МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Романюка Михайла Вадимовича
(прізвище, ім'я, по батькові здобувача)

на тему
Розробка гри "Пазли" для розвитку дітей дошкільного віку
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД

(повна назва бази дослідження)

науковий керівник к.е.н., доцент _____ Лисенко В.С.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025 р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи Розробка гри “Пазли” для розвитку дітей дошкільного віку

Керівник роботи к.е.н., доцент Лисенко В.С.
затверджені наказом закладу вищої освіти від «04» березня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об’єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Розробка алгоритму розв’язання задачі

Розділ 3. Організація інформаційного забезпечення

Розділ 4. Розробка програмного забезпечення задачі

Об’єкт дослідження: комп’ютерна гра

Предмет дослідження: розробка програмного забезпечення комп’ютерної гри

Мета кваліфікаційної роботи: *створення гри «Пазли» для розвитку дітей дошкільного віку*

5. Дата видачі завдання «04» березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Лисенко В.С.___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Романюк М.В.____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка гри “Пазли” для розвитку дітей дошкільного віку»

Романюка Михайла Вадимовича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня магістра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.і

У бакалаврській дипломній роботі розроблена комп’ютерна гра “Пазли” для дітей дошкільного віку. Гра розроблена на мові програмування C# з використанням технології ADO .NET для роботи з базами даних. Перевагою даної гри є завантаження картинки, як з бази даних, так і з довільного графічного файлу. Вікно гри не має жорстких границь та може динамічно змінюватись. Розроблено власний клас для окремого елементу картинки GraphItem, який дозволяє вдало реалізувати логіку гри. Відповідна гра розвиває логіку та пам’ять у дітей дошкільного віку. Веб-версія гри розроблена з використанням технології ASP .NET. За допомогою OpenGL відбувається виведення карток Домана у 3D-графіці за якими у подальшому відбуватиметься процес навчання.

Розроблене програмне забезпечення рекомендується до використання у навчальному процесі в дошкільних закладах. Впровадження створеного програмного забезпечення сприятиме значному покращенню освітнього процесу в дошкільних закладах.

Ключові слова: ПЗ, СУБД, C#, ADO .NET.

ЗМІСТ

ВСТУП

У наш час розвиток цивілізованого суспільства справедливо пов'язують із періодом інформатизації. Сучасне виробництво та інші сфери діяльності дедалі більше залежать від інформаційної підтримки та обробки великого обсягу даних. Комп’ютери стали універсальними інструментами для роботи з інформацією, відіграючи роль каталізатора інтелектуального розвитку як окремих людей, так і суспільства в цілому. Завдяки комп’ютерам і засобам комунікації, що їх використовують, забезпечується ефективний обмін інформацією. Поява та вдосконалення комп’ютерної техніки стали ключовими факторами інформатизації. У межах інформаційного суспільства формуються нові типи масової комунікації, соціальних зв’язків, мислення та способу життя, а також нові моделі в економіці, політиці та управлінні. До цього необхідно готуватися заздалегідь, і саме на це повинна орієнтуватися система освіти та виховання.

Інформатизація освіти – це впровадження в освітню сферу методологічних підходів, технологій і практичного досвіду використання сучасних інформаційно-комунікаційних засобів, які спрямовані на досягнення психолого-педагогічних цілей навчання та виховання в умовах, що сприяють збереженню здоров’я і комфортному навчанню.

Це означає поступовий перехід до більш сучасної моделі освіти, яка передбачає активну участь учнів у самостійному пошуку знань і встановленні взаємозв’язків між різними поняттями та явищами. Застосування новітніх інформаційних технологій трансформує роль педагога – він стає не лише викладачем, а й дослідником, координатором і наставником.

Використання інновацій в освіті сприяє:

- індивідуалізації навчання з урахуванням особистих характеристик, здібностей та інтересів учнів;
- активізації пізнавальної діяльності, орієнтованої на самостійність і пошук;

- формуванню мотивації до саморозвитку і постійного навчання;
- інтеграції знань через міждисципліпарні підходи;
- підвищенню гнучкості та оновлюваності освітнього процесу;
- зміні підходів до організації дозвілля та позаурочної діяльності дітей.

У «Концепції дошкільного виховання» наголошено, що всі українські програми мають на меті повагу до особистості дитини та любов до неї. Водночас централізоване управління вихованням втрачає актуальність, адже єдина типова програма сприяла формуванню жорсткої навчально-дисциплінарної моделі.

Основною метою дипломної роботи є створити комп’ютерну гру для розвитку дітей дошкільного віку на мові C# технологіями ASP .NET. Програмне забезпечення, що розроблюється, дозволить розвинути у дитині уважність, цілеспрямованість та наполегливість до перемоги.

Для досягнення мети, поставленні наступні задачі:

– визначення предметної області;

– аналіз існуючого програмного забезпечення;

– розробка структури програмного забезпечення;

– розробка інтерфейсу користувача задачі;

– розробка програмного забезпечення задач;

Для розв'язання поставлених задач застосовувались такі методи дослідження:

– теоретичний аналіз наукової та навчальної літератури з проблеми дослідження;

– розробка та тестування програмного забезпечення.

Вибір методів дослідження визначався особливостями розв'язаних завдань.

Об'єктом дослідження є розвиваюча гра.

Предметом дослідження є створення програмного забезпечення розвиваючої гри для дітей дошкільного віку.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Загальна характеристика розвитку ІКТ у сфері дошкільної освіти

Формування України як демократичної держави та її інтеграція до європейського простору стимулюють позитивні зміни в стратегії розвитку системи дошкільної освіти. В умовах глобалізації особливо актуальним стає оновлення змісту дошкільної освіти, орієнтація на гуманістичні цінності та принципи, а також акцент на розвитку особистості дитини як головного ресурсу суспільного поступу.

Необхідність використання комп'ютерних технологій у навчанні передбачена чинними нормативно-правовими документами. Зокрема, у Державній національній програмі «Освіта України XXI століття» в розділі «Дошкільне виховання» підкреслюється потреба розвивати освіту на основі інноваційних підходів, впроваджувати сучасні педагогічні технології та враховувати науково-методичні досягнення. Одним із важливих завдань є підготовка нової генерації педагогів, з високим рівнем фахової підготовки та загальної культури.

Ці орієнтири спрямовані передусім до фахівців, які працюють у практичній сфері освіти, закликаючи їх до оновлення професійного підходу відповідно до сучасних вимог. У «Національній доктрині розвитку освіти» (стаття IX) також наголошується на тому, що пріоритетним напрямом є впровадження інформаційно-комунікаційних технологій. Їх застосування дозволяє покращити навчально-виховний процес, зробити освіту більш доступною та ефективною, а також підготувати молодь до життя в умовах інформаційного суспільства.

Цей документ також регламентує створення сучасних освітніх засобів, які відповідають міжнародному науково-технічному рівню і є ключовими для реалізації ефективної освітньої стратегії.

Сучасний розвиток суспільства, науки й техніки вимагає від системи дошкільної освіти впровадження нових засобів навчання, адже саме дошкільний період є найсприятливішим для розвитку творчих здібностей і пізнавальної активності дитини. Якісна дошкільна освіта виступає фундаментом для подальшого успішного навчання протягом усього життя. Саме тому важливо забезпечити всебічний розвиток дитини з найраніших років, стимулюючи її інтерес до пізнання світу.

Використання комп'ютерних технологій у дошкільній освіті передбачене чинною правовою базою. Система дошкільної освіти потребує постійного оновлення й впровадження інноваційних технологій, які мають стати інтегрованим елементом освітнього процесу, підвищуючи його результативність. Це, разом із професійною підготовкою педагогів і методичним забезпеченням, слугує показником ефективності як шкільних, так і дошкільних закладів.

Інформаційне середовище відіграє важливу роль у забезпеченні ефективної взаємодії між учасниками освітнього процесу – дітьми, педагогами та батьками. Інформаційно-комунікаційні технології мають широке застосування не лише у викладанні, а й у сфері управління та методичної роботи.

Інноваційні педагогічні технології формуються на стику педагогіки, психології, соціології, економіки освіти та менеджменту і нині є провідним напрямом освітнього розвитку. Серед дієвих засобів вирізняються мультимедійні презентації, які роблять заняття емоційно насиченими та привабливими для дітей. Вони збуджують інтерес, візуалізують навчальний матеріал і сприяють кращому запам'ятовуванню. Під час вивчення об'єктів та їхніх властивостей формуються навички зорового сприймання, аналізу, зосередженості, що позитивно впливає на розвиток пам'яті й уваги.

Застосування комп'ютерів, мультимедіа й ІКТ як дидактичних інструментів сприяє підвищенню мотивації до навчання, індивідуалізації освітнього процесу, розвитку креативності та створенню сприятливого

емоційного клімату на заняттях. Доведено, що людина краще запам’ятовує інформацію, якщо вона подається одночасно через зір і слух – понад 50% інформації засвоюється саме в такий спосіб.

Таким чином, використання яскравих образів у презентаціях полегшує сприйняття матеріалу та сприяє його ефективному засвоєнню. Комп’ютерні технології створюють умови для підвищення ефективності навчання та розширення вікових меж його можливостей.

Методична робота в закладі освіти – це комплекс заходів, що спрямовані на вдосконалення професійної майстерності педагогів, розвиток їхнього творчого потенціалу, а в підсумку – на зростання рівня освіти й вихованості дітей.

Головне завдання служби інформаційного забезпечення полягає у:

– створенні інформаційно-освітнього середовища закладу з поступовим вирішенням завдань цифровізації освіти;

– інтеграції сучасних інформаційних технологій у навчальний та управлінський процеси;

– наданні інформаційно-методичної підтримки освітньої діяльності;

– впровадженні електронного документообігу.

Відповідно, методична служба має на меті формування згуртованого педагогічного колективу та розробку ефективної моделі методичної роботи, яка дозволить обрати найкращі технології для організації навчального процесу. У зв’язку з цим методисти використовують різноманітні форми співпраці з педагогами, зокрема:

– організацію тижнів педагогічної майстерності з відкритими заняттями, де педагоги демонструють успішний досвід застосування ІКТ;

– проведення майстер-класів, у рамках яких вихователі опановують прийоми використання технологій у роботі з дітьми;

– створення педагогічних майстерень і парного навчання, де відбувається обмін досвідом та підтримка колег у впровадженні нових методів.

Навчальні й тематичні семінари спрямовані на вдосконалення професійних умінь вихователів, наприклад: «Створення мультимедійної презентації», «Підготовка інтерактивних дидактичних матеріалів за допомогою Інтернет-ресурсів» тощо. Також доцільно проводити конкурси:

– електронне портфоліо педагога;

– відеопрезентації «Познайомтеся – моя група»;

– створення електронних методичних матеріалів і навчальних презентацій.

Застосування сучасних ІКТ у системі дошкільної освіти є однією з найактуальніших тем сучасної педагогіки. Їх впровадження суттєво змінює підходи до навчання, змінює його зміст і методи. Перед вихователями постає завдання пошуку ефективних інструментів навчання, і комп'ютер, який так приваблює дітей, стає потужним засобом мотивації та залучення в освітній процес.

Мультимедіа не лише пришвидшує передачу знань і покращує їхнє засвоєння, а й активно сприяє розвитку когнітивних і мовленнєвих процесів, формує естетичне сприйняття, розвиває емоційну сферу та моральні якості дитини.

Таким чином, ІКТ у дошкільній освіті відіграють роль трансформуючого чинника, що збагачує зміст навчання та відповідає сучасним вимогам до освітнього процесу.

1.2. Аналіз існуючого програмного забезпечення

Розвиваючі комп'ютерні ігри роблять позитивний вплив на цілий ряд психічних процесів дитини. Сучасна педагогіка використовує нові комп'ютерні технології для навчання дітей вже в дошкільному віці.

Як відомо, особливу розумову активність дитина проявляє в ході досягнення ігрової мети як на занятті, так і в повсякденному житті. Гра є однією з форм практичного мислення дітей. І тому використання

комп’ютерних ігор у освітній діяльності є природним для дитини і слугує ефективним засобом підвищення мотивації і забезпечення індивідуалізації навчання та розвитку особистісних здібностей. У грі дитина оперує своїми знаннями, досвідом, враженнями, що відображаються у формі ігрових способів дії, ігрових знаків, які набувають в смисловому полі гри.

Однією з найважливіших функцій комп’ютерних ігор є навчальна. Комп’ютерні ігри створені так, що дитина може собі уявити окреме поняття або конкретну ситуацію, а одержати узагальнене поняття про всі схожі предмети або ситуації. У такий спосіб у дитини розвиваються такі важливі операції мислення як узагальнення і класифікація. Під час гри на комп’ютері, дитина рано починає розуміти, що предмети на екрані - це не реальні речі, а тільки знаки цих реальних речей. Таким чином, у дітей починає розвиватися так звана знакова функція свідомості, тобто розуміння того, що є кілька рівнів навколишнього середовища - це й реальні речі, і картинки, схеми, слова або числа і т.ін.

Комп’ютерні ігри вчать дітей переборювати труднощі, контролювати виконання дій, оцінювати результати. Завдяки комп’ютеру стає ефективним навчання цілеспрямованості, плануванню, контролю і оцінки результатів самостійної діяльності дитини, через сполучення ігрових і не ігрових моментів. Дитина входить у сюжет ігор, засвоює правила, відповідно діє і прагне досягнення результатів. Крім того, практично у всіх іграх є свої герої, яким потрібно допомогти виконати завдання. Таким чином, комп’ютер допомагає розвити не тільки інтелектуальні здібності дитини, але й виховати вольові якості, такі як самостійність, зібраність, зосередженість, посидючість, спонукає дитину до співпереживання, допомоги героям ігор тощо, збагачуючи тим самим його ставлення до навколишнього світу.

Слово puzzle з англійської перекладається як "загадка" або "головоломка". Ця розвиваюча гра являє собою малюнок різного ступеня складності та розміру, розрізане на безліч шматочків практично однакової форми. Завдання гравця, зібрати цей малюнок, орієнтуючись на відтінки

кольорів і незначні відмінності у формі фрагментів.

Головоломка пазл це не тільки захоплююче і цікаве заняття , воно має велику розвиваючу цінність. По-перше , дитина вчиться концентрувати свою увагу та вчиться з дрібних деталей скласти єдину картинку. Причому, спочатку цей процес відбувається в голові, в мозку дитини. Він вчиться візуальному аналізу, уважності, посидючості, терпінню, логічно мислити, уяву та пам’ять.

По-друге, збирати пазл, особливо складні, можна одночасно декільком учасникам. Спільне рішення єдиного завдання розвиває комунікативні навички дітей. Вони вчаться працювати однією командою, розподіляючи рішення задачі на кількох учасників. Діти вчаться терпінню і умінню допомагати один одному, колективно вирішувати проблему і дійти до поставленої мети, не відступати при виникненні труднощів.

По-третє, пазли знайомлять дітей з дивовижними природними явищами, різними географічними пам'ятками , з відомими творами живопису і скульптури, з різними представниками тваринного світу. Збираючи такі картини , діти починають цікавитися тими об'єктами , які зображені на них. Це справжній стимул для вивчення нового.

Розглянемо один з прикладів вже існуючих розвиваючих ігор. На рисунку 1.1 представлена головна сторінка сайту www.igraemsa.ru:

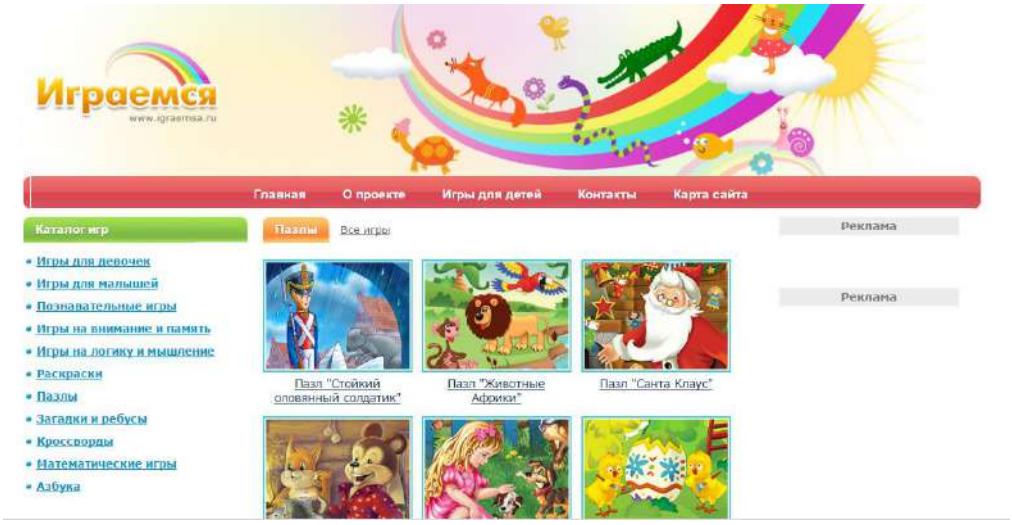

Рис. 1.1. Головна сторінка сайту www.igraemsa.ru

На рисунку ми бачимо, що головна сторінка містить меню, категорії вже існуючих ігор, а також новинки на сайті.

У розділі «Пазли» є гра, яка працює по принципу створюваної нами гри (рис. 1.2).

Рис. 1.2. Пазл «Чаювання»

Ця гра досить приваблююча, має доступний і інтуїтивно зрозумілий інтерфейс, але все ж має ряд недоліків, порівняно з ПЗ, яке буде розроблено.

– Розмір ігрового поля фіксований;

– Наявний лише один малюнок для збирання пазлу;

– Можливість користування лише on-line;

– Форма комірок ігрового поля лише одна.

Отже, комп’ютерна гра, яка буде розроблена, матиме ряд переваг перед вже існуючим розробкам.

Висновки до розділу 1

Отже, у цьому розділі було успішно сформульовано характеристику задачі, мету її вирішення та цілі, які вона переслідує. Була визначена структура та зміст вхідної та вихідної інформації, основні вимоги до їх оформлення, джерела, які забезпечують задачу вхідною інформацією, та особи, які мають отримати вихідну інформацію. На основі цих даних можна переходити до наступного етапу розробки програмного забезпечення.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

Першим кроком роботи з програмою, яка написана на мові програмування C# здійснюється на основі управління через пункт головного меню рис. 2.1.

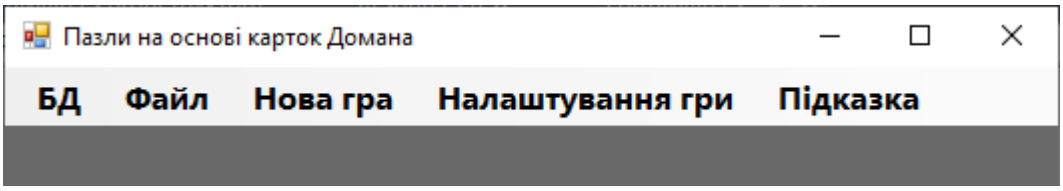

Рис. 2.1. Головне меню програми

Розроблений програмний продукт працює відповідно до нижче наведеного алгоритму (рис. 2.2).

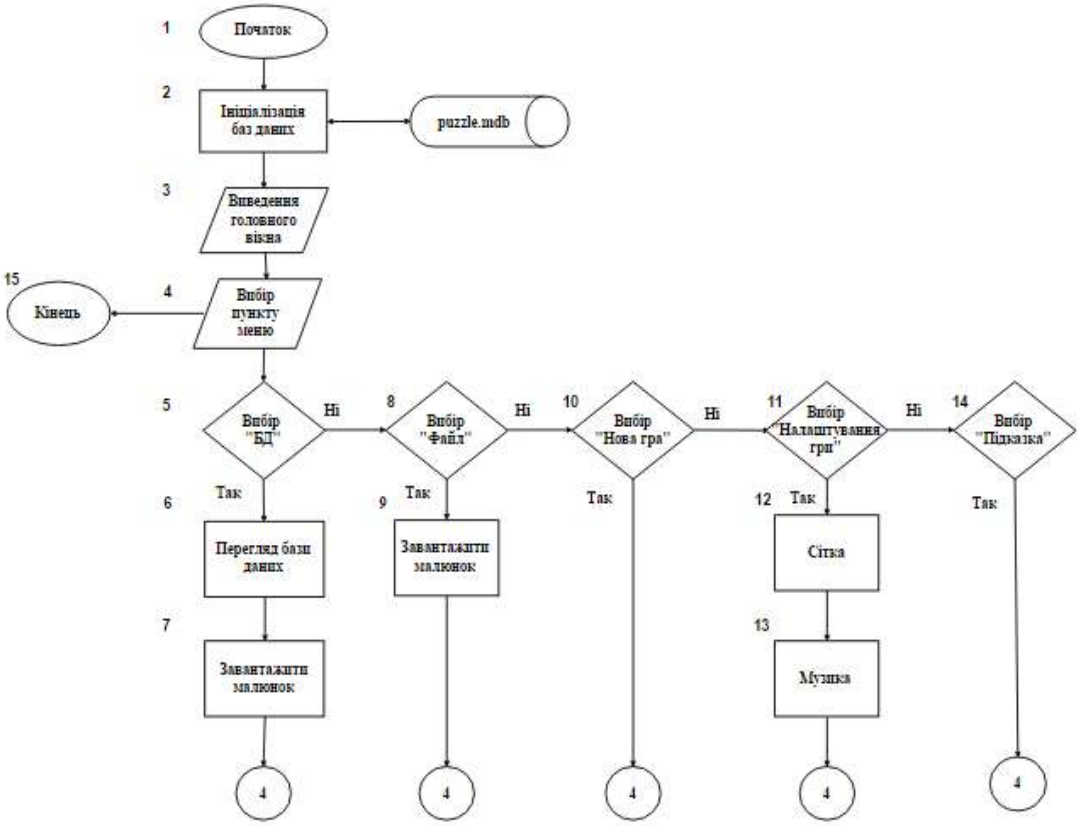

Рис. 2.2. Блок-схема алгоритму програми

Як зображено на рис. 2.2, після завантаження головного вікна програми, користувачеві надається можливість обрати п’ять пунктів меню. Розглянемо їх більш детальніше. В блоці 4 користувач обирає необхідний пункт меню.

Під блоком 5 починається робота з базою даних. Якщо база з самого початку була успішно підключена, то ми можемо переглянути базу даних (блок 6) та завантажити малюнок для гри з бази даних (блок 7), якщо база не підключилась або ж її нема, то буде у новому вікні викликано попередження, що «БД відсутнє» і (блоки 5-7) будуть відсутніми.

При виборі блоку 8 ми маємо змогу вибрати малюнок з файлу, а потім завантажити малюнок (блок 9).

При виборі блоку 11 користувачеві надається можливість обрати розмір ігрового поля (сітки) та обрати, чи буде музика ввімкнута або ж увімкнена (блоки 12-13).

Вибір блоку 14 надає можливість підказку користувачеві, у новому вікні показується малюнок, який він повинен зібрати.

Висновки до розділу 2

Отже, у цьому розділі були розглянуті алгоритми вирішення поставленої задачі та розроблений алгоритм використання програми у вигляді блок-схеми.

РОЗДІЛ 3
ОРГАНІЗАЦІЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

У роботі була створена одна база даних «puzzlel.mdb», яка підключається до програми, створеної на мові C#. В цій базі є дві таблиці: «Category», в ній міститься перелік категорій малюнків для гри, та «Picture», яка містить самі малюнки.

Опис властивостей стовпчиків таблиць «Category» і «Picture» представлено у таблиці 3.1 і 3.2 відповідно.

Таблиця 3.1

Опис структури таблиці «Category»

Найменування	Поле	Тип поля	Довжина	Ключ
Код	ID	Лічільник	Довге ціле	+
Категорія	Cat	Короткий текст	50	-

Таблиця 3.2

Опис структури таблиці «Picture»

Найменування	Поле	Тип поля	Довжина	Ключ
Код	ID	Лічільник	Довге ціле	+
Категорія	ID_Cat	Числовий	Довге ціле	-
Назва	Name	Текстовий	50	-
Малюнок	Picture	Поле об’экта OLE	-	-

Структура таблиць зображена на рисунку 3.1 та рисунку 3.2 відповідно:

Category

Имя поля	Тип данных
ID	Счетчик
Cat	Текстовый

Рис. 3.1. Структура таблиці «Category»

Picture

Имя поля	Тип данных
ID	Счетчик
ID_Cat	Числовой
Name	Текстовый
Picture	Поле объекта OLE

Рис. 3.2. Структура таблиці «Picture»

Нижче наведено код підключення до бази даних:

```
//Підключення до БД
            try
            {
                conn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=puzzle.mdb");
                conn.Open();
            }

            catch (Exception)
            {
                MessageBox.Show("База даних puzzle.mdb відсутня!\nПри роботі з програмою
не буде доступна опція БД!",
                    "Увага!",MessageBoxButtons.OK,MessageBoxIcon.Information);

                бДToolStripMenuItem.Visible = false;
            }
```

Висновки до розділу 3

В цьому розділі була розглянута загальна характеристика інформаційного забезпечення, наведено опис розробленої бази даних для даного програмного забезпечення.

РОЗДІЛ 4
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАДАЧІ

4.1. Розробка програмного забезпечення на мові C#

Мова програмування C# є сучасним високорівневим інструментом розробки, що вирізняється рядом важливих переваг. Вона структурована та має інтуїтивно зрозумілий синтаксис, завдяки чому її легко вивчати та використовувати. Потужні можливості для налагодження й редагування коду сприяють комфортній і продуктивній роботі програміста. C# базується на концепціях мов Java та C++, успішно поєднуючи їхні найкращі риси та доповнюючи власними сучасними рішеннями. При цьому мова не перевантажена — вона містить лише ті елементи, які справді потрібні для ефективної розробки програмного забезпечення.

Мова програмування C# має низку характерних особливостей [1–3]:

– Вона повністю підтримує принципи об’єктно-орієнтованого програмування, включно зі спадкуванням, інтерфейсами, віртуальними методами та перевантаженням операторів.

– У мові представлено чітко визначений набір базових типів.

– Передбачено автоматичне створення XML-документації прямо з коду.

– Система керування пам’яттю автоматично очищає динамічно виділену пам’ять.

– Можна використовувати користувацькі атрибути для класів і методів – це зручно для документування, а також може впливати на процес компіляції (наприклад, маркування функцій для компіляції лише в режимі відладки).

– Мова має доступ до багатої бібліотеки базових класів платформи .NET, а також до API Windows.

– Підтримує властивості та події у стилі Visual Basic.

– Гнучкі налаштування компіляції дозволяють створювати виконувані файли або бібліотеки компонентів .NET, які можуть бути використані іншими програмами подібно до компонентів ActiveX.

– C# може застосовуватись для створення динамічних веб-сторінок за допомогою технології ASP.NET [9-12].

C# повністю інтегрована з платформою .NET і базується на її типах даних. Потужна бібліотека .NET бере на себе значну частину рутинної роботи, що дає змогу розробникам зосередитись на вирішенні складних задач, використовуючи готові компоненти.

Технологія ADO.NET забезпечує доступ до даних у застосунках, створених на базі .NET. Вона не є прямим продовженням ADO, а виступає самостійною складовою платформи. ADO.NET орієнтована на роботу з локальними копіями даних за допомогою об’єктів DataSet, які можуть містити кілька пов’язаних таблиць. Такі об’єкти дозволяють виконувати дії з даними навіть без постійного підключення до бази даних.

На відміну від ADO, нова технологія є частиною керованого середовища .NET, з підтримкою пам’яті CLR та уніфікованої системи типів (класи, інтерфейси, делегати тощо). Класи ADO.NET зберігаються у бібліотеці System.Data.dll і доступні з будь-якої мови, сумісної з .NET.

Інтерфейс користувача програми починається з завантаження головного вікна, що містить елементи керування: `MenuStrip`, `ToolStrip` і `OpenFileDialog`.

- `MenuStrip` – контейнер для меню, яке може містити пункти типу `ToolStripMenuItem`, а також елементи `ComboBox` і `TextBox`. Зазвичай у меню використовуються саме `ToolStripMenuItem`, інші елементи – на панелях інструментів.
- `ToolStrip` – базовий клас, що лежить в основі `MenuStrip`, `StatusStrip` і `ContextMenuStrip`, дозволяє створювати гнучкі панелі інструментів з розширеними можливостями. Клас

`ToolStripManager` допомагає керувати стилем і макетом елементів керування.

- `OpenFileDialog` – клас, який забезпечує вибір файлів із файлової системи. Після вибору файлу його можна відкрити або за допомогою `StreamReader`, або використати метод `OpenFile`.

Після того, як користувач вибирає файл, який потрібно відкрити, існує два підходи до відкриття файлу. Якщо розробник вважає за краще працювати з потоками файлів, можна створити екземпляр класу StreamReader. З іншого боку, для відкриття обраного файлу можна використовувати метод OpenFile.

На рис. 4.1 ми бачимо головне вікно програми «Пазли».

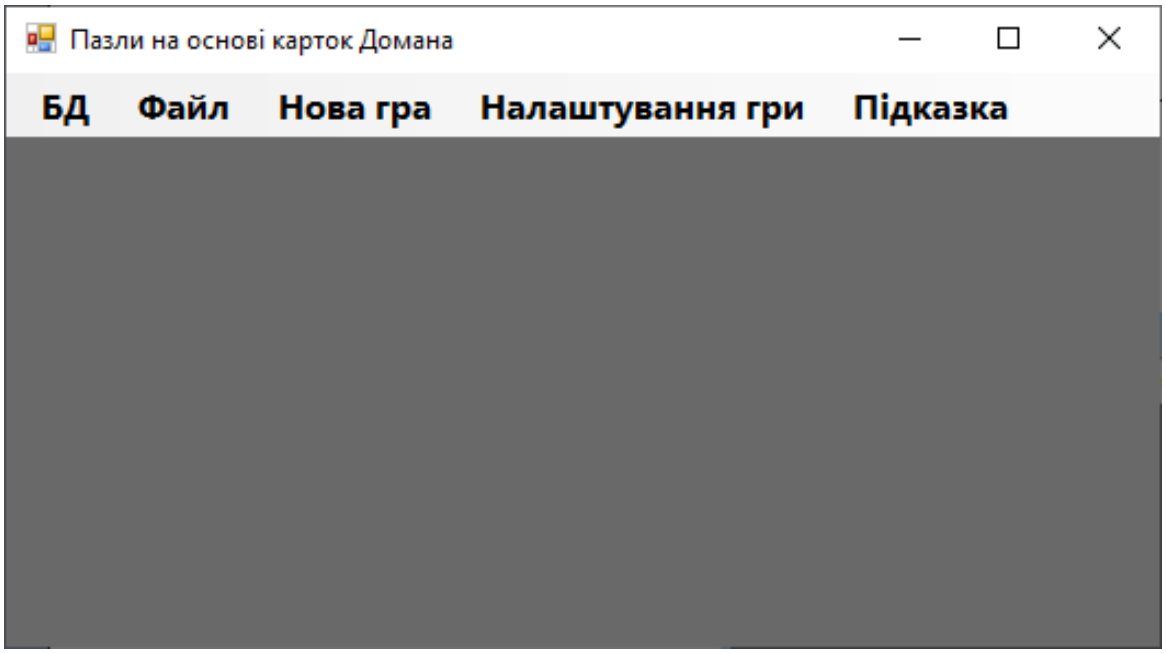

Рис. 4.1. Головне вікно програми

Головне меню програми складається з таких пунктів [додаток А]:

1) БД:
   - Перегляд БД;
   - Завантажити малюнок;
2) Файл:
   - Завантажити малюнок;
3) Нова гра;
4) Налаштування:

– Музика;

– Сітка;

5) Підказка.

Тепер розглянемо ці пункти більш детально.

Пункт меню «БД» дозволяє роботу з базою даних. Ми можемо переглянути базу або завантажити малюнок через базу даних. При натисканні «Перегляд БД» та «Завантажити малюнок» з'являється нова форма (рис. 4.2). На формі знаходиться label, button – для вибору малюнка для гри (при «Перегляд БД» ця кнопка не є активною), comboBox – в ньому вибираємо категорію карток Домана, а в pictureBox демонструються малюнки з comboBox.

Завантаження файлу виконується так:

```
//Завантаження зображення з БД
        private void загрузитьКартинкуToolStripMenuItem_Click(object sender, EventArgs e)
        {

            FormBD bd = new FormBD();
            bd.conn = conn;
            bd.kod = 2;

            if (bd.ShowDialog() == DialogResult.OK)
            {
                Picture = new Bitmap(bd.pictureBox1.Image);
                FormirPuzzle();
                Peremesh(GItems);
                FormMain_Resize(null, null);
            }
        }
```

Для заповнення comboBox і для вибору елемента з БД застосовується такий код:

```
//Заповнення combobox та прив'язка даних
        private void FormBD_Load(object sender, EventArgs e)
        {

            if (kod == 1)
            {
                button1.Visible = false;
                Text = "Перегляд БД";
            }

            if (kod == 2)
            {
                button1.Visible = true;
                Text = "Вибір БД";
```

```
            }

            try
            {
                string select = "SELECT Cat, ID FROM Category";
                OleDbCommand cmd = new OleDbCommand(select, conn);
                OleDbDataReader reader = cmd.ExecuteReader();

                while (reader.Read())
                    comboBox1.Items.Add(new Name(reader[0].ToString(), (int)reader[1]));
                reader.Close();
                cmd.Dispose();
                comboBox1.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        //Вибір елементу
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {

            try
            {

                if (da != null) da.Dispose();
                pictureBox1.DataBindings.Clear();
                bindingSource1.DataSource = null;
                string str = "Select Picture From Picture where ID_Cat = " +
((Name)comboBox1.SelectedItem).ItemData;
                da = new OleDbDataAdapter(str, conn);
                ds.Tables.Clear();
                da.Fill(ds);

                bindingSource1.DataSource = ds.Tables[0];
                bindingNavigator1.BindingSource = bindingSource1;

                //Прив'язка picturebox до зображення
                pictureBox1.DataBindings.Add("Image", bindingSource1, "Picture", true);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Також на формі присутні елемент BindingNavigator та BindingSource BindingNavigator представляє стандартний спосіб навігації і управління даними на формі. У більшості випадків елемент управління BindingNavigator з'єднується попарно з елементом управління BindingSource, щоб забезпечувати переміщення між записами даних на формі і взаємодіяти з ним.

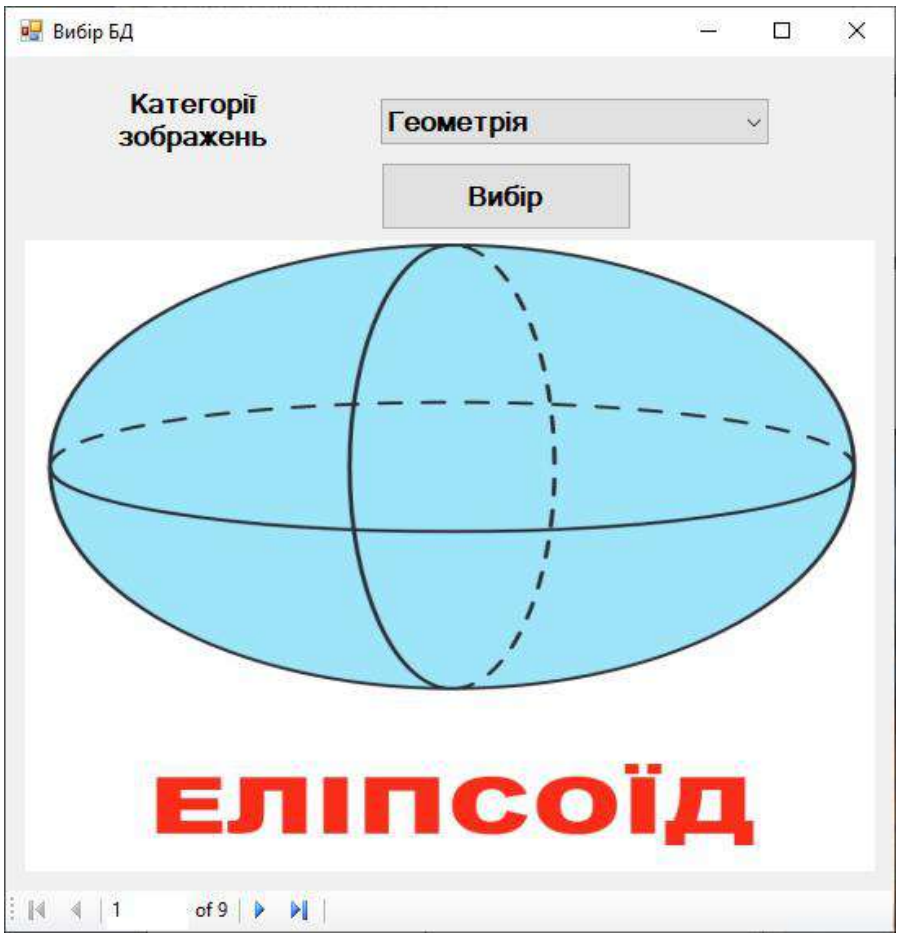

Рис. 4.2. Форма для БД

У цих випадках властивість BindingSource встановлюється на відповідний компонент System.Windows.Forms.BindingSource, який діє в якості джерела даних. За замовчуванням, користувальницький інтерфейс елементу управління BindingNavigator включає до свого складу ряд кнопок ToolStrip, текстових вікон, а також статичні текстові елементи для найбільш загальних дій, пов'язаних з даними, таких як додавання даних, видалення даних і перехід між даними. Кожен з цих елементів управління може вилучатись або встановлюватися за допомогою відповідного члена елементу управління BindingNavigator. Крім того, є також відповідність членам в класі BindingSource, які програмно виконують аналогічні функції.

Пункт меню «Файл» міститься в собі підпункт «Завантажити малюнок», що дозволяє користувачеві завантажити малюнок довільного розміру,бо розмір картинки буде підстроюватись під розміри прямокутника

(PB.SizeMode = PictureBoxSizeMode.StretchImage;) та формату BMP, JPEG, JPG з будь-якої папки особистого комп'ютера або будь-якого комп'ютера в мережі (рис. 4.3).

Нижче наведено код завантаження файлу:

```
//Завантаження з файлу
        private void загрузитьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //openFileDialog1.InitialDirectory = Directory.GetCurrentDirectory();
            openFileDialog1.Filter = "Файли зображень
(*.bmp;*.jpg;*.jpeg;)|*.bmp;*.jpg;.jpeg";
            if (openFileDialog1.ShowDialog() != DialogResult.OK) return;
            try
            {
                ////////////////////////////////////
                if (Picture != null) Picture.Dispose();
                Picture = new Bitmap(openFileDialog1.FileName);
                FormirPuzzle();
                Peremesh(GItems);
                FormMain_Resize(null, null);
            }
            catch (Exception)
            {
                MessageBox.Show("Невідповідність формату файла!\nОберіть зображення!",
                    "Помилка!",MessageBoxButtons.OK, MessageBoxIcon.Stop);
            }
        }
```

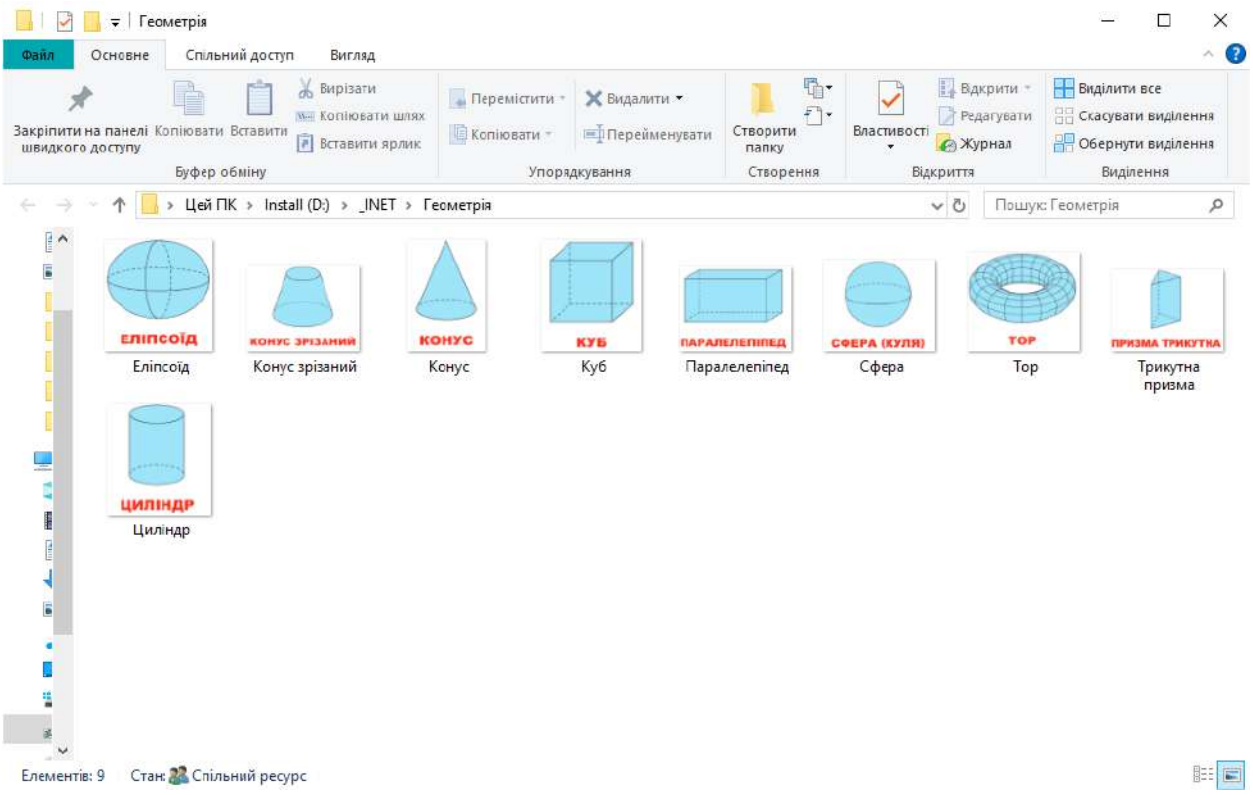

Рис. 4.3. Завантаження малюнка

При натисканні на пункт меню «Налаштування», ми працюємо з налаштуванням музики та сітки. Ми можемо вибрати, чи буде грати музика на протязі гри., чи ні за допомогою функцій private void вклToolStripMenuItem_Click(object sender, EventArgs e) та private void выклToolStripMenuItem_Click(object sender, EventArgs e) відповідно.Код наведено нижче:

```
//Увімкнення музики
        private void вклToolStripMenuItem_Click(object sender, EventArgs e)
        {
            nastr_puzzle.musik = true;
            вклToolStripMenuItem.Checked = true;
            выклToolStripMenuItem.Checked = false;
            sp_music.PlayLooping();
        }

        //Вимкнення музики
        private void выклToolStripMenuItem_Click(object sender, EventArgs e)
        {
            nastr_puzzle.musik = false;
            выклToolStripMenuItem.Checked = true;
            вклToolStripMenuItem.Checked = false;
            sp_music.Stop();
        }
```

Використовується клас SoundPlayer. Клас SoundPlayer надає простий інтерфейс для завантаження і відтворення WAV-файлів. Клас SoundPlayer підтримує завантаження WAV-файлу із зазначеного шляху до файлу, URL-адреси, потоку Stream, що містить WAV-файл, або впровадженого ресурсу, що містить WAV-файл. Завантажувати і відтворювати WAV-файли можна в синхронному або асинхронному режіме. Ми використовуємо асинхронний режим, при асинхронному виклику метода завантаження або відтворення викликаючий потік продовжить виконання без перерываній. Метод PlayLooping() циклічно вдтворює звуковий файл формату WAV з використанням нового потоку. Метод PlayLooping викликається до завантаження WAV-файлу в пам’ять, тоді цей файл буде завантажен до початку відтворення. WAV-файл буде відтворюватися до тих пір, поки не буде викликав метод sp_music.Stop.

При виборі пункту «Сітка» з’являється нова форма (рис. 4.4), в якій ми можемо вибрати зі скількох елементів буде складатись пазл. За малювання та розрахунок сітки відповідає функція ComputeCellsCoordinate. В ній рахується

розмір сітки в залежності від кількості комірок. Прив'язка розміру сітки йде до мінімальної сторони - ширина або висота. Малювання сітки відбувається так: визначається менша довжина ігрового вікна (довжина або ширина), а потім саме ця довжина ділиться на кількість елементів ігрового поля. Потім по стовпцям розраховуємо розташування комірок, і в результаті комірки розташовані вертикально. Наведемо приклад коду:

```
// Виведення та перерахунок графічних елементів для 1-го режима
        // GItems - масив граф. елементів
        // kod = 0 - 1 виведення
        // kod = 1 - виведення ліворуч
        // kod = 2 - виведення праворуч

        void ComputeCellsCoordinate(GraphItem[] GItems, int kod)
        {

            int num = kol_kl;

            // довжина сторони квадрату області розташування графелементів
            int lenside = 0;

            // використаємо розміри клієнтської області
            int delta = 20;
            int width  = 0;
            int nach = 0;
            if (kod == 0) width = this.ClientRectangle.Width - delta;
            if (kod == 1 || kod == 2) width = this.ClientRectangle.Width / 2 - delta;
            if (kod == 2) nach =  this.ClientRectangle.Width / 2;

            int height = this.ClientRectangle.Height -delta - panel1.Top;

            // довжина сторони області графелементів трохи менше клієнтської висоти
            lenside = (height < width) ? height : width;

            // довжина сторони комірки
            int lenCell = lenside / num;

            // по стовпчикам розраховуємо розташування комірок графелементів
            int x,y,count = 0;
            for (x = 0; x < num; x++)
            {
                for (y = 0; y < num; y++)
                {
                    GItems[count].CellCoordinate = new Rectangle(nach + x * lenCell + 10,
y * lenCell + delta / 2 + panel1.Top, lenCell, lenCell);
                    count++;
                }
            }

            // У підсумку комірки розташовані у такому порядку:
            ////////////////////////////////
            // 0 10 20 30 40 50 60 70 80 90 //
            // 1 11 21 31 41 51 61 71 81 91 //
            // 2 12 22 32 42 52 62 72 82 92 //
            // 3 13 23 33 43 53 63 73 83 93 //
            // 4 14 24 34 44 54 64 74 84 94 //
            // 5 15 25 35 45 55 65 75 85 95 //
            // 6 16 26 36 46 56 66 76 86 96 //
            // 7 17 27 37 47 57 67 77 87 97 //
```

```
            // 8 18 28 38 48 58 68 78 88 98 //
            // 9 19 29 39 49 59 69 79 89 99 //
            ////////////////////////////////

            Invalidate();
        }
```

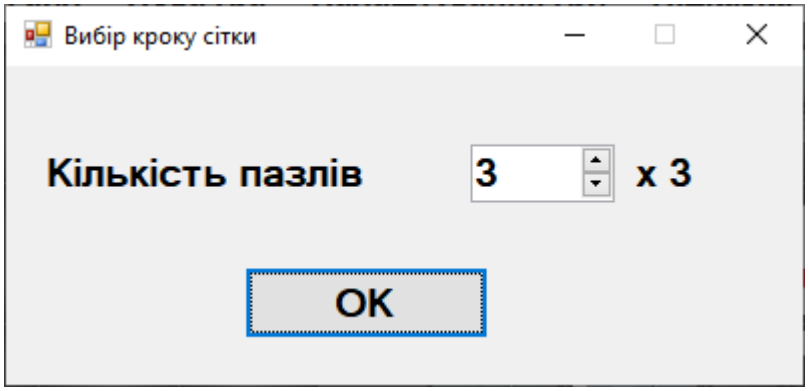

Рис. 4.4. Налаштування сітки

При натисканні на пункт меню «Підказка» викликається нова форма, в якій є pictureBox і ми бачимо оригінал пазла (рис. 4.5).

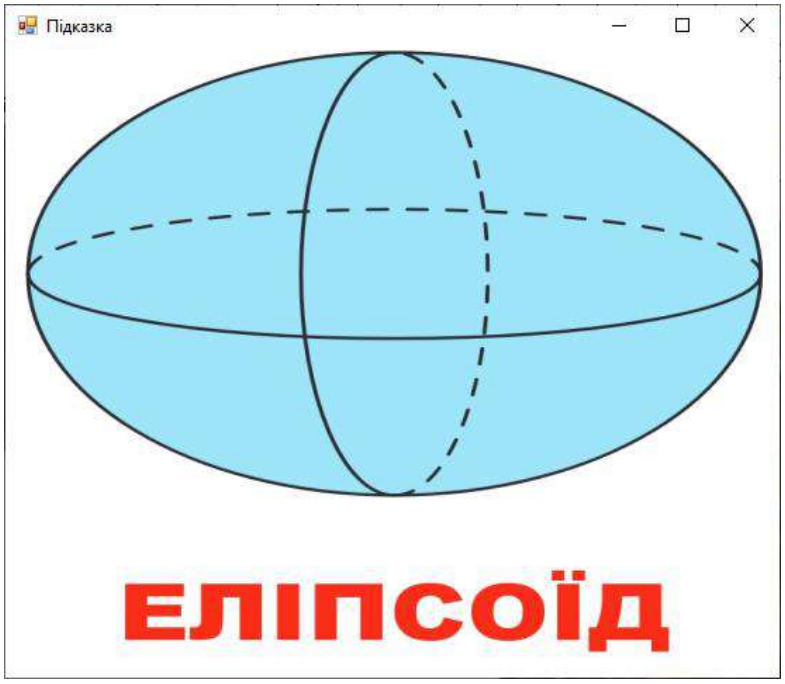

Рис. 4.5. Підказка

Вибравши пункт меню «Нова гра» за допомогою функції FormirPuzzle формується пазл і порожній масив. У разі першого завантаження гри використовуються налаштування за умовчуванням. Збереження і використання збережених налаштувань виконується завдяки серіалізації в двійковий формат (BinnaryFormatter). Так само виконується збереження файлу і його зчитування. Щоб зробити об'єкт серіалізуємим потрібно

забезпечити кожен пов'язаний з ним клас або структуру атрибутом [Serializable]. Якщо є поля, які з якоїсь причини потрібно виключити з серіалізації, їх необхідно позначити атрибутом [NonSerialized]. BinaryFormatter зберігає стан об'єкта в двійковому форматі. Серіалізуються усі поля, незалежно від їх області видимості. Виняток становлять поля помічені атрибутів [NonSerialized]. Крім збереження даних полів, BinaryFormatter також зберігає повне кваліфіковане ім'я кожного типу, повне ім'я збірки, де він визначений, сюди входить інформація про ім’я, версії, маркер загальнодоступного ключа (public key) і культуру. Звідси випливає, що BinaryFormatter це ідеальний вибір в ситуаціях, коли необхідно зберігати повні копії об'єктів для подальшої їх передачі за значенням між доменами для використання в .NET додатках. Тут полягає основний мінус BinaryFormatter - дані, збережені з його допомогою, можуть бути відтворені тільки в інфраструктурі CLI. Причому кожен, хто буде відновлювати дані, повинен мати збірку з серіалізуємим типом.

Серіалізація відбувається за допомогою двох ключових методів Serialize () і Deserialize (). Перший зберігає граф об'єктів у вигляді послідовності байт в зазначений потік. Другий навпаки – перетворює збережені електронні дані в граф об'єктів.

```
//Функція завантаження налаштувань
        private bool LoadSetting()
        {
            bool voz = true;
            // Завантиаження ini-файлу
            FileStream f = null;
            nastr vr = new nastr();
            try
            {
                f = new FileStream("puzzle.ini", FileMode.Open);
                BinaryFormatter bf = new BinaryFormatter();
                vr = (nastr)bf.Deserialize(f);
            }
            catch (FileNotFoundException)
            {
                MessageBox.Show("Файл ініціалізації puzzle.ini відсутній\nНалаштування
прийняті за умовчанням",
                    "Помилка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                voz = false;
            }

            catch (Exception ex)
            {
```

```
                MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                voz = false;
            }

            finally
            {
                if (f != null) f.Close();
            }

            //Перевірка ini

            if (voz)
            {
                if (vr.kol_puzzle < 2 || vr.kol_puzzle > 10) voz = false;
                if (voz) nastr_puzzle = vr;
                else MessageBox.Show("Порушена структура файлу puzzle.ini\nНалаштування
прийняті за умовчанням",
                     "Увага!", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            return voz;
        }

        //Функція зберігання налаштувань
        private void SaveSetting()
        {
            // Зберігання налаштувань

            FileStream f = null;
            try
            {
                f = new FileStream("puzzle.ini", FileMode.Create);
                BinaryFormatter bf = new BinaryFormatter();
                bf.Serialize(f, nastr_puzzle);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Помилка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }

            finally
            {
                if (f != null) f.Close();
            }

        }
```

Перед завантаженням форми ми виділяємо пам'ять для всіх графічних елементів та видаляємо старі картинки. Далі формуємо порожній масив GItemsPuzzle.

```
// Функція формування пазлу та порожнього масиву
        private void FormirPuzzle()
        {
            // Перед завантаженням форми виділимо пам'ять для усіх графічних елементів

            int i, countX = 0;
            int countY = 0;

            int w = Picture.Width / kol_kl;
            int h = Picture.Height / kol_kl;
```

```
            int NumGraphItems = kol_kl * kol_kl;

            // Видалення попередніх зображень
            if (GItems != null)
                for (i = 0; i < GItems.Length; i++)
                    if (GItems[i].Puzzle != null) GItems[i].Puzzle.Dispose();

            GItems = new GraphItem[NumGraphItems];

            //Формування порожнього масиву
            GItemsPuzzle = new GraphItem[NumGraphItems];


            for (i = 0; i < NumGraphItems; i++)
            {
                GItems[i] = new GraphItem(this);
                ////////////////////////////////
                GItemsPuzzle[i] = new GraphItem(this);

                GItems[i].Visible = true;

                // Розріз зображення та поміщення його у масив PictureBox

                // Розміри і координати розміщення створеного прямокутника
                PB = new PictureBox();

                // Копіювання частини зображення
                PB.Image = Picture.Clone(new RectangleF(countX * w, countY * h, w, h),
Picture.PixelFormat);
                ////////////////////////////////////////////
                countY++;
                if (countY == kol_kl)
                {
                    countY = 0;
                    countX++;
                }
                PB.Tag = i; // зберігаємо індекс елементу

                PB.BorderStyle = BorderStyle.None;

                // розміри зображення будуть підлаштовуватись під розміри прямокутника
                PB.SizeMode = PictureBoxSizeMode.StretchImage;
                //Прив'язка зображення до елементу
                GItems[i].Puzzle = PB;
            }

        }
```

Потім робиться розрізання картинки і переміщення її в масив PictureBox і кожному елементу присвоюється Tag індекс. За допомогою функції Peremesh робиться перемішування графічних елементів завдяки цьому класу (рис. 4.6).

```
// Перемішування графічних елементів
        private void Peremesh(GraphItem[] GItems)
        {
            PictureBox t;
            int i, r = 0;
```

```
        for (i = 0; i < GItems.Length; i++)
        {
            r = rand.Next(0, GItems.Length);
            t = GItems[r].Puzzle;
            GItems[r].Puzzle = GItems[i].Puzzle;
            GItems[i].Puzzle = t;
        }

        formir_puzzle = false;
    }
```

Рис. 4.6. Перемішування графічних елементів

Вся графіка виконана за допомогою створеного класу GraphItem. Малювання, видалення, перенесення і активність елементів відбувається саме завдяки цьому класу. Головними властивостями створеного графічного елементу є:

– видимість;

– активність;

– зникання.

Будь-який активний елемент, обраний на ігровому полі є пульсуючим. Ця пульсація досягається шляхом використання таймеру активності, який періодично збільшує і зменшує розміри елементу, а таймер зникання спочатку поступово зменшує елемент, а потім він повністю зникає.

У нижче наведеному коді ми знаходимо активний елемент графічний елемент (якщо його немає, то просто виконується порожній цикл), потім перевіряємо чи може активний графічний елемент перемістити на вказане користувачем місце, і далі міняємо місцями і властивостями активний і невидимий графічні елементи.

```
for (a = 0; a < GItems1.Length; a++)
{
    if (GItems1[a].Active == true || GItems2[a].Active==true)
    {
        if (indexClick != -1)
        {
                if (GItems1[a].Active == true) GItemsA = GItems1;
                if (GItems2[a].Active == true) GItemsA = GItems2;

                // Міняємо місцями і властивостями обчислені графелементи
                PictureBox puzzle = GItems[indexClick].Puzzle;
                GItems[indexClick].Puzzle = GItemsA[a].Puzzle;
                GItems[indexClick].Active = false;
                GItemsA[a].Puzzle = puzzle;
                GItemsA[a].Visible = false;
                GItemsA[a].Active = false;
                GItems[indexClick].Visible = true;
                Invalidate();

         // Робимо невелику затримку даної функції, без затримки роботи додатку,
        // для акцентування уваги користувача на переміщення графелементу.
                Application.DoEvents();
                Thread.Sleep(80);
                Invalidate();
        }

        break;
    }
}
```

Коли завантажується малюнок, тож кожному графічному елементу присвоюється свій індекс, де він повинен находитись. Потім же робиться перемішування. І в тому випадку, якщо індекс зображення збігся з тегом – можна вважати, що пазл збігся і користувач бачить у себе на моніторі повідомлення про перемогу (рис. 4.7). А також звучить музика про перемогу.

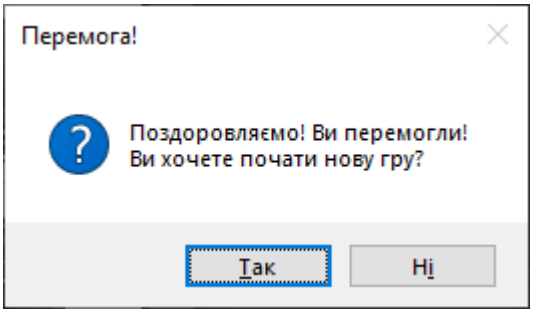

Рис. 4.7. Повідомлення про перемогу

Код функції перевірки на збіг пазлів:

```
//Перевірка на співпадіння пазлу
            if (Proverka(GItemsPuzzle))
            {
                formir_puzzle = true;

                Stream res = Properties.Resources.tush;
                SoundPlayer sp = new SoundPlayer(res);
                sp.PlaySync();
                if (nastr_puzzle.musik) sp_music.PlayLooping();
                DialogResult dd = MessageBox.Show("Поздоровляємо! Ви перемогли!\nВи
хочете почати нову гру?",
                    "Перемога!", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (dd == DialogResult.Yes)
                {
                    FormirPuzzle();
                    Peremesh(GItems);
                    FormMain_Resize(null, null);
                }
            }
```

4.2. Розробка програмного забезпечення з використанням технології ASP .NET

ASP .NET – це технологія створення веб-додатків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft .NET і розвитком старішої технології Microsoft ASP.

Звісно, ASP .NET зовні багато в чому зберігає схожість зі старішою технологією ASP, що дозволяє розробникам відносно легко перейти на ASP .NET. У той же час внутрішній пристрій ASP .NET істотно відрізняється від ASP, оскільки вона заснована на платформі .NET і, отже, використовує всі нові можливості, що надаються цією платформою.

Хоча ASP .NET бере свою назву від старої технології Microsoft, ASP, вона значно від неї відрізняється. Microsoft повністю перебудувала ASP .NET, який є основою всіх додатків Microsoft. NET. Програмісти можуть писати код для ASP .NET, використовуючи різні мови програмування, підтримувані в .NET Framework, зазвичай Visual Basic .NET, JScript .NET або C #, а також «відкриті» мови, наприклад, Perl і Python. ASP .NET має перевагу в швидкості в порівнянні з іншими технологіями, заснованими на скриптах, так як при першому зверненні код компілюється і поміщається в

спеціальний кеш, і згодом тільки виконується, не вимагаючи витрат часу на оптимізацію, і т.д.

ASP .NET є потужною об'єктно-орієнтованої технологією. Вона не тільки надає коду повний доступ до всіх об'єктів .NET Framework, але і дозволяє використовувати всі концепції об'єктно-орієнтованого програмування (ООП). Наприклад, вона дозволяє створювати придатні для повторного використання класи, стандартизувати код за допомогою інтерфейсів, розширювати існуючі класи за рахунок успадкування і об'єднувати корисні функціональні можливості в розповсюджуваний скомпільований компонент.

Одним з найкращих прикладів підтримки об'єктно-орієнтованого поведінки в ASP .NET є серверні елементи управління. Ці елементи управління являють собою інкапсуляцію в мініатюрі. Розробники можуть маніпулювати об'єктами елементів управління програмно з використанням коду для налаштування їх зовнішнього вигляду, надання даних, що відображаються і навіть реагування на події.

Вся HTML-розмітка нижнього рівня, яку візуалізують ці елементи управління, ховається з поля зору. Замість того щоб змушувати розробника писати низькорівневий код HTML-розмітки вручну, об'єкти цих елементів управління самі перетворюються в відповідні HTML-елементи безпосередньо перед відправкою веб-сервером сторінки клієнту.

Після додавання атрибута runat = "server" цей статичний фрагмент HTML перетворюється в повністю функціональний серверний елемент управління, яким можна маніпулювати в коді. Після цього можна працювати з генеруються їм подіями, встановлювати атрибути і прив'язувати його до джерела даних.

На момент створення ASP .NET існувало два типи мислення. Одних розробників ASP .NET більше цікавили серверні елементи управління, які б в точності відповідали існуючого набору елементів управління HTML. Такий підхід дозволяє створювати інтерфейси для веб-сторінок ASP .NET в

редакторах HTML і надає швидкий шлях для міграції існуючих сторінок ASP.

Однак інші розробники ASP .NET бачили майбутнє в іншому - в багатофункціональних серверних елементах управління, які б не просто імітували окремі дескриптори HTML. Ці елементи управління могли б візуалізувати свій інтерфейс з десятків окремих елементів HTML і при цьому все одно надавати програмістам простий об'єктний інтерфейс для роботи і дозволяти їм використовувати програмовані меню, календарі, списки даних, засоби перевірки даних і т.д.

У більшості випадків серверні елементи управління HTML застосовуються для забезпечення зворотної сумісності і виконання швидкої міграції, а веб-елементи управління - для створення нових проектів.

Одним з найбільших випробувань для розробників веб-додатків є необхідність в забезпеченні підтримки безлічі різних браузерів. У різних браузерах, версіях і конфігураціях технології HTML, CSS і JavaScript підтримуються по-різному. Розробники веб-додатків повинні вибирати, візуалізувати вміст відповідно до найменшими загальними характеристиками або додавати фрагменти для прийняття до уваги специфічних особливостей популярних браузерів.

У ASP .NET ця проблема вирішена дивно інтелектуальним способом. Хоча всередині сторінки ASP .NET можна витягти інформацію про браузер клієнта і його можливості, в дійсності в ASP .NET рекомендується ігнорувати такий підхід і користуватися розвиненим комплектом серверних веб-елементів управління.

Ці серверні елементи управління генерують розмітку адаптивним чином, беручи до уваги всі можливості клієнта. Прикладом можуть служити запропоновані в ASP .NET елементи управління верифікацією, які використовують JavaScript і DHTML (динамічний HTML) для поліпшення своєї поведінки в разі, якщо вони підтримується клієнтом.

Переваги ASP .NET перед ASP:

– компіліруемий код виконується швидше, більшість помилок отлавливается ще на стадії розробки;

– значно поліпшена обробка помилок часу виконання, з використанням блоків try.catch;

– користувальницькі елементи управління (controls) дозволяють виділяти часто використовувані шаблони, такі як меню сайту;

– використання метафор, вже застосовуються в Windows-додатках, наприклад, таких як елементи управління і події;

– розширюваний набір елементів управління і бібліотек класів дозволяє швидше розробляти додатки;

– ASP .NET спирається на багатомовні можливості .NET, що дозволяє писати код сторінок на VB.NET, Delphi.NET, Visual C #, J # і т. Д.;

– можливість кешування всієї сторінки або її частини для збільшення продуктивності;

– можливість кешування даних, що використовуються на сторінці;

– можливість поділу візуальної частини та бізнес логіки по різних файлах («code behind»);

– розширувана модель обробки запитів;

– розширена подієва модель;

– розширувана модель серверних елементів управління;

– наявність master-сторінок для завдання шаблонів оформлення сторінок.

Розглянемо особливості публікації проекта на зовнішньому веб сервері. Після створення веб-додатки ASP .NET в Visual Studio 2015 розгортання проекту зазвичай виконується на веб-сервері, де інші користувачі можуть отримати доступ до додатка. Розгортання зазвичай передбачає не просто копіювання файлів програми з одного сервера на інший. Також може знадобитися виконання додаткових завдань, таких як:

– Поширення даних або структур даних в базах даних, які використовуються веб-додатком.

– Налаштування параметрів IIS на цільовому комп'ютері, таких як пул додатків, спосіб перевірки автентичності, дозвіл або заборона на перегляд каталогів і обробка помилок.

– Установка сертифікатів безпеки.

– Установка параметрів в реєстрі цільового комп'ютера.

– Установка збірок додатків в глобальному кеші складок на цільовому комп'ютері.

– Розширення для Microsoft Internet Information Services (IIS) під назвою Веб-розгортання дозволяє автоматизувати більшість завдань розгортання. Щоб спростити розгортання проекту веб-додатки, в Visual Studio передбачені засоби, що працюють з Веб-розгортання.

Користувач програми розпочинає роботу з завантаження головного вікна програми WebFormGlav (рис. 4.8), на якому розташовані такі елементи: Image, Button.

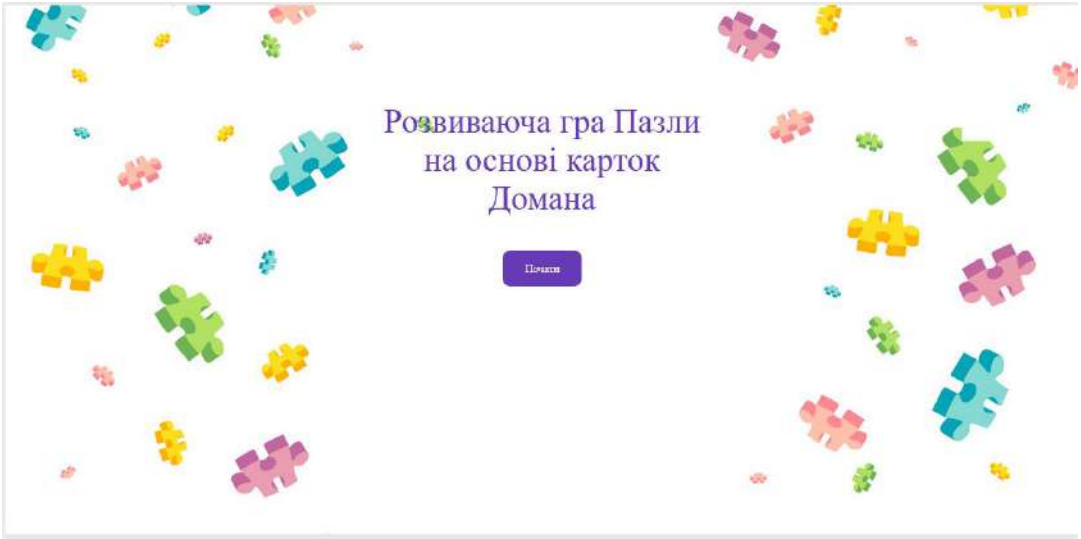

Рис. 4.8. Головне вікно програми

Натиснувши на кнопку Button1 з текстом «Почати», користувач здійснює перехід на наступну сторінку WebForm1, таким чином починаючи гру.

Розглянемо функції сторінки WebForm1 – налаштування гри (рис. 4.9). Та визначимо основні компоненти взаємодії користувача та програми.

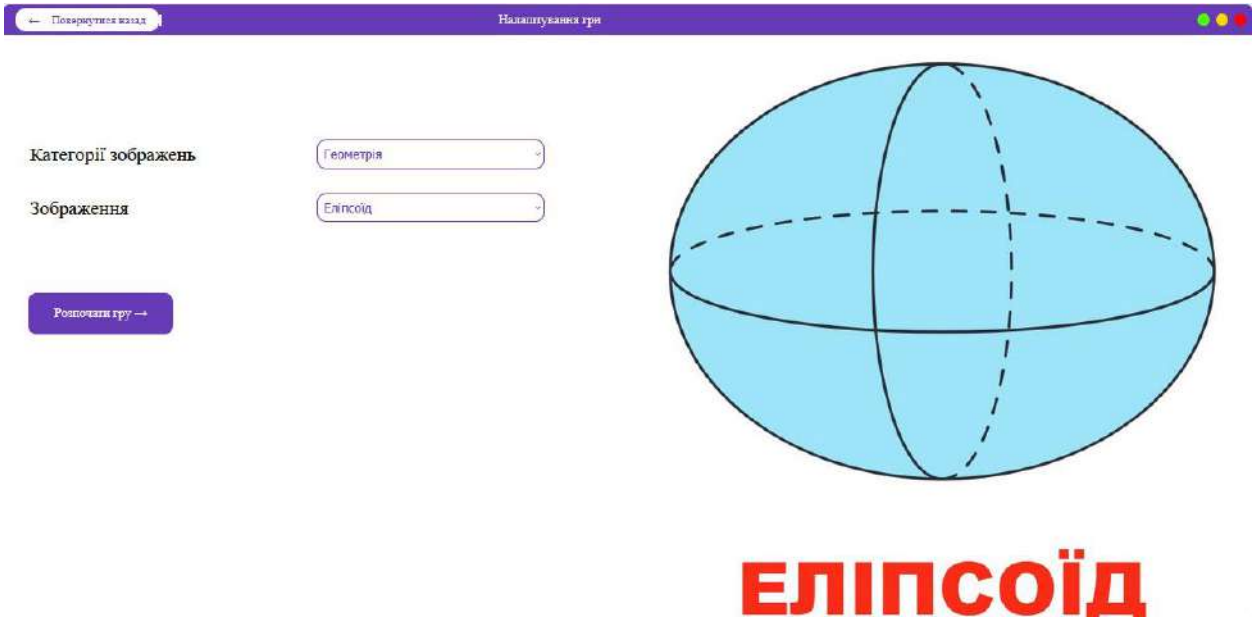

Рис. 4.9. Налаштування гри

Метод Page_Load відповідає за формування сторінки. Об'єкт OleDbConnection представляє унікальне підключення до джерела даних нашому випадку до puzzle.mdb. Властивість AutoPostBack отримує або задає значення, яке вказує, чи проводиться автоматична зворотна передача на сервер при зміні користувачем вибору елементів списку. Подія SelectedIndexChanged відповідає за відгук при виборі зображення.

Елементи управління Label1 та Label2 інформують користувача про змінні параметри, обрання категорії та обрання зображення для гри відповідно.

DropDownList – елемент управління, який дозволяє користувачеві обрати один елемент зі списку. Елементи DropDownList1 та DropDownList2 на цій сторінці відповідають за налаштування розмірів сітки та елемента пазла відповідно. В DropDownList1 за допомогою функції FormirCombo(), формується набір категорій зображень.

Функція FormirCombo() формує набір категорій зображень. У цьому методі створюються обь’єкти Connection и Recordset, що дозволяють зчитати та записати категорії зображень до об’єкту керування (в нашому випадку до DropDownList1). Метод reader.Close() перериває зчитування компонентів, cmd.Dispose() звільняє ресурси, які використовуються об’єктом Component.

Функція FormirCombo2() формує набір зображень з раніше обраної категорії. У цьому методі створюються обь'єкти Connection и Recordset, що дозволяють зчитати та записати назви зображень до об'єкту керування (в нашому випадку до DropDownList2). Метод reader.Close() перериває зчитування компонентів, cmd.Dispose() звільняє ресурси, які використовуються об'єктом Component.

DropDownList1_SelectedIndexChanged – метод обробки взаємодії з елементом керування DropDownList1 та вибору категорії зображення. DropDownList2_SelectedIndexChanged – метод обробки взаємодії з елементом керування DropDownList2 та вибору зображення з обраної раніше категорії.

Button1_Click – метод обробки події натискання на кнопку Button1, що починає гру та здійснює перехід до вікна поточної гри. WebFormGlav.katalog – формування посилання на обране зображення. Response.Redirect ("WebForm2.aspx") – переадресація користувача на сторінку WebForm2.aspx – Поточна гра (рис. 4.10).

Button2_Click – метод обробки події натискання на кнопку Button2, що починає гру та здійснює перехід до головної сторінки програми.

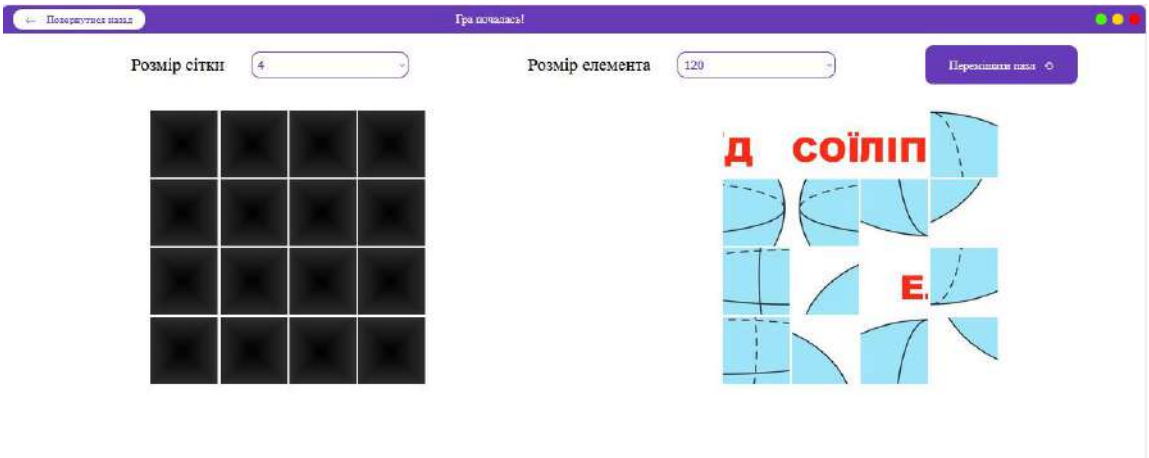

Рис. 4.10. Поточна гра

Розглянемо сторінку поточної гри, та визначимо основні компоненти взаємодії.

Завдяки компонентам Label3 та Label4 відображаються назви налаштувань. Елементи DropDownList1 та DropDownList2 на цій сторінці відповідають за налаштування розмірів сітки та елемента пазла відповідно.

Також тут знаходяться дві кнопки Button2 та Button3. Button3 дозволяє перейти на сторінку назад (сторінку налаштувань гри), а Button2 відповідає за випадкове перемішування елементів пазла.

Основні поля гри, а саме поле зіставлення пазла, та поле елементів пазла формуються завдяки двом таблицям Table1 та Table2.

Розглянемо основні функції сторінки WebForm2.aspx.

Метод Page_Load відповідає за формування сторінки. На початку методу відбувається перевірка на обрання зображення та зчитування встановленних розмірів Далі відбувається формування полей для гри та елементів пазла за допомогою методів Formir_table1 та Formir_table1.

Після цього здійснюється завантаження зображення для пазла з обраної категорії, шляхом пошуку даного зображення в директорії категорії в базі даних. Далі формуються розміри сітки та розміри елементів, які , в свою чергу записуються до компонентів взаємодії DropDownList1 та DropDownList2. Наступник кроком є остаточне формування полів для гри.

Формування поля для складення пазла здійснюється за допомогою методу Formir_table1. Спочатку здійснюється очищення поля функцією Table1.Rows.Clear(). Далі виконується цикл заповнення елементів поля. Кількість елементів визначається за обраним значенням у полі вибору розміру сітки, по тому ж принципу обирається і розмір елемента, а саме його ширина та висота. В початковому стані елементи поля замальовуються чорним кольором.

Формування поля для елементів пазла здійснюється за допомогою методу Formir_table2. Спочатку здійснюється очищення поля функцією Table2.Rows.Clear(). Далі виконується цикл заповнення елементів поля. Кількість елементів та їх розмір також визначається за обраним значенням у полі вибору розміру сітки та розміром елемента відповідно.

Наступним кроком є заповнення елементів частинками обраного зображення. На цьому етапі відбувається заповнення елементів поля частинками, що були сформовані з обраного зображення та записані у

директорію /Temp_kart. Особливістю є те, що відбувається 2 етапи заповнення, початкове та заповнення зображення. Початкове заповнення формує елементи пазлу, які вже потім заповнюються зображеннями. Усе це відбувається завдяки циклу.

Одним з найголовніших методів є формування елементів зображення для перетворення їх у частинки пазла. Саме це відбувається завдяки методу FormirPuzzle().

На початку цього методу здійснюється очищення директорії /Temp_kart. Потім виділяється пам’ять для усіх графічних елементів. Наступним кроком, є збереження частинок зображення в окремі файли для вставки їх до частинки пазла. Формування відбувається завдяки стандартному методу Picture.Clone(), а збереження завдяки методу Save().

Після цього здійснюється обов’язкове звільнення ресурсів методом Dispose().

Перемішування елементів відбувається завдяки методу Peremesh(), що викликається завдяки взаємодії з кнопкою Button2. У даному методі частинки пазлу додаються в масив, де завдяки стандартному методу rand.Next перемішуються та змінюють своє положення на полі.

Методи обробки натискання на елементи пазла та полів для складання пазла ImageButton1_Click та ImageButton2_Click є найголовнішими, адже увесь ігровий процес відбувається завдяки їм. Послідовне натискання на елементи пазлів та на елементи поля для складання переміщує їх з одного поля на інше.

Панель з повідомленням про перемогу (рис. 4.11), викликається у методі Page_Load() завдяки умові складення пазлу. І саме на цій панелі знаходиться кнопка Button3. Обробка натискання на кнопку Button3 характеризується переходом назад на сторінку (WebForm1), очищенням ігрових полів та початком нової гри.

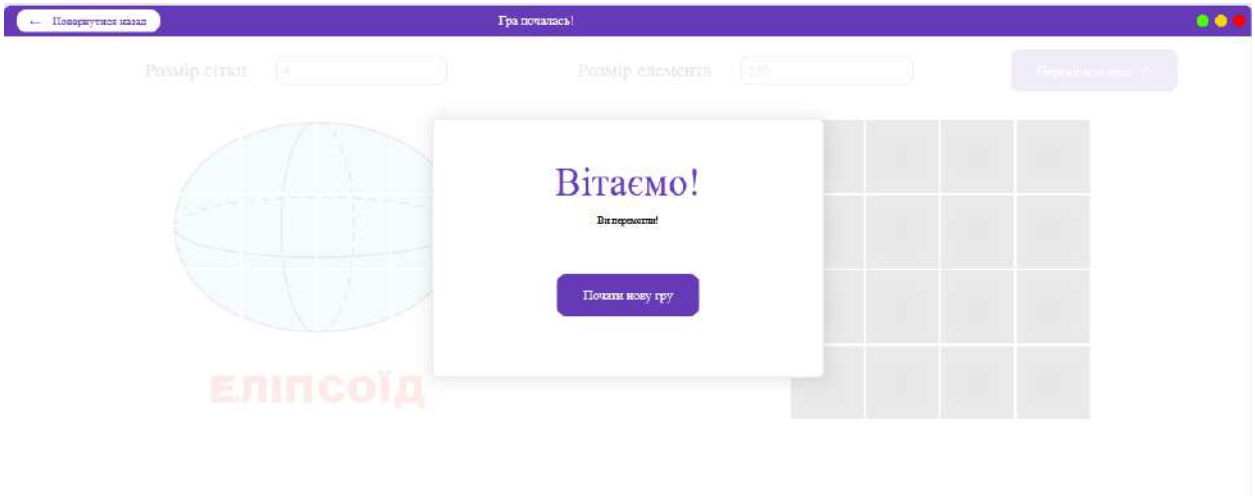

Рис. 4.11. Повідомлення про перемогу

4.3. Опис програми гри пазли «Puzzle_3D»

За допомогою мови Visual C++ та використання бібліотек OpenGL було створене 3D зображення виведення карток Домана для навчання дітей кожної з трьох груп: Геометрія, Сонячна система, Транспорт. Була передбачена робота зі світлом, накладення текстур на землю, робота з камерою та акторами.

На рис. 4.12 зображено завантажені картки Домана з геометрії.

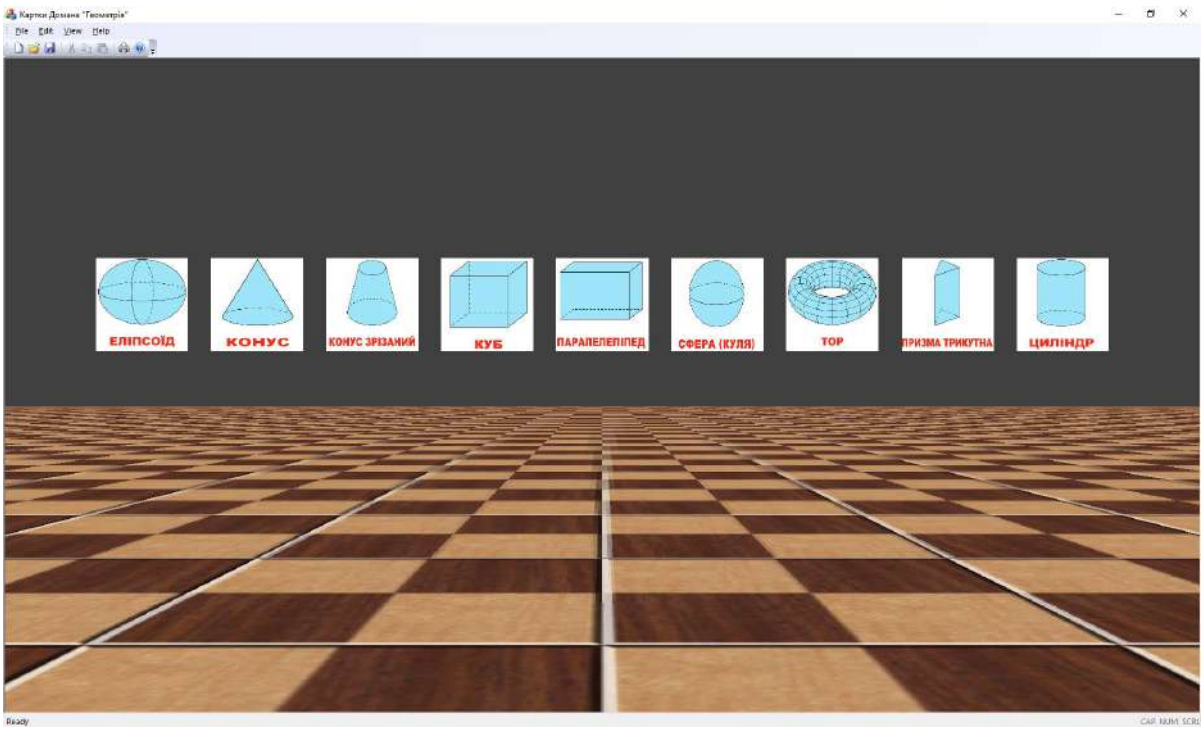

Рис. 4.12. Картки Домана «Геометрія»

Передбачена можливість пересування по сцені, як за допомогою клавіш курсору, так і за допомогою лівої кнопки мишки. Крім того, можна наближати та віддаляти зображення, використовуючи коліщатко мишки.

На рис. 4.13 зображено завантажені картки Домана «Сонячна система».

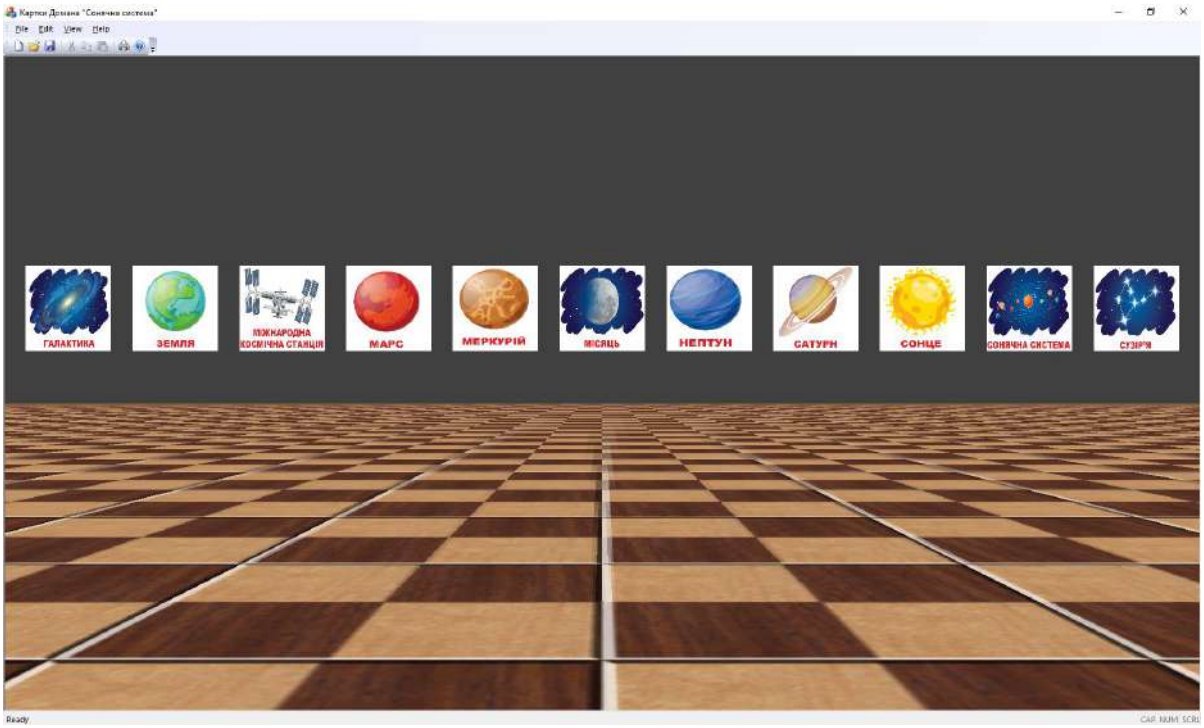

Рис. 4.13. Картки Домана «Сонячна система»

На рис. 4.14 зображено завантажені картки Домана з транспортними засобами.

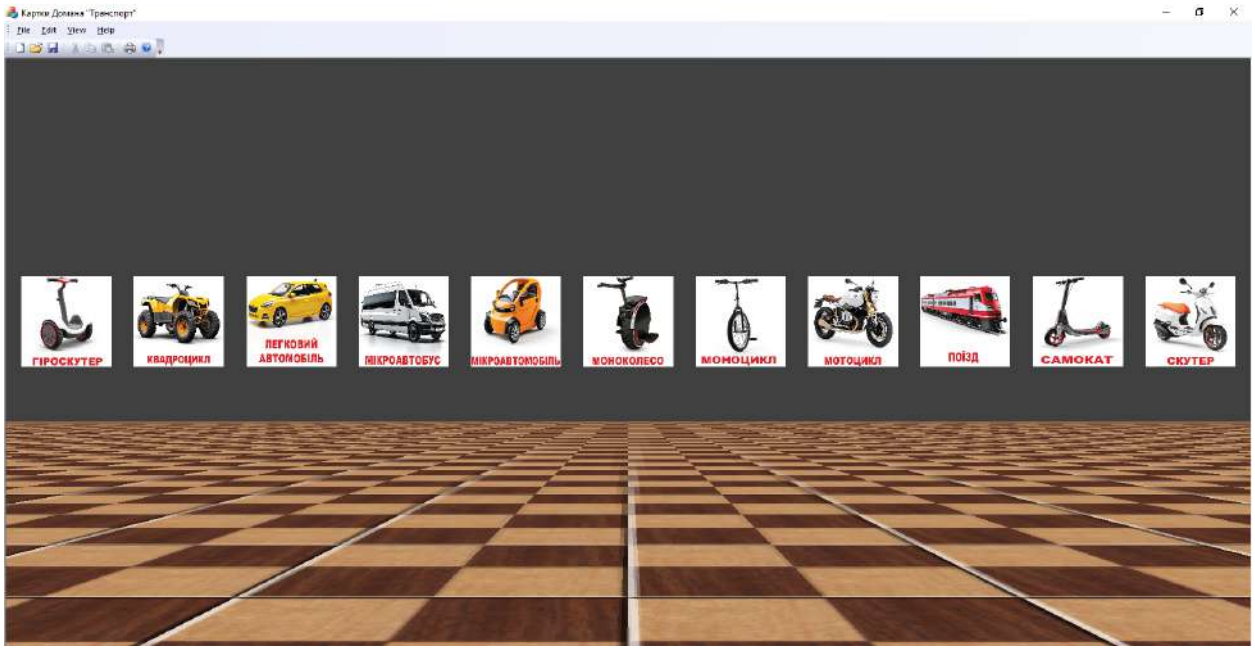

Рис. 4.14. Картки Домана «Транспорт»

Розроблений додаток «Puzzle_3D» дає можливість наочно проаналізувати вхідну інформацію у вигляді карток Домана за певними темами, які завантажуються і будуть використані для розвитку дітей дошкільного віку.

На наступному етапі зображення з кожної категорії будуть розбиватися на шматки і дитина буде їх збирати до цільного зображення, що і передбачає гра пазли.

Висновки до розділу 4

Отже у четвертому розділі продемонстроване розроблене програмне забезпечення гри пазли для розвитку дітей дошкільного віку, яке складається з трьох частин: програма «Puzzle» розроблена на C#; програма «Puzzle_3D» демонструє картки Домана по різним категоріям; у якості web-частини розроблено гру пазли «Puzzle_ASP» з використанням технології ASP .NET.

ВИСНОВКИ

У результаті виконання даної дипломної роботи було розроблене програмне забезпечення розвиваючої гри «Пазли» для дітей дошкільного віку.

Було сформульовано характеристику задачі, мету її вирішення та задачі, які вона переслідує. Була визначена структура та зміст вхідної та вихідної інформації, основні вимоги до їх оформлення, джерела, які забезпечують задачу вхідною інформацією, та особи, які мають отримати вихідну інформацію.

В процесі роботи було обрано методи розв'язання задачі та сформульовано проблеми області розробки.

Для того, щоб розв’язати перелічені проблеми й було розроблене програмне забезпечення на мові Microsoft Visual C# з використанням технологій ASP .NET.

Переваги даного програмного забезпечення:

– простий та зрозумілий інтерфейс;

– можливість обирати вже існуючі пазли, або створювати власті;

– швидкий доступ до кожного розділу меню;

– встановлення індивідуальних налаштувань під час гри;

– робота з базою даних;

Створене програмне забезпечення на сьогоднішній день є конкурентоспроможнім, задовольняючи наступний комплекс властивостей, що приваблює споживача:

– Якість продукту (програма була протестована експертами в галузі комп’ютерних технологій);

– Технічний рівень (у порівнянні із аналогами, дане програмне забезпечення є інтегрованим з іншими мовами програмування);

– Дизайн (розроблений дизайн був націлений на простоту та впевненість у виконанні дій, швидке відновлення фокусу зору).

Особливістю розробки програмного додатку технологіями ASP.NET є значна інтеграція низькорівневої мови розмітки HTML та таблиць стилів CSS. Завдяки цьому розробка інтерфейсної частини значно спрощується та дозволяє легко інтегрувати можливості мови програмування C#. Доволі важливим компонентом розроблюваного додатку є використання можливостей віддаленого доступу до програмного додатку. Таким чином, можна стверджувати, що розроблений проект є легкодоступний з будь-якої точки планети та дозволяє дистанційно проводити навчання дітей дошкільного та молодшого шкільного віку.

Що стосується інтерфейсу програмного забезпечення, при його розробці були детально проаналізовані помилки декількох схожих проектів, та навички потенційних користувачів. А завдяки засобам гнучкої інтеграції веб сторінки до різних розмірів та типів екранів можна стверджувати, що розроблений програмний додаток є доволі адаптивним та коректно відображається на усіх популярних портативних пристроїв, від планшетів до мобільних телефонів нового покоління.

Також був проведений ретельний аналіз зручності програмного додатку, на основі якого здійснено побудову ігрової логіки та навігації для зручної роботи користувача. Адже запорукою дійсно корисного програмного продукту є його зручність та інтуїтивно-зрозумілість кожного кроку взаємодії користувача з додатком.

Створена програма спрощує і оптимізує навчальний процес, робить навчання і вирішення різних задач більш цікавим і захопливим для малюків. Використання інформаційних технологій у навчанні не тільки збільшує швидкість передачі інформації дітям та підвищує рівень її засвоєння, а й сприяє розвитку таких процесів як увага, пам'ять, мислення, уява, мовлення, розвиває почуття кольору, композиції, бере участь у інтелектуальному, емоційному та моральному розвитку дітей.

Використання інформаційно-комунікативних технологій у дошкільному навчальному закладі є збагачувальним і перетворювальним

фактором, який є одним із шляхів оновлення змісту освіти згідно сучасним вимогам.

Створюючи програмний додаток, було здійснено крок для переходу до більш сучасного підходу до навчання, що передбачає значне розширення самостійної пошукової діяльності учнів та пошук учнями зв'язків між поняттями і явищами. Адже впровадження нових інформаційних технологій у навчально-виховний процес призводить до корінної зміни функцій педагога, який стає дослідником, організатором, консультантом.

На основі проведеної роботи, можна з успіхом розвивати та вдосконалювати розроблену гру, задля підвищення її функціональності. У майбутньому допускається продовжувати роботу над програмним забезпеченням, надавати їй нову функціональність, застосовувати нові технології.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агуров, Павел C#. Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 с.

2. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари, Бен Албахари. - М.: БХВ-Петербург, 2012. - 944 с.

3. Албахари, Джозеф C# 3.0. Справочник / Джозеф Албахари, Бен Албахари. - М.: БХВ-Петербург, 2013. - 944 с.

4. Альфред, В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 266 с.

5. Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2013. - 472 с.

6. Вагнер, Билл C# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.

7. Зиборов, В.В. Visual C# 2012 на примерах / В.В. Зиборов. - М.: БХВ-Петербург, 2013. - 480 с.

8. Зиборов, Виктор Visual C# 2010 на примерах / Виктор Зиборов. - М.: "БХВ-Петербург", 2011. - 432 с.

9. Ишкова, Э. А. Самоучитель C#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 с.

10. Касаткин, А. И. Профессиональное программирование на языке си. Управление ресурсами / А.И. Касаткин. - М.: Высшая школа, 2012. - 432 с.

11. Лотка, Рокфорд C# и CSLA .NET Framework. Разработка бизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.

12. Мак-Дональд, Мэтью Silverlight 5 с примерами на C# для профессионалов / Мэтью Мак-Дональд. - М.: Вильямс, 2013. - 848 с.

13. Марченко, А. Л. Основы программирования на C# 2.0 / А.Л. Марченко. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2011. - 552 с.

14. Подбельский, В. В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.

15. Прайс, Джейсон Visual C# 2.0. Полное руководство / Джейсон Прайс, Майк Гандэрлой. - М.: Век +, Корона-Век, Энтроп, 2010. - 736 с.

16. Рихтер, Джеффри CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2013. - 928 с.

17. Смоленцев, Н. К. MATLAB. Программирование на Visual C#, Borland JBuilder, VBA (+ CD-ROM) / Н.К. Смоленцев. - М.: ДМК Пресс, 2011. - 456 с.

18. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 с.

19. Троелсен, Эндрю Язык программирования C# 2008 и платформа .NET 3.5 / Эндрю Троелсен. - М.: Вильямс, 2010. - 370 с.

20. Фримен, Адам ASP.NET MVC 3 Framework с примерами на C# для профессионалов / Адам Фримен , Стивен Сандерсон. - М.: Вильямс, 2011. - 672 с.

21. Дж.Рихтер. CLR via C# . Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд./Дж.Рихтер – Питер, 2013. - 896с.

22. Дж.Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#, 2-е изд/ Дж.Рихтер - Питер, 2007. - 656 с.

23. Зеленков Ю. Введение в базы данных / Ю. Зеленков – СПб.: Питер, 2003

24. Кузнецов С.Д. Основы современных баз данных. 2-е издание испр. / С.Д.Кузнецов,– М.:Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний– 2007. – 484с.

25. Амонашвили Ш.А. «Здравствуйте, дети!» Пособие для учителя/ Ш.А.Амонашвілі -М.:Просвещение, 1983. – 203с.

26. А.К. Бондаренко. Дидактичні ігри в дитячому садку/Бондаренко А.К. - М., Просвітництво, 1991. - 300с.

27. Л.А. Венгер. "Воспитание сенсорной культуры детей"/Венгер Л.А. - М., Просвещение, 1988. – 144с.

28. Э.Г.Пилюгина. Занятие по сенсорному воспитанию с детьми раннего

возраста/ Пилюгина Э.Г. – М.:Просвещение, 1983. - 96с

29. Р.І. Жуковська. Гра і її педагогічне значення/ Жуковська Р.І. – К., 1999.

30. Н.К. Крупська.Про дошкільне виховання / Крупська Н.К.-М., 1100 г.

31. С.А. Козлова. Дошкільна Педагогіка/ Козлова С.А.-М., 2000 р.

32. А.І. Максаков. Вивчайте граючи/ Максаков А.І.-М.-1981р.

33. Д.В.Менджерицкая. Воспитателю о детской игре. Пособие для воспитателя детского сада/ Менджерицкая Д.В. - М.: Просвещение, 1982. — 128 с.

34. А.И.Сорокина. Дидактические игры в детском саду/ Сорокина А.И. - М.: Просвещение, 1982 - 98 с.

35. А.А.Столяр.Давайте поиграем: математические игры для детей 5-6 лет: книга для воспитателя детского сада/ Столяр А.А. - Москва: Просвещение, 1991. – 84 с.

ДОДАТКИ

Додаток А

Лістинг опису функцій класу «FormMain» (Головна форма програми)

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.Media;
using System.IO;
using System.Threading;
using System.Data.OleDb;
using System.Runtime.Serialization.Formatters.Binary;
using System.Collections;

namespace Puzzle
{

    public partial class FormMain : Form
    {

        //Класс настроек (сериализованный класс для записи и чтения)
        [Serializable]
        public class nastr
        {
            public int left;
            public int top;
            public int width;
            public int height;
            public bool maximize;
            ////////////////////////
            public int kol_puzzle;
            public bool musik;
        }

        //Соединение с БД
        OleDbConnection conn;

        public nastr nastr_puzzle = new nastr();


        // Подготовим генератор псевдослучайных чисел
        // для инициализации начала игры(разброс графэлементов)
        // и добавление графэелементов в процессе игры.
        Random rand = new Random();

        GraphItem[] GItems = null; // главные действующие лица, графические элементы
(пазл)
        GraphItem[] GItemsPuzzle = null; // пазл, который нужно собрать


        //Количество пазлов по одной из сторон
        public static int kol_kl = 2;

        bool formir_puzzle = false;

        // Массив объектов прямоугольников для хранения сегментов картинки.
        PictureBox PB = null;
        // Объект хранения картинки.
        Bitmap Picture = null;

        //Объект для фоновой музыки
        SoundPlayer sp_music;
```

Продовження додатку А

```
        //Конструктор
        public FormMain()
        {
            InitializeComponent();

            //Добавление музыки
            sp_music = new SoundPlayer(Properties.Resources.Детские_игры);

            //Загрузка лошадки из ресурсов
            Picture = Properties.Resources.horse;

            //Подключение к БД
            try
            {
                conn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=puzzle.mdb");
                conn.Open();
            }

            catch (Exception)
            {
                MessageBox.Show("БД puzzle.mdb отсутствует!\nПри работе с программой не
будет доступна опция БД!",
                    "Внимание!",MessageBoxButtons.OK,MessageBoxIcon.Information);

                бДToolStripMenuItem.Visible = false;
            }


            // Параметры по умолчанию
            if (!LoadSetting())
            {
                nastr_puzzle.left = this.Left;
                nastr_puzzle.top = this.Top;
                nastr_puzzle.width = this.Width;
                nastr_puzzle.height = this.Height;
                nastr_puzzle.maximize = this.WindowState == FormWindowState.Maximized;
                nastr_puzzle.musik = true;
                nastr_puzzle.kol_puzzle = 2;
            }
            else
            {
                //Применение считанных настроек
                kol_kl = nastr_puzzle.kol_puzzle;
                вклToolStripMenuItem.Checked = nastr_puzzle.musik;
                выклToolStripMenuItem.Checked = !nastr_puzzle.musik;

                this.Left = nastr_puzzle.left;
                this.Top = nastr_puzzle.top;
                this.Width = nastr_puzzle.width;
                this.Height = nastr_puzzle.height;
                if (nastr_puzzle.maximize) this.WindowState = FormWindowState.Maximized;
            }

            if (nastr_puzzle.musik) sp_music.PlayLooping();

            //FormirPuzzle();
            //Peremesh(GItems);
            //FormMain_Resize(null, null);
        }
```

```
        //Функция считывания настроек
        private bool LoadSetting()
        {
            bool voz = true;
            // Считывание ini-файла
            FileStream f = null;
            nastr vr = new nastr();
            try
            {
                f = new FileStream("puzzle.ini", FileMode.Open);
                BinaryFormatter bf = new BinaryFormatter();
                vr = (nastr)bf.Deserialize(f);
            }
            catch (FileNotFoundException)
            {
                MessageBox.Show("Файл инициализации puzzle.ini отсутствует\nДанные будут
приняты по умолчанию",
                    "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                voz = false;
            }

            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                voz = false;
            }

            finally
            {
                if (f != null) f.Close();
            }

            //Проверка ini

            if (voz)
            {
                if (vr.kol_puzzle < 2 || vr.kol_puzzle > 10) voz = false;
                if (voz) nastr_puzzle = vr;
                else MessageBox.Show("Нарушена структура файла puzzle.ini\nНастройки
приняты по умолчанию.",
                    "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            return voz;
        }
        //Функция сохранения настроек
        private void SaveSetting()
        {
            // Сохранение настроек
            FileStream f = null;
            try
            {
                f = new FileStream("puzzle.ini", FileMode.Create);
                BinaryFormatter bf = new BinaryFormatter();
                bf.Serialize(f, nastr_puzzle);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
```

Продовження додатку А

```
            finally
```

```
            {
                if (f != null) f.Close();
            }

        }

        // Отклики

        //Клик мышки на форме
        private void FormMain_MouseClick(object sender, MouseEventArgs e)
        {
            if (GItemsPuzzle == null) return;
            if (formir_puzzle) return;

            //Перемещение граф. элементов
            GraphClick(GItemsPuzzle,GItems, e.X, e.Y);

            //Проверка на совпадение пазла
            if (Proverka(GItemsPuzzle))
            {
                formir_puzzle = true;

                Stream res = Properties.Resources.tush;
                SoundPlayer sp = new SoundPlayer(res);
                sp.PlaySync();

                if (nastr_puzzle.musik) sp_music.PlayLooping();

                DialogResult dd = MessageBox.Show("Поздравляем! Вы выиграли!\nВы хотите
начать новую игру?",
                    "Победа!", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

                if (dd == DialogResult.Yes)
                {
                    FormirPuzzle();
                    Peremesh(GItems);
                    FormMain_Resize(null, null);
                }

            }
        }


        //Рисование
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics;

            if (GItems == null) return;

            // Рисование элементов
            for (int i = 0; i < GItems.Length; i++)
            {
                GItems[i].Draw(g);
                GItemsPuzzle[i].Draw(g);
            }


        }
```

Продовження додатку А

```
        // Изменение размеров окна
        private void FormMain_Resize(object sender, EventArgs e)
```

```
        {
            // При изменении размеров окна корректируем
            // координаты сетки поля и положения граф.элементов.
            if (GItems == null) return;

            ComputeCellsCoordinate(GItemsPuzzle, 1);
            ComputeCellsCoordinate(GItems, 2);
        }

        //Новая игра
        private void toolStripButtonNewGame_Click(object sender, EventArgs e)
        {
            FormirPuzzle();
            Peremesh(GItems);
            FormMain_Resize(null, null);
        }


        //Пользовательские функции

        // Функция формирования пазла и пустого массива
        private void FormirPuzzle()
        {
            // Перед загрузкой формы выделим память для всех графических элементов.

            int i, countX = 0;
            int countY = 0;

            int w = Picture.Width / kol_kl;
            int h = Picture.Height / kol_kl;

            int NumGraphItems = kol_kl * kol_kl;

            // Удаление старых картинок
            if (GItems != null)
                for (i = 0; i < GItems.Length; i++)
                    if (GItems[i].Puzzle != null) GItems[i].Puzzle.Dispose();

            GItems = new GraphItem[NumGraphItems];

            //Формирование пустого массива
            GItemsPuzzle = new GraphItem[NumGraphItems];


            for (i = 0; i < NumGraphItems; i++)
            {
                GItems[i] = new GraphItem(this);
                ////////////////////////////////
                GItemsPuzzle[i] = new GraphItem(this);

                GItems[i].Visible = true;

                // Разрезание картинки и помещение ее в массив PictureBox

                // Размеры и координаты размещения созданного прямоугольника.
                PB = new PictureBox();
                // Копирование части картинки
                PB.Image = Picture.Clone(new RectangleF(countX * w, countY * h, w, h),
Picture.PixelFormat);
                ////////////////////////////////////////////
```

Продовження додатку А

```
                countY++;
                if (countY == kol_kl)
                {
```

```
                    countY = 0;
                    countX++;
                }

                PB.Tag = i; // сохраним индекс элемента

                PB.BorderStyle = BorderStyle.None;
                PB.SizeMode = PictureBoxSizeMode.StretchImage; // размеры картинки будут
подгоняться под размеры прямоугольника

                //Привязка картинки к элементу
                GItems[i].Puzzle = PB;
            }
        }

       //Вывод и пересчет графических элементов для 1-го режима
        // GItems - массив граф. элементов
        // kod = 0 - 1 вывод
        // kod = 1 - вывод слева
        // kod = 2 - вывод справа

        void ComputeCellsCoordinate(GraphItem[] GItems, int kod)
        {

            int num = kol_kl;

            int lenside = 0;// длина стороны квадрата области расположения графэлементов

            // используем размеры клиентской области
            int delta = 20;
            int width  = 0;
            int nach = 0;
            if (kod == 0) width = this.ClientRectangle.Width - delta;
            if (kod == 1 || kod == 2) width = this.ClientRectangle.Width / 2 - delta;
            if (kod == 2) nach =  this.ClientRectangle.Width / 2;

            int height = this.ClientRectangle.Height -delta - panel1.Top;

            // длина стороны области графэлементов чуть меньше клиентской высоты
            lenside = (height < width) ? height : width;

            // длина стороны ячейки
            int lenCell = lenside / num;

            // по колонно расчитываем расположение ячеек графэлементов
            int x,y,count = 0;
            for (x = 0; x < num; x++)
            {
                for (y = 0; y < num; y++)
                {
                    GItems[count].CellCoordinate = new Rectangle(nach + x * lenCell + 10,
y * lenCell + delta / 2 + panel1.Top, lenCell, lenCell);
                    count++;
                }
            }
            // В итоге ячейки расмещены в таком порядке:
            ////////////////////////////////
            // 0 10 20 30 40 50 60 70 80 90 //
            // 1 11 21 31 41 51 61 71 81 91 //
```

Продовження додатку А

```
            // 2 12 22 32 42 52 62 72 82 92 //
            // 3 13 23 33 43 53 63 73 83 93 //
            // 4 14 24 34 44 54 64 74 84 94 //
            // 5 15 25 35 45 55 65 75 85 95 //
```

```
            // 6 16 26 36 46 56 66 76 86 96 //
            // 7 17 27 37 47 57 67 77 87 97 //
            // 8 18 28 38 48 58 68 78 88 98 //
            // 9 19 29 39 49 59 69 79 89 99 //
            ////////////////////////////////

            Invalidate();
        }


        //Перемещение или удаление граф. элемента в области
        //GItems1 - граф. элементы 1, GItems2 - граф. элементы 2, X, Y - точка
        void GraphClick(GraphItem[] GItems1,GraphItem[] GItems2, int X, int Y)
        {
            int indexClick = -1; // индекс кликнутого граф-элемента
            int i, p, a;

            GraphItem[] GItems = null;
            GraphItem[] GItemsA = null;

            // Если кликнули по видимому графэлементу(с геометрической фигурой), сделаем
его активным,
            // т.е. мигающим размером и выходим из функции.

            for (i = 0; i < GItems1.Length; i++)
            {
                if (GItems1[i].CellCoordinate.Contains(X, Y) == true ||
GItems2[i].CellCoordinate.Contains(X, Y) == true)
                {
                    indexClick = i;
                    if (GItems1[i].CellCoordinate.Contains(X, Y) == true) GItems =
GItems1;
                    if (GItems2[i].CellCoordinate.Contains(X, Y) == true) GItems =
GItems2;

                    if (GItems[i].Visible == true)
                    {
                        for (p = 0; p < GItems1.Length; p++)
                        {
                            GItems1[p].Active = false;
                            GItems2[p].Active = false;
                        }

                        GItems[i].Active = true;
                        return;
                    }
                }
            }


            // Если кликнули по невидимому графэлементу(без геометрической фигуры),
проводим нижеследующие
            // процедуры,
            // 1. находим активный графэлемент (если его нет выполнится просто пустой
цикл)
            // 2. проверим может ли активный графэлемент переместится на указаное
            // пользователем место.
            // 3. меняем местами и свойствами активный и невидимый графэлементы.
```

Продовження додатку А

```
            for (a = 0; a < GItems1.Length; a++)
            {
                if (GItems1[a].Active == true || GItems2[a].Active==true)
                {
                    if (indexClick != -1)
```

```
                {

                        if (GItems1[a].Active == true) GItemsA = GItems1;
                        if (GItems2[a].Active == true) GItemsA = GItems2;

                        // Меняем местами и свойствами вычисленные графэлементы
                        PictureBox puzzle = GItems[indexClick].Puzzle;
                        GItems[indexClick].Puzzle = GItemsA[a].Puzzle;
                        GItems[indexClick].Active = false;
                        GItemsA[a].Puzzle = puzzle;
                        GItemsA[a].Visible = false;
                        GItemsA[a].Active = false;
                        GItems[indexClick].Visible = true;
                        Invalidate();

                        // Делаем небольшую задержку данной функции, без задержки
работы приложения,
                        // для акцентирования внимания пользователя на перемещение
графэлемента.
                        Application.DoEvents();
                        Thread.Sleep(80);

                        Invalidate();

                }
                break;
            }
        }

    }

    // Перемешивание графических элементов
    private void Peremesh(GraphItem[] GItems)
    {
        PictureBox t;
        int i, r = 0;
        for (i = 0; i < GItems.Length; i++)
        {
            r = rand.Next(0, GItems.Length);
            t = GItems[r].Puzzle;
            GItems[r].Puzzle = GItems[i].Puzzle;
            GItems[i].Puzzle = t;
        }

        formir_puzzle = false;
    }

    //Функция проверки на совпадение пазла
    private bool Proverka(GraphItem[] GItems)
    {
        for (int i = 0; i < GItems.Length; i++)
        {
            if (!GItems[i].Visible) return false;
            if (i != (int)GItems[i].Puzzle.Tag) return false;
        }
        return true;
    }
```

Продовження додатку А

```
    //Загрузка из файла
    private void загрузитьToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //openFileDialog1.InitialDirectory = Directory.GetCurrentDirectory();
```

```
            openFileDialog1.Filter = "Файлы картинок
(*.bmp;*.jpg;*.jpeg;)|*.bmp;*.jpg;.jpeg";
            if (openFileDialog1.ShowDialog() != DialogResult.OK) return;


            try
            {
                ////////////////////////////////////
                if (Picture != null) Picture.Dispose();
                Picture = new Bitmap(openFileDialog1.FileName);
                FormirPuzzle();
                Peremesh(GItems);
                FormMain_Resize(null, null);
            }
            catch (Exception)
            {
                MessageBox.Show("Несответствие формата файла!\nВыберите
картинку!","Ошибка!",MessageBoxButtons.OK, MessageBoxIcon.Stop);
            }
        }

        // Перемешивание пазла
        private void перемешиваниеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Peremesh(GItems);
            Invalidate();
        }


        //Загрузка формы сетки
        private void сеткаToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            FormSetka f = new FormSetka();
            f.kol = kol_kl;
            if (f.ShowDialog() == DialogResult.OK)
            {
                kol_kl = f.kol;
                FormirPuzzle();
                Peremesh(GItems);
                FormMain_Resize(null, null);
            }
        }
        //Включение музыки
        private void вклToolStripMenuItem_Click(object sender, EventArgs e)
        {
            nastr_puzzle.musik = true;
            вклToolStripMenuItem.Checked = true;
            выклToolStripMenuItem.Checked = false;
            sp_music.PlayLooping();
        }

        //Выключение музыки
        private void выклToolStripMenuItem_Click(object sender, EventArgs e)
        {
            nastr_puzzle.musik = false;
            выклToolStripMenuItem.Checked = true;
```

Продовження додатку А

```
            вклToolStripMenuItem.Checked = false;
            sp_music.Stop();
        }

        //Подсказка
        private void подсказкаToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            FormPict f = new FormPict();
            f.pictureBox1.Image = Picture;

            f.ShowDialog();
        }

        //Закрытие формы
        private void FormMain_FormClosing(object sender, FormClosingEventArgs e)
        {

            //Считывание данных
            nastr_puzzle.maximize = this.WindowState == FormWindowState.Maximized;
            if (nastr_puzzle.maximize) this.WindowState = FormWindowState.Normal;
            nastr_puzzle.left = this.Left;
            nastr_puzzle.top = this.Top;
            nastr_puzzle.width = this.Width;
            nastr_puzzle.height = this.Height;
            nastr_puzzle.musik = вклToolStripMenuItem.Checked;
            nastr_puzzle.kol_puzzle = kol_kl;

            //Запись данных
            SaveSetting();
        }

        //Просмотр БД
        private void просмотрБДToolStripMenuItem_Click(object sender, EventArgs e)
        {
            FormBD bd = new FormBD();
            bd.conn = conn;
            bd.ShowDialog();
        }

        //Загрузка картинки из БД
        private void загрузитьКартинкуToolStripMenuItem_Click(object sender, EventArgs e)
        {
            FormBD bd = new FormBD();
            bd.conn = conn;
            bd.kod = 2;
            if (bd.ShowDialog() == DialogResult.OK)
            {
                Picture = new Bitmap(bd.pictureBox1.Image);
                FormirPuzzle();
                Peremesh(GItems);
                FormMain_Resize(null, null);
            }
        }


    }
```

Додаток Б

Лістинг опису функцій класу «FormBD» (Перегляд бази даних та вибір картинки з БД)

```
System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Puzzle
{
    public partial class FormBD : Form
    {

        public OleDbConnection conn;
        public int kod = 1;
        OleDbDataAdapter da;
        DataSet ds;


        public FormBD()
        {
            InitializeComponent();
            ds = new DataSet();
        }

        //Заполнение combobox и привязка данных
        private void FormBD_Load(object sender, EventArgs e)
        {

            if (kod == 1)
            {
                button1.Visible = false;
                Text = "Просмотр БД";
            }
            if (kod == 2)
            {
                button1.Visible = true;
                Text = "Выбор БД";
            }


            //Создание объекта Connection и Recordset
            try
            {
                string select = "SELECT Cat, ID FROM Category";

                OleDbCommand cmd = new OleDbCommand(select, conn);
                OleDbDataReader reader = cmd.ExecuteReader();

                while (reader.Read())
                    comboBox1.Items.Add(new Name(reader[0].ToString(), (int)reader[1]));
                reader.Close();
                cmd.Dispose();
                comboBox1.SelectedIndex = 0;
```

Продовження додатку Б

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }

        //Выбор элемента
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                if (da != null) da.Dispose();
                pictureBox1.DataBindings.Clear();
                bindingSource1.DataSource = null;

                string str = "Select Picture From Picture where ID_Cat = " +
((Name)comboBox1.SelectedItem).ItemData;
                da = new OleDbDataAdapter(str, conn);
                ds.Tables.Clear();

                da.Fill(ds);

                bindingSource1.DataSource = ds.Tables[0];
                bindingNavigator1.BindingSource = bindingSource1;

                //Привязка picturebox к изображению
                pictureBox1.DataBindings.Add("Image", bindingSource1, "Picture", true);

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }

        private void button1_Click(object sender, EventArgs e)
        {
            DialogResult = System.Windows.Forms.DialogResult.OK;
        }


    }
}
```

Додаток В

Лістинг опису функцій класу «GraphItem» (Формування власного елементу керування)

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

namespace Puzzle
{
    // Класс графического элемента, состоит из квадратной ячейки
    // и геометрической фигуры.
    // Геометрическая фигура может принимать разные формы,
    // при выборе игроком демонстрировать активность.
    public class GraphItem
    {
        // Удобство статической переменной - ее изменение в любом месте
        // программы приведет к изменению вида геометрической фигуры во всех
        // объектах класса GraphItem
        //public static TypeGraphItem CurrentTypeGraphItem;

        // Вспомогательное свойство, требуется для правильного
        // изменения цвета геометрической фигуры игроком через окно настроек.
        public int Tag = -1;

        private PictureBox puzzle=null;

        //Свойство PictureBox
        public PictureBox Puzzle
        {
            get { return puzzle;}
            set { puzzle = value; }
         }

        // Координаты ячейки граф.элемента.
        public Rectangle CellCoordinate;

        // Цвет для геометрической фигуры.
        public Color Color = Color.Gray;
        //public enum TypeGraphItem { tRhombus, tEllipse, tRectangle };

        // Нам нужен только метод Invalidate() родительского окна данный метод
наследуется от класса Control.
        // Используется в таймерах активности и исчезания.
        Control Parent = null;

        // Таймеры нам дают псевомногопоточную работу приложения.
        Timer timerActive = new Timer();
        Timer timerVanish = new Timer();
        /// <summary>
        /// Заказной конструктор
        /// </summary>
        /// <param name="parent">ссылка на родительское окно</param>
        public GraphItem(Control parent)
        {
            Parent = parent;
            // ---- Инициализация таймеров ----
            // таймер активности геометрической фигуры
            timerActive.Interval = 40;
```

Продовження додатку В

```
            timerActive.Tick += new EventHandler(timerActive_Tick);
            timerActive.Enabled = false;

            // таймер визуального исчезания геометрической фигуры
            timerVanish.Interval = 40;
            timerVanish.Tick += new EventHandler(timerVanish_Tick);
            timerVanish.Enabled = false;
            deltaWidth = inflateSize;
            deltaHeight = inflateSize;
        }


        #region Рисование графэлемента в родительском окне
        /// <summary>
        /// Декоративная кисть ячейки.
        /// </summary>
        /// <returns>готовая кисть</returns>
        Brush BrushCell()
        {
            // Подготовка координат для прорисовки ромба
            Point point1 = new Point(CellCoordinate.Left + CellCoordinate.Width / 2,
CellCoordinate.Top);
            Point point2 = new Point(CellCoordinate.Left + CellCoordinate.Width,
CellCoordinate.Top + CellCoordinate.Height / 2);
            Point point3 = new Point(CellCoordinate.Left + CellCoordinate.Width / 2,
CellCoordinate.Top + CellCoordinate.Height);
            Point point4 = new Point(CellCoordinate.Left, CellCoordinate.Top +
CellCoordinate.Height / 2);

            Point[] pt = { point1, point2, point3, point4 };

            // Каков вид геометрической фигуры такая и кисть.
            GraphicsPath gp = new GraphicsPath();

            gp.AddRectangle(CellCoordinate);



            // Раскраска геометрической фигуры.
            PathGradientBrush pathBrush = new PathGradientBrush(gp);
            pathBrush.SurroundColors = new Color[] { Color.FromArgb(50, 50, 50) };
            pathBrush.CenterPoint = new PointF(CellCoordinate.Left + CellCoordinate.Width
/ 2, CellCoordinate.Top + CellCoordinate.Height / 2);
            pathBrush.CenterColor = Color.Gray;

            return pathBrush;
        }

        int inflateSize = -4; // переменная величины пульсирования геометрической фигуры
        int deltaWidth = 0;   // изменения по ширине
        int deltaHeight = 0;  // изменение по высоте
        /// <summary>
        /// Рисование графики в графическом контексте.
        /// </summary>
        /// <param name="g">graphics окна вывода</param>
        public void Draw(Graphics g)
        {

            g.FillRectangle(BrushCell(), CellCoordinate.X + 1,
                CellCoordinate.Y + 1, CellCoordinate.Width - 1, CellCoordinate.Height -
1);
```

Продовження додатку В

```
    // При невидимой геометрической фигуре прорисовывается только квадрат ячейки.
    if (visible == false) return;

    // Вспомогательный квадрат для эффекта исчезания и эффекта
    // активности в выбранном состоянии.
    Rectangle rectInflate = CellCoordinate;
    rectInflate.Inflate(deltaWidth, deltaHeight);

    //Вывод картинки

    if (rectInflate.Width < 0 || rectInflate.Height < 0) return;

    if (Puzzle!=null) g.DrawImage(Puzzle.Image, rectInflate);
}

#endregion

#region Свойства графэлемента - видимости, активности, исчезания

bool visible = false;
/// <summary>
/// Невидимым считается даже исчезающая геометрическая фигура.
/// </summary>
public bool Visible
{
    get
    {
        if (vanish == true || visible == false)
            return false;
        return true;
    }
    set
    {
        visible = value;
    }
}
bool active = false;
/// <summary>
/// Свойство активности геометрической фигуры.
/// </summary>
public bool Active
{
    get
    {
        return active;
    }
    set
    {
        active = value;

        if (active == true)
        {
            timerActive.Enabled = true;
        }
        else
        {
            timerActive.Enabled = false;
            deltaWidth = inflateSize;
            deltaHeight = inflateSize;
        }
    }
}
```

Продовження додатку В

```
// Формируем свойство исчезания.
```

```
        bool vanish = false;
        /// <summary>
        /// Свойство плавного исчезания геометрической фигуры.
        /// </summary>
        public bool Vanish
        {
            get
            {
                return vanish;
            }
            set
            {
                vanish = value;
                if (vanish == true)
                {
                    timerVanish.Enabled = true;
                }
            }
        }
        #endregion

        #region Таймеры графэлемента - таймер активности и таймер исчезания
        int k = 1;
        /// <summary>
        /// Таймер активности, периодически увеличивает и уменьшает геометрическую
фигуру,
        /// создавая эффект активности.
        /// </summary>
        void timerActive_Tick(object sender, EventArgs e)
        {
            deltaWidth += 1 * k;
            deltaHeight += 1 * k;
            if (deltaWidth >= -2)
                k = -1;
            else if (deltaWidth <= -7)
                k = 1;
            Parent.Invalidate(CellCoordinate);
        }
        /// <summary>
        /// Таймер исчезания, сначала графэлемент уменьшается,
        /// а потом исчезает.
        /// </summary>
        void timerVanish_Tick(object sender, EventArgs e)
        {
            deltaWidth += -2;
            deltaHeight += -2;
            if (deltaWidth < -14)
            {
                timerVanish.Enabled = false;
                visible = false;
                vanish = false;
                deltaWidth = inflateSize;
                deltaHeight = inflateSize;
            }

            Parent.Invalidate(CellCoordinate);
        }
        #endregion
    }
}
```