пристрою).

У ряду об'єктів з бібліотеки візуальних компонент Delphiє властивість Canvas, яка надає простий шлях для малювання на них. Ці об'єкти – TBitmap, TComboBox, TDBComboBox, TDBGrid, TDBListBox, TDirectoryListBox, TDrawGrid, TFileListBox, TForm, TImage, TListBox, TOutline, TPaintBox, TPrinter, TStringGrid. Canvas є в свою чергу об'єктом, що об'єднує в собі поле для малювання, олівець (Pen), кисть (Brush) і шрифт (Font). Canvas володіє також рядом графічних методів: Draw, TextOut, Arc, Rectangle і ін. Ми використовуємо в нашій програмі об’єкт TImage з ім’ям Image_ForDraw , який було описанопри розробці інтерфейсу.

У таблиці 3.2 наведені основні властивості та методи компонентаCanvas для малювання

Нашою програмою крім анімації передбачено малювання 8 основних елементів, додавання тексту з різним кутом та орієнтацією та вставка растрового зображення.

Розглянемо окремо створені в процесі виконання дипломного проекту для цього методи з описом застосованих інструментів.

Першим найпростішим елементом є пряма лінія, яку користувач може намалювати на полі. Зрозуміло, що для цього краще всього використовувати метод LineTo.

Користувач для початку малювання прямої лінії обирає меню з основними фігурами та піктограму з прямою (див. рис.3.7)

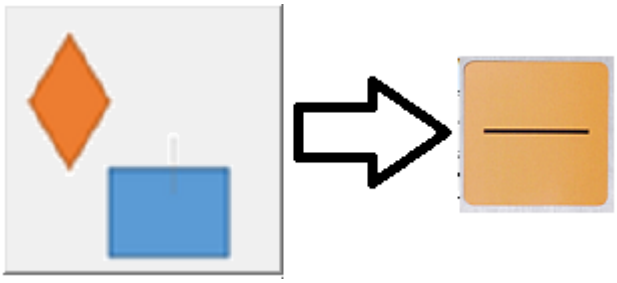

Рисунок 3.7 – Вибір прямої лінії для малювання

Далі на робочій області користувач курсором вказує точку початку лінії та натискає ліву кнопку миші, при цьому при переміщенні курсору до нових

координат точки малюється пряма лінія. Якщо користувач відпускає ліву кнопку миші лінія залишається на полі для малювання. Якщо під час малювання користувач натискає праву кнопку миші намальована лінія зникає.

Таблиця3.2 – Властивості и методи TCanvas

Властивість (метод)	Тип значень або параметри	Опис
Pixels	Матриця TColor	Надає доступ к будь-якому пікселю холсту, щоб дізнатися або змінити його колір
Pen	TPen	Властивості пера для креслення ліній
Brush	TBrush	Властивостіпензлика для заповнення внутрішніх областей фігур
Font	TFont	Властивості шрифта для виводу тексту
MoveTo	X, Y: Integer	Встановлюєпоточну позицію пера
LineTo	X, Y: Integer	Проводитьлінію відпоточної позиціїдовказаної
TextOut	X, Y: Integer; const Text: string	Виводить заданий текст, починаючизвказаних координат
Rectangle	X1, Y1, X2, Y2: Integer або Rect: TRect	Малює прямокутник вказаних розмірів. Колір рамки визначаєтьсяпоточним значеннямвластивсті Pen, а колір заливки – властивістю Brush
Ellipse	X1, Y1, X2, Y2: Integer або Rect: TRect	Малюєеліпс, вписанийу прямокутниквказаних розмірів. Колір рамки визначається поточним значеннямвластивості Pen, а колір заливки – властивістю Brush
Polygon	Points: array of TPoint	Малює багатокутник по вказаним вершинам
PolyLine	Points: array of TPoint	Малює ламаную линию, що з’єднана вказаними точками
Draw	X, Y: Integer; Graphic: TGraphic	Виводить графічне зображення, починаючивідвказаних координат (відносно лівого верхнього кута)

Ці дії виконуються наступним чином:

a) Після вибору прямої лінії користувачем – глобальна змінна FigureSelected:=true та FigureType:=1, що означає, що користувач обрав фігуру для малювання та це фігура першого типу;

b) Чекаємо на подію OnMouseDown компонента Image_ForDraw;

c) В обробнику події перевіряємо щоб була натиснена саме ліва кнопка миші Button=mbLeft оскільки та сама подія буде викликати припинення малювання лінії коли вона буде викликатись правою кнопкою;

d) Запам’ятовуємо координати миші (LineBeginX, LineBeginY), встановлюємо позицію пера методом MoveTo в ті координати та встановлюємо глобальну змінну DrawLineBegin:=true;

e) Якщо не відбулась подія OnMouseUp чи повторно не відбулась подія OnMouseDownз параметром Button=mbRight – обробляємо подію OnMouseMove в якій з початкових координат (LineBeginX, LineBeginY) встановивши позицію пера методом MoveTo, виконуємо метод LineTo в поточні координати миші при цьому видаляючі лінію, що була намальована в попередніх координатах;

f) Якщо відбулась подія OnMouseDown з параметром Button=mbRight, або було обрано іншу фігуру для малювання, або проведена інша дія з якимось меню– видаляємо лінію та робимо скид координат початку в 0. Відповідно занулюємо або відповідно переписуємо усі змінні/ознаки.

Наступним елементом є малювання лінії зі стрілочкою. Користувач обирає меню з основними фігурами та піктограму з відповідною лінією (див. рис.3.8)

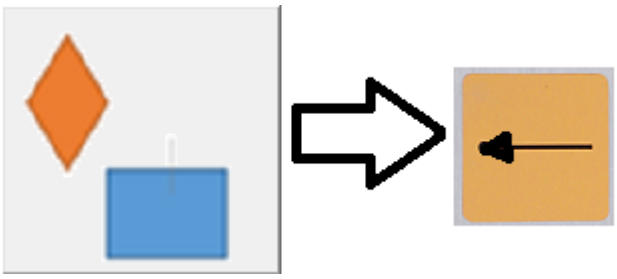

Рисунок 3.8 – Вибір лінії зі стрілочкою для малювання

З точки зору векторного малювання лінія зі стрілочкою це пряма лінія на одному чи на обох краях присутній трикутник, а отже логіка роботи буде ідентична як і у випадку просто з прямою лінією з різницею в тому, що на початку і в кінці цієї лінії, або тільки в кінці (якщо з однією стрілочкою) буде промальовано трикутник за допомогою метода Polygon в якості параметрів у якого буде масив з трьох точок.

Наступним графічним елементом є трикутник. Для його вибору користувач обирає меню з основними фігурами та піктограму з трикутником (див. рис.3.9)

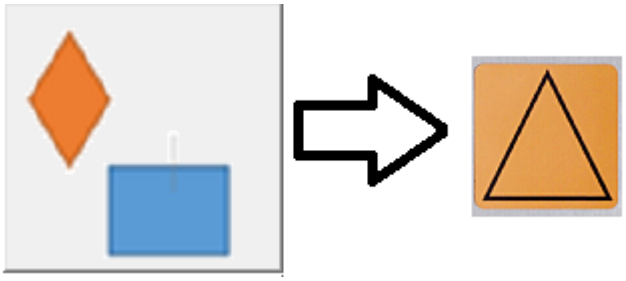

Рисунок 3.9 – Вибір трикутника для малювання

Логіка роботи розробленої програми при малюванні трикутника дещо схожа на малювання прямої лінії (обробляємо події OnMouseDown, OnMouseUp, OmMouseMove). Відмінність полягає в тому що при виборі першої точки користувач натискає ліву кнопку, при виборі другої точки навпаки відпускає ліву кнопку і при виборі третьої точки знов її натискає на деякий час. Якщо під час виконання якоїсь з цих дій була натиснена права кнопка дії відкатуються на одну назад. Коли отримані всі три точки вершин будується полігон методом Polygon в якості параметрів у якого буде масив з трьох вказаних вершин.

```
procedure Triangle(pol: array[1..3] of TPoint);
begin
pol[1].x :=10;
pol[1] .y :=50;
```

```
pol[2]-x := 40;
pol[2].y := 10;
pol[3].x := 70;
pol[3].y : = 50;
Forml.Canvas.Polygon(pol);
end;
```

Для вибору можливості малювання чотирикутника (ромба, прямокутника, квадрата, трапеції – в залежності від параметрів сторін та кутів) користувач обирає меню з основними фігурами та піктограму з прямокутником (див. рис.3.10)

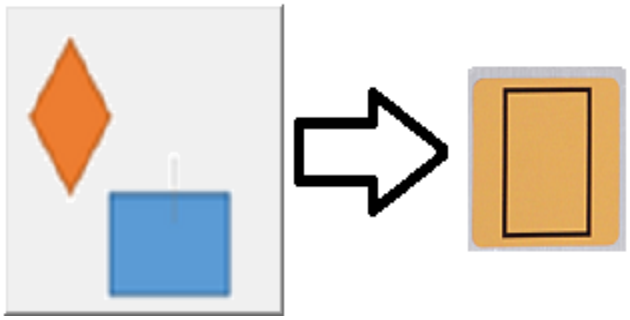

Рисунок 3.10 – Вибір чотирикутника для малювання

Логіка роботи розробленої програми при малюванні чотирикутника повторює алгоритм малювання трикутника (обробляємо події OnMouseDown, OnMouseUp, OmMouseMove). Відмінність полягає в тому, що при виборі четвертої точки відпускає ліву кнопку. Якщо під час виконання якоїсь з цих дій була натиснена права кнопка дії відкатуються на одну назад. Коли отримані всі чотири точки вершин будується полігон методом Polygon в якості параметрів у якого буде масив з чотирьох вказаних вершин.

Для вибору можливості малювання еліпса (еліпса, кола як приватного випадку – в залежності від параметрів сторін) користувач обирає меню з основними фігурами та піктограму з еліпсом (рис.3.11)

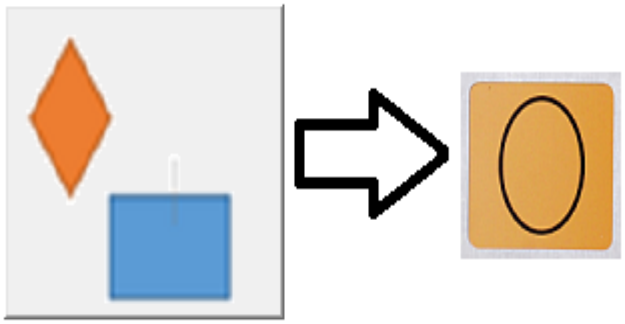

Рисунок 3.11 – Вибір еліпса для малювання

Основним елементом програмної реалізації малювання еліпса є звісно процедура Ellipse. Однак треба зазначити, що у нас в програмі параметри еліпса задаються на прямокутником в який цей еліпс вписано, а трьома координатами:

- координати центру;
- координати вершини великої напіввісі;
- координати вершини малої напіввісі.

Це зроблено для зручності роботи користувача. Коли користувач обирає малювання еліпсу він одразу вказує на полі координату центру розташування та натискає ліву кнопку, потім вказує координату вершини великої напіввісі натискає ліву кнопку і на останок обирає координату вершини малої напіввісі.

Після введення користувачем цих даних програма перераховує параметри, щоб вказати їх у вигляді еліпса, що вписаний у прямокутник – оскільки це стандартна функція Delphi.

Для вибору можливості малювання сектору еліпса (сектору еліпса, сектору кола як приватного випадку – в залежності від параметрів сторін) користувач обирає меню з основними фігурами та піктограму з сектором еліпса (рис.3.12)

Рисунок 3.12 – Вибір дуги або частини еліпса для малювання

Креслення дуги виконує метод Arc, інструкція виклику якого взагальному вигляді виглядає таким чином:

Canvas .Arc (xl,yl,x2,y2, x3, y3,x4,y4)

де : xl, yl, x2, y2 – параметри, що визначають еліпс (коло), частиною якого є викреслюють дуга;

хз, уз – параметри, що визначають початкову точку дуги;

х4, у4 – параметри, що визначають кінцеву точку дуги.

Початкова (кінцева) точка – це точка перетину кордону еліпса і прямої, проведеної з центру еліпса в точку з координатами хз і уз (х4, у4). Дуга викреслюється проти годинникової стрілки від початкової точки до кінцевої.

Колір, товщина і стиль лінії, якою викреслюється дуга, визначаються значеннями властивості Pen поверхні (canvas), на яку виконується висновок.

Для вибору можливості малювання кривої лінії (лінії, що складається з багатьох секторів еліпсів друга координата яких задається у вигляді точки траєкторії) користувач обирає меню з основними фігурами та піктограму з кривою лінією (рис.3.13).

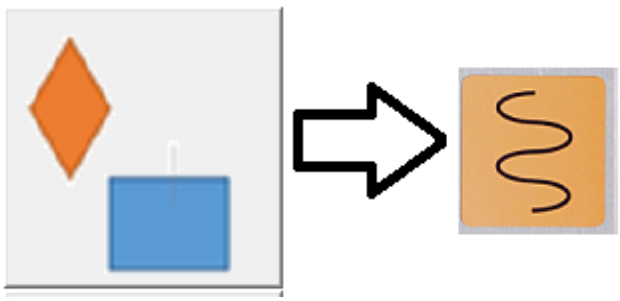

Рисунок 3.13 – Вибір складної кривої лінії для малювання

Для малювання кривої Безьє поділяємо інтервал між P1 і P2 на кілька відрізків (їх кількість впливає на точність відтворення кривої, 3 – 4 точки цілком достатньо), потім в циклі створюємо масив точок, використовуємо описану вище процедуру з параметром t від 0 до 1 і малюємо даний масив точок, використовуючи функцію polyline ().

```
PBezierPoint = ^TBezierPoint;
TBezierPoint = record
  X, Y: double;   // основнойузел
Xl, Yl: double; // левая контрольная точка
  Xr, Yr: double; // правая контрольная точка
end;
```

P1 та P2 - дві точки TBezierPoint, розташовані між 0 и 1:
коли t=0 X=P1.X, Y=P1.Y; когда t=1 X=P2.X, Y=P2.Y;

```
procedure BezierValue(P1, P2: TBezierPoint; t: double; var X, Y: double);
var
  t_sq, t_cb, r1, r2, r3, r4: double;
begin
  t_sq := t * t;
  t_cb := t * t_sq;
r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.X;
r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Xr;
r3 := (3 * t_sq - 3 * t_cb) * P2.Xl;
r4 := (t_cb) * P2.X;
X := r1 + r2 + r3 + r4;
r1 := (1 - 3 * t + 3 * t_sq - t_cb) * P1.Y;
r2 := (3 * t - 6 * t_sq + 3 * t_cb) * P1.Yr;
r3 := (3 * t_sq - 3 * t_cb) * P2.Yl;
r4 := (t_cb) * P2.Y;
Y := r1 + r2 + r3 + r4;
```

end;

Для вибору можливості малювання багатокутника (замкненої фігури, що має більше п’яти вершин) користувач обирає меню з основними фігурами та піктограму з багатокутником (рис.3.14).

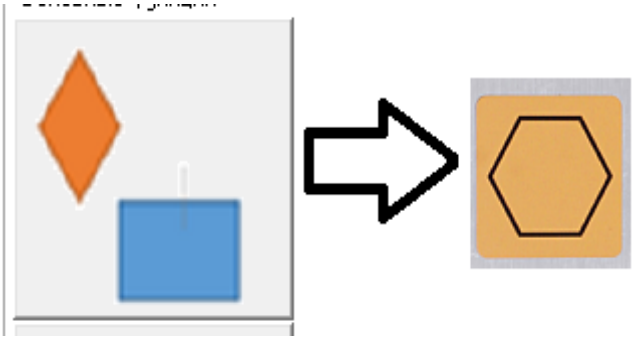

Рисунок 3.14 – Вибір багатокутника для малювання

Метод polygon викреслює багатокутник. Як параметр метод отримує масив типу TPoint. Кожен елемент масиву представляє собою запис, поля (x, y) якої містять координати однієї вершини багатокутника. Метод Polygon викреслює багатокутник, послідовно поєднуючи прямими лініями точки, координати яких знаходяться в масиві, першу з другою, другу з третьою, третю з четвертою і т. д. Потім з'єднуються остання і перша точки.

Колір і стиль кордону багатокутника визначаються значеннями властивості Реп, а Івет і стиль заливки області, обмеженої лінією кордону, - значеннями властивості Brush, причому область зафарбовується з використанням поточного кольору і стилю кисті.

У таблиці 3.1 біло наведено перелік векторних елементів з вказанням їх типів та параметрів при зберіганні у файлі. Оскільки безперервно зберігання в файл робити досить складно, особливо при проведені користувачем змін – проміжні стани зберігаються у робочих елементах типу record.

Програмою передбачено автозбереження даних раз у визначений налаштуваннями час. Для того, щоб авто збереження не займало багато часу

воно проводиться у форматі JSONy вигляді одного довгого рядка в файлі Autosave.fra, що знаходиться у основній директорії програми.

3.3 Вибір типів анімації та методів їх відображення

В програмі пропонується наступний перелік анімаційних ефектів:

- масштабування;
- рух по заданій траєкторії;
- обертання;
- поява через заданий час;
- зникнення через заданий час.

Масштабування представляє собою зміну розмірів фігури без зміни форми.

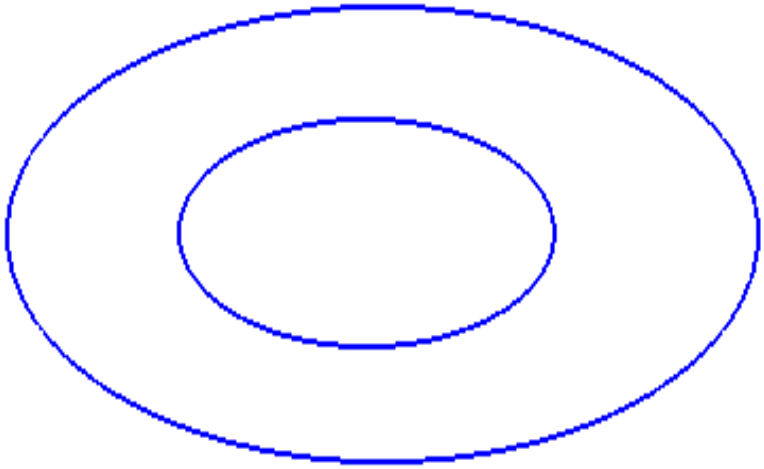

Рисунок 3.15 – Приклад анімації «масштабування» на прикладі еліпсу

Параметрами анімації типу «масштабування» є:

- початковий коефіцієнт відносно параметрів фігури;
- кінцевий коефіцієнт відносно параметрів фігури;
- кількість часу за який буде проводитися масштабування.

Рух по заданій траєкторії представляє собою рух заданої фігури по вказаних користувачем точках (рис.3.16).

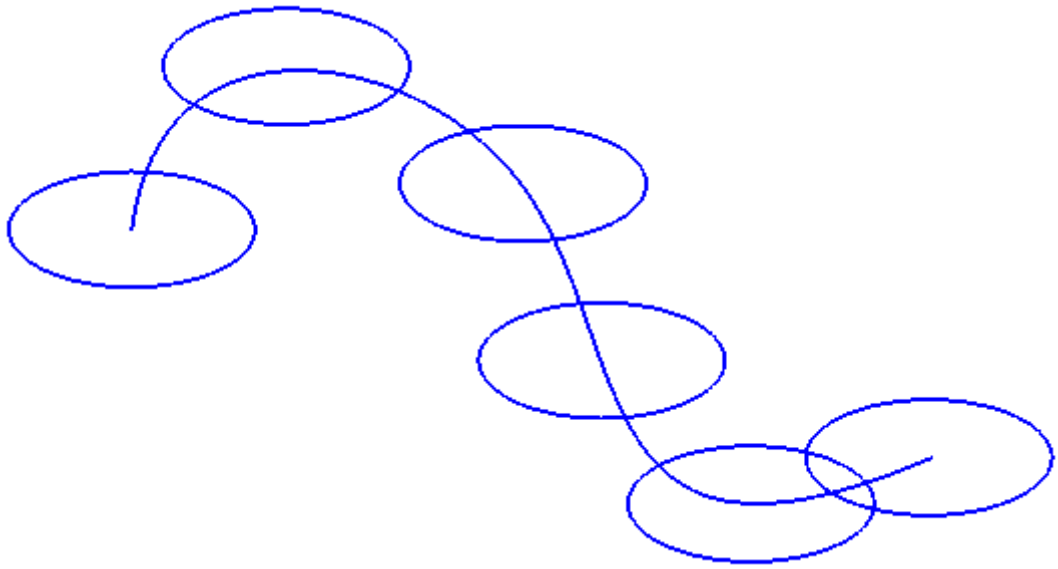

Рисунок 3.16 – Приклад анімації «рух по заданій траєкторії» на прикладі еліпсу

Параметрами анімації типу «рух по заданій траєкторії» є:

− масив точок траєкторії руху;

− кількість часу за який буде проводитися рух по вказаним точкам.

Обертання представляє собою рух заданої фігури відносно свого центру на визначений градус (рис.3.17).

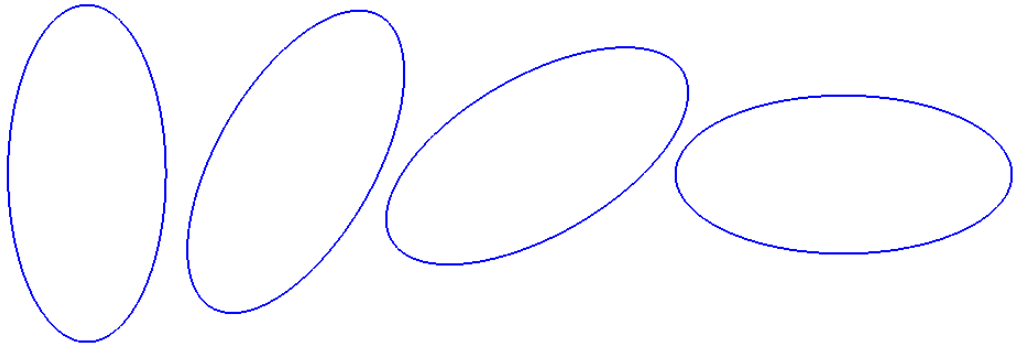

Рисунок 3.17 – Приклад анімації «обертання» на прикладі еліпсу

Параметрами анімації типу «Обертання» є:

− кут на який треба повернути фігуру;

− кількість часу за який буде проводитися поворот.

«Поява через заданий час» представляє поступове підвищення контрастності заданої фігури від нуля до встановленої при її створенні.

Параметрами анімації типу «Поява через заданий час» є:

− кількість часу за який буде підвищуватись контрастність.

«Зникнення через заданий час» представляє поступове зниженняконтрастності заданої фігури до нуля від встановленої при її створенні.

Параметрами анімації типу «Зникнення через заданий час» є:

− кількість часу за який буде підвищуватись контрастність.

3.4 Написання коду програми для виконання анімаційних ефектів

Для програмної реалізації кожного типу анімації були застосовані різні програмні ефекти.

Анімація «Масштабування» в програмі реалізується через перемалювання фігури або через збереження окремої фігури в форматі PNG(оскільки цей формат дозволяє залишати прозорим фон) з використанням функції розтягування растрового зображення.

```
function CalcPictureRect(PlaceRect : TRect; PictureWidth, PictureHeight : Integer): TRect;
var
aCoeff : Double;
begin
if PlaceRect.Bottom - PlaceRect.Top <= 0 then
Result := Rect (0, 0, 1, 1)
else begin
```

```
        { scale image }
        aCoeff := PictureWidth / PictureHeight;
        Result := Rect (0, 0,
            Min ((PlaceRect.Right - PlaceRect.Left), Trunc ((PlaceRect.Bottom -
    PlaceRect.Top) * aCoeff)),
            Min ((PlaceRect.Bottom - PlaceRect.Top), Trunc ((PlaceRect.Right -
    PlaceRect.Left) / aCoeff)));

        { centering Result Rect }
          OffsetRect (Result, ((PlaceRect.Right - PlaceRect.Left) - Result.Right) div 2,
           ((PlaceRect.Bottom - PlaceRect.Top) - Result.Bottom) div 2);
        end;
        end;

        procedure TForm1.FormResize(Sender: TObject);
        begin
        Image1.BoundsRect := CalcPictureRect (Image1.Parent.ClientRect,
    Image1.Picture.Width, Image1.Picture.Height);
        end;
```

Анімація «рух по заданій траєкторії» в програмі реалізується через перемалювання фігури з центром на прямих, які з'єднують точки траєкторії вказані користувачем.

```
FillRect(clipRect);
for i:=0 to число_точек_траектории-1 do
begin
  Canvas.Ellipse(x[i]-2, y[i]-2, x[i]+3, y[i+3];
Application.ProcessMessages;
Sleep(pause);
end;

procedure TForm1.MoveArray;
var
```

```
  i: integer;
begin
  imageDraw.Canvas.Brush.Color := clWhite;
  imageDraw.Canvas.Pen.Color := clGray;
  ImageDraw.Canvas.Rectangle(0, 0, imageDraw.Width, ImageDraw.Height);
  for i := 1 to High(allImage) do
   allImage[1].Draw(ImageDraw.Canvas);
  allImage[2].Draw(ImageDraw.Canvas);
end;
```

Анімація «обертання» виконується в програмі завдяки використанню функції повороту зазаданий кут.

```
procedurerotateright(bitmap : timage);
var firstc, lastc, c, r : integer;

procedurefixpixels(c,r : integer);
var savepix, savepix2 : tcolor;
i, newc, newr : integer;
begin
savepix := bitmap.canvas.pixels[c,r];
for i := 1to4dobegin
newc := bitmap.height-r+1;
newr := c;
savepix2 := bitmap.canvas.pixels[newc,newr];
bitmap.canvas.pixels[newc,newr] := savepix;
savepix := savepix2;
c := newc;
r := newr;
end;
end;

begin
if bitmap.width <> bitmap.height thenexit;
```

```
bitmap.visible := false;
with bitmap.canvas dobegin
firstc := 0;
lastc := bitmap.width;
for r := 0to bitmap.height div2dobegin
for c := firstc to lastc dobegin
fixpixels(c,r);
end;
inc(firstc);
dec(lastc);
end;
end;
bitmap.visible := true;
end;
```

Анімація «Зникнення череззаданий час» та «Поява через заданий час» виконується в програмі завдяки процедурі, що змінює контраст до значення вказаного у якості вхідного параметру.

```
procedureContrast(Bitmap: TBitmap; Value: Integer; Local: Boolean);

  functionBLimit(B: Integer): Byte;
  begin
    ifB < 0then
      Result := 0
    elseifB > 255then
      Result := 255
    else
      Result := B;
  end;

var
  Dest: pRGBTriple;
  x, y, mr, mg, mb,
```

```
  W, H, tr, tg, tb: Integer;
 vd: Double;

begin
 ifValue = 0then
  Exit;
 W := Bitmap.Width - 1;
 H := Bitmap.Height - 1;
 ifLocal then
 begin
  mR := 128;
  mG := 128;
  mB := 128;
 end
 else
 begin
  tr := 0;
  tg := 0;
  tb := 0;
  fory := 0toH do
  begin
   Dest := Bitmap.ScanLine[y];
   forx := 0toW do
   begin
    withDest^ do
    begin
     Inc(tb, rgbtBlue);
     Inc(tg, rgbtGreen);
     Inc(tr, rgbtRed);
    end;
    Inc(Dest);
   end;
  end;
  mB := Trunc(tb / (W * H));
  mG := Trunc(tg / (W * H));
```

```
    mR := Trunc(tr / (W * H));
  end;
  ifValue > 0then
    vd := 1+ (Value / 10)
  else
    vd := 1- (Sqrt(-Value) / 10);
  fory := 0toH do
  begin
    Dest := Bitmap.ScanLine[y];
    forx := 0toW do
    begin
      withDest^ do
      begin
        rgbtBlue := BLimit(mB + Trunc((rgbtBlue - mB) * vd));
        rgbtGreen := BLimit(mG + Trunc((rgbtGreen - mG) * vd));
        rgbtRed := BLimit(mR + Trunc((rgbtRed - mR) * vd));
      end;
      Inc(Dest);
    end;
  end;
end;
```

Оскільки всі типи анімації потребують використання таймера затримки для промальовування етапів в програмі використовується таймер для анімації та для його обробки написана відповідна функція.

```
procedure TForm1.TimerMoveTimer(Sender: TObject);
  var
  x, y, i, v, a1,g,o,k: integer;
  a: real;
 image,image2:TMyImage;
  begin
    if y1 < ImageDraw.Height+1 then
        begin
```

```
        a:=25 * pi / 180;
        v:=100;

        x1:=x1+1;

        y1:=x1-Round(x1*tan(a)-10*sqr(x1)/(2*sqr(v)*sqr(cos(a))));
       image := Tadro.Create(x,y);
        image.Draw(imageDraw.Canvas);
        k :=length(allImage);
        setLength(allImage,k+1);
        allImage[k] := image;
        MoveArray;
        image2 := TAdro.Create(x1,y1);
        image2.Draw(imageDraw.Canvas);

end;
 end;
```

Отже, було проаналізовано існуючи формати векторного зберігання зображень. Аналіз показав, що існуючі формати або закриті та використовуються окремими програмними продуктами без розголошення їх структури, або складні для запису, або не дозволяють зберігати анімаційні ефекти і відповідно жоден з них нам не підходить. Тому у розділі було розроблено власний формат зберігання фреймових структур.

Крім того в розділі було описано написання програмного коду для промальовування різних типів фреймових структур з використання вбудованих в середу розробки процедур та з написанням власних.

В розділі були запропоновані типи анімації, що будуть використовуватись в програмі та описано програмний код реалізації цих анімаційних ефектів.

ВИСНОВКИ

У висновку варто сказати, що середовище фреймового представлення в наш час є досить актуальним, так як від правильної подачі матеріалу залежить успіх та ефективність навчального процесу. Воно допоможе полегшити підготовку студентам, так само буде зручним для викладу матеріалу для викладачів.

В даній дипломній роботі було розглянуто фреймовий метод в освітньому процесі, сучасні існуючі інструменти та актуальність проблем, які існують на цей час. Було визначено сучасні методи подачі навчального матеріалу та розглянуто існуючі проблеми для застосування фреймового методу. Після чого було обрано напрямок оптимізації та мова програмування, інструментами якої керувався автор. Як результат роботи розроблено програмне середовище, що відповідає сучасним стандартам, використовується в навчальних закладах, та вирішує поставлені задачі.

Плюси від впровадження даної технології зрозумілі всім. Адже це економія часу для викладача, або студента, середовище може створювати фреймові об'єкти, які можуть бути динамічними при показі, так само користувач може прописати потрібний текст і вибрати потрібну форму для фрейма. Даний додаток також володіє простотою в управлінні, зрозумілим і звичним оку інтерфейсом.

У майбутньому планується розвиток додатку та розширення функціоналу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Соколов Д. П. Фрейм як формат репрезентації знань про одиниці політичного вокабуляру. Дніпро : Мовознавство, 2015. 146с.

2. Хоменська І. Формування основних сегментів терміносистеми когнітивної лінгвістики. Львів : Проблеми української термінології, 2014. 169 с.

3. Онищук М. Фрейм як одиниця репрезентації грамотичного значення в перекладі художнього тексту. Південний архів. Київ : Перекладознавство, 2017. 171 с.

4. Щербакова Е. Е., Мухина Т. Г., Плешков А. В. Фрейм технология как условие развития креативности студентов. Москва : Современные проблемы науки и образования, 2016.

5. Колодочка Т. Н. Фреймовая технология в среднем профессианальном образовании. Санкт-Петербург : Школьные технологии, 2004. 30 с.

6. Аналог фреймового методу безкоштовний. UMLet. URL : http://freeanalogs.ru/UMLet (дата звернення: 06.02.2016).

7. Аналог фреймового методу безкоштовний. yEd. URL : https://www.yworks.com/ (дата звернення: 15.11.2019).

8. Аналог фреймового методу безкоштовний. DiagramDesigner. URL : http://logicnet.dk/DiagramDesigner/ (дата звернення: 01.11.2019).

9. Малязина М. А. Фреймы как инструмент оптимизации образовательного процесса. Санкт-Петербург : Педагогика, 2018. 25 с.

10. Соколова Е. Е., Федорова С. И. Теоритические основы и реализация фреймового подхода в обучении. Ульяновск : УлГУ, 2008. 200 с.

11. Соколова Е. Е., Гурина Р. В. Фреймовое представление знаний. Москва : Народное образование школьных технологий, 2005. 51 с.

ДОДАТОК А

Модуль програми

```
unit u_bezier;

interface

uses Windows, Graphics, SysUtils;

type TArrayPoint = arrayof TPoint; //массивточек

const num_slices: integer = 20; //число отрезков между двумя точками
  krivizna: integer = 30; //кривизна кривой (длина плеча направляющей)

procedure DrawBezier(acanv: TCanvas; var ArrPoint: TArrayPoint);

implementation
uses unit1;

type
  TBezierPoint = record//точкаБезье
x, y: integer; //основной узел
    xl, yl, //левая контрольная точка
      xr, yr: single; //правая контрольная точка
end;
  TArrayBezierPoint = arrayof TBezierPoint; //массивточекБезье

const grad_to_rad = pi / 180; //переводградусовврадианы
```

```
  rad_to_grad = 180 / pi; //переводрадианвградусы
rad_90 = 90 * grad_to_rad; //90 градусов в радианах
  rad_180 = 180 * grad_to_rad; //180 градусов в радианах
  rad_270 = 270 * grad_to_rad; //270 градусов в радианах
  rad_360 = 360 * grad_to_rad; //360 градусов в радианах

var Canvas: TCanvas; //рабочий холст, на котором происходит рисование


//определить угол в радианах между точкой и положительным направлением оси x

function GetAngle(dx, dy: single): single;
begin
if dx = 0 thenbegin
if dy = 0 then Result := 0
elseif dy < 0 then Result := rad_270
else Result := rad_90;
exit
end;
Result := arctan(abs(dy) / abs(dx));
if dy < 0 then
if dx < 0 then Result := rad_180 + Result
else Result := rad_360 - Result
else
if dx < 0 then Result := rad_180 - Result
end;


//определить направляющие линии к точке p

procedure GetCooPerpendikular(a, o, b: TPoint; var p: TBezierPoint);
var alfa, beta, gamma, dx, dy, angle_napr: single;
  l1, l2: single;
begin
dx := a.x - o.x; dy := a.y - o.y;
```

```
alfa := GetAngle(dx, dy);
l1 := sqrt(dx * dx + dy * dy) * (krivizna / 100); //растояние oa
dx := b.x - o.x; dy := b.y - o.y;
beta := GetAngle(dx, dy);
l2 := sqrt(dx * dx + dy * dy) * (krivizna / 100); //растояние ob
gamma := (alfa + beta) / 2; //биссектрисаугла aob

if alfa > beta then angle_napr := gamma + rad_90
else angle_napr := gamma - rad_90;

p.xl := o.x + l1 * cos(angle_napr);
p.yl := o.y + l1 * sin(angle_napr);
p.xr := o.x + l2 * cos(angle_napr + rad_180);
p.yr := o.y + l2 * sin(angle_napr + rad_180)
end;


//вычислить координаты точки, лежащей на участке кривой между
//двумя точками Безье в пределах от 0 до 1

procedure BezierValue(P1, P2: TBezierPoint; t: single; var X, Y: integer);
var t_sq, t_cb, r1, r2, r3, r4: single;
begin
  t_sq := t * t;
  t_cb := t * t_sq;
r1 := (1 - 3 * t + 3 * t_sq - t_cb);
r2 := (3 * t - 6 * t_sq + 3 * t_cb);
r3 := (3 * t_sq - 3 * t_cb);
r4 := (t_cb);
X := round(r1 * p1.x + r2 * p1.xr + r3 * p2.xl + r4 * p2.x);
Y := round(r1 * p1.y + r2 * p1.yr + r3 * p2.yl + r4 * p2.y)
end;


//рисуй участок кривой между двумя точками Безье
```

```
procedure DrawSlice(p1, p2: TBezierPoint);
var i: integer;
x, y: integer;
r1, r2: TRect;
begin
//  если убрать комментарий, то количество отрезков между соседними
//  точками будет расчитываться исходя из растояния между ними
//  num_slices:=trunc(sqrt(sqr(p1.x-p2.x)+sqr(p1.y-p2.y)));
Canvas.MoveTo(p1.x, p1.y);
for i := 1 to num_slices - 1 dobegin
BezierValue(p1, p2, i / num_slices, x, y);
   Canvas.LineTo(x, y)
end;
Canvas.LineTo(p2.x, p2.y)
end;


//рисуй кривую на холсте acanv по точкам массива ArrPoint

procedure DrawBezier(acanv: TCanvas; var ArrPoint: TArrayPoint);
var ArrBezPoint: TArrayBezierPoint;
i, num_point: integer;
a, o, b: TPoint;
begin
Canvas := acanv;
  num_point := high(ArrPoint) + 1;
SetLength(ArrBezPoint, num_point);
for i := 0 to num_point - 1 dobegin
   ArrBezPoint[i].x := ArrPoint[i].x;
   ArrBezPoint[i].y := ArrPoint[i].y;
end;
ArrBezPoint[0].xr := ArrPoint[0].x;
ArrBezPoint[0].yr := ArrPoint[0].y;
ArrBezPoint[0].xl := ArrPoint[0].x;
```

```
ArrBezPoint[0].yl := ArrPoint[0].y;
for i := 1 to num_point - 2 dobegin
a := ArrPoint[i - 1];
o := ArrPoint[i];
b := ArrPoint[i + 1];
GetCooPerpendikular(a, o, b, ArrBezPoint[i])
end;
ArrBezPoint[num_point - 1].xr := ArrPoint[num_point - 1].x;
ArrBezPoint[num_point - 1].yr := ArrPoint[num_point - 1].y;
ArrBezPoint[num_point - 1].xl := ArrPoint[num_point - 1].x;
ArrBezPoint[num_point - 1].yl := ArrPoint[num_point - 1].y;

for i := 1 to num_point - 1 do
DrawSlice(ArrBezPoint[i - 1], ArrBezPoint[i])
end;

end.
```

ДОДАТОК Б

Основна процедура анімації «обертання»

```
procedure RotateBitmap(Bitmap: TBitmap; Angle: Double; BackColor: TColor);
type
  TRGB = record
    B, G, R: Byte;
end;
  pRGB = ^TRGB;
  pByteArray = ^TByteArray;
  TByteArray = array[0..32767] of Byte;
  TRectList = array[1..4] of TPoint;

var
  x, y, W, H, v1, v2: Integer;
  Dest, Src: pRGB;
  VertArray: arrayof pByteArray;
  Bmp: TBitmap;

procedure SinCos(AngleRad: Double; var ASin, ACos: Double);
begin
    ASin := Sin(AngleRad);
    ACos := Cos(AngleRad);
end;

function RotateRect(const Rect: TRect; const Center: TPoint; Angle: Double):
    TRectList;
var
    DX, DY: Integer;
    SinAng, CosAng: Double;
function RotPoint(PX, PY: Integer): TPoint;
```

```
begin
    DX := PX - Center.x;
    DY := PY - Center.y;
    Result.x := Center.x + Round(DX * CosAng - DY * SinAng);
    Result.y := Center.y + Round(DX * SinAng + DY * CosAng);
end;
begin
  SinCos(Angle * (Pi / 180), SinAng, CosAng);
  Result[1] := RotPoint(Rect.Left, Rect.Top);
  Result[2] := RotPoint(Rect.Right, Rect.Top);
  Result[3] := RotPoint(Rect.Right, Rect.Bottom);
  Result[4] := RotPoint(Rect.Left, Rect.Bottom);
end;

function Min(A, B: Integer): Integer;
begin
if A < B then
    Result := A
else
    Result := B;
end;

function Max(A, B: Integer): Integer;
begin
if A > B then
    Result := A
else
    Result := B;
end;

function GetRLLimit(const RL: TRectList): TRect;
begin
  Result.Left := Min(Min(RL[1].x, RL[2].x), Min(RL[3].x, RL[4].x));
  Result.Top := Min(Min(RL[1].y, RL[2].y), Min(RL[3].y, RL[4].y));
  Result.Right := Max(Max(RL[1].x, RL[2].x), Max(RL[3].x, RL[4].x));
```

```
    Result.Bottom := Max(Max(RL[1].y, RL[2].y), Max(RL[3].y, RL[4].y));
end;


procedure Rotate;
var
    x, y, xr, yr, yp: Integer;
    ACos, ASin: Double;
    Lim: TRect;
begin
    W := Bmp.Width;
    H := Bmp.Height;
    SinCos(-Angle * Pi / 180, ASin, ACos);
    Lim := GetRLLimit(RotateRect(Rect(0, 0, Bmp.Width, Bmp.Height), Point(0, 0),
      Angle));
    Bitmap.Width := Lim.Right - Lim.Left;
    Bitmap.Height := Lim.Bottom - Lim.Top;
    Bitmap.Canvas.Brush.Color := BackColor;
    Bitmap.Canvas.FillRect(Rect(0, 0, Bitmap.Width, Bitmap.Height));
for y := 0 to Bitmap.Height - 1 do
begin
      Dest := Bitmap.ScanLine[y];
      yp := y + Lim.Top;
for x := 0 to Bitmap.Width - 1 do
begin
        xr := Round(((x + Lim.Left) * ACos) - (yp * ASin));
        yr := Round(((x + Lim.Left) * ASin) + (yp * ACos));
if (xr > -1) and (xr < W) and (yr > -1) and (yr < H) then
begin
          Src := Bmp.ScanLine[yr];
          Inc(Src, xr);
          Dest^ := Src^;
end;
       Inc(Dest);
end;
end;
```

```
end;


begin
  Bitmap.PixelFormat := pf24Bit;
  Bmp := TBitmap.Create;
try
    Bmp.Assign(Bitmap);
    W := Bitmap.Width - 1;
    H := Bitmap.Height - 1;
if Frac(Angle) <> 0.0 then
      Rotate
else
case Trunc(Angle) of
       -360, 0, 360, 720: Exit;
       90, 270:
begin
           Bitmap.Width := H + 1;
           Bitmap.Height := W + 1;
           SetLength(VertArray, H + 1);
v1 := 0;
           v2 := 0;
if Angle = 90.0 then
            v1 := H
else
            v2 := W;
for y := 0 to H do
            VertArray[y] := Bmp.ScanLine[Abs(v1 - y)];
for x := 0 to W do
begin
            Dest := Bitmap.ScanLine[x];
for y := 0 to H do
begin
             v1 := Abs(v2 - x) * 3;
with Dest^ do
begin
```

```
            B := VertArray[y, v1];
            G := VertArray[y, v1 + 1];
            R := VertArray[y, v1 + 2];
end;
          Inc(Dest);
end;
end
end;
     180:
begin
for y := 0 to H do
begin
          Dest := Bitmap.ScanLine[y];
          Src := Bmp.ScanLine[H - y];
          Inc(Src, W);
for x := 0 to W do
begin
            Dest^ := Src^;
            Dec(Src);
            Inc(Dest);
end;
end;
end;
else
     Rotate;
end;
finally
  Bmp.Free;
end;
end;
```