МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Кафедра комп’ютерної інженерії та кібер безпеки

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Об’єктно-орієнтоване програмування»

на тему:

**Гра «Водопровідник»**

студента І курсу групи КІ-20-1

спеціальності 123 «Комп’ютерна інженерія»

Овсеюков Єгор Юрійович\_\_ \_\_\_\_\_\_\_\_\_

(прізвище, ім’я та по-батькові)

Керівник:

Старший викладач Левківський В.Л ,

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський \_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Марчук .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Чижмотря \_

(підпис) (прізвище та ініціали)

Житомир – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 123 «Комп’ютерна інженерія»

«ЗАТВЕРДЖУЮ»

В.о.зав. кафедри

\_\_\_\_\_\_\_\_\_\_\_\_ А.В.Морозов

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Овсеюкову Єгору Юрійовичу

1. Тема роботи: Розробка гри «водопровідник»,

керівник роботи: Старший викладач Левківський Віталій Леонідович

1. Строк подання студентом: “ 25 ” травня 2020р.
2. Вихідні дані до роботи: Розробити гру «водопровідник»
3. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
   * + 1. Постановка завдання
       2. Аналіз аналогічних розробок
       3. Алгоритми роботи програми

4. Опис роботи програми

5. Програмне дослідження

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)

1. Презентація до КП\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. Посилання на репозиторій: https://gitlab.com/LemonsLover1/plumber-game.git

Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Левківський В.Л., ст. викладач каф. КН |  |  |
| 1,2 | Марчук Г.В., ст. викладач каф. КН |  |  |
| 1,2 | Чижмотря О.В., ст. викладач каф. ІПЗ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “ 15 ” квітня 2021 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсового проекту | Строк виконання етапів проекту | Примітки |
| 1 | Постановка задачі | 25.03.2021 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 30.03.2021 |  |
| 3 | Формулювання технічного завдання | 05.04.2021 |  |
| 4 | Опрацювання літературних джерел | 10.04.2021 |  |
| 5 | Проектування структури | 20.04.2021 |  |
| 6 | Написання програмного коду | 10.05.2021 |  |
| 7 | Відлагодження | 20.05.2021 |  |
| 8 | Написання пояснювальної записки | 15.06.2021 |  |
| 9 | Захист | 31.06.2021 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Овсеюков Є.Ю

(підпис) (прізвище та ініціали)

**Керівник проекту** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Левківський В.Л

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Завданням курсового проекту (роботу) була розробка гри «водопровідник»

Пояснювальна записка до курсового проекту (роботи) на тему гри «водопровідник» складається з вступу, трьох розділів, висновків, списку використаної літератури та додатків.

Текстова частина викладена на 20 сторінках друкованого тексту. Пояснювальна записка має 26 сторінок додатків. Список використаних джерел містить 10 найменувань і займає 1 сторінку. В роботі наведено 15 рисунки Загальний обсяг роботи – 46 сторінок.

Ключові слова: С#, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ГРА, ВОДОПРОВІДНИК, ООП.

ЗМІСТ

ВСТУП6

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ7

1.1 Аналіз задачі, засобів та методів її вирішення7

1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекту (роботи) 7

Висновки до першого розділу 11

РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ11

2.1 Проектування загального алгоритму роботи програми11

2.2 Розробка функціональних алгоритмів роботи програми11

2.3 Розробка програмного забезпечення12

Висновки до другого розділу 12

РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ13

3.1 Опис роботи з програмним додатком (Опис інтерфейсу)13

Висновки до третього розділу 18

ВИСНОВКИ18

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ19

ДОДАТКИ20

ВСТУП

Ігри у сучасному світі є невід’ємною частиною життя великої кількості людей, вони допомагають скоротити час після роботи та розвивають багато навичок у цікавій формі. Одним із жанрів ігор є ігри-головоломки, вони ідеальні коли хочеться скоротати невелику кількість часу розв’язуючи якісь цікаві задачі. Гра «водопровідник» як раз одна з таких, шукання рішень для рівнів може затягнути людину на деякий час, при цьому розвиваючі логічне мислення.

Мова програмування С# є однією з найкращих ООП мов, що робить її дуже зручною для розробки великої кількості додатків. Одна з головних переваг C# є велика кількість зручних бібліотек та фреймворків які дозволяють сильно покращити швидкість та якість написання коду.

Цей курсовий проект включає в себе розробку гри, яка була реалізована можливостями C# та Windows Forms.

**Об'єкт дослідження:** Обектно орієнтовні можливості Windows Forms

**Предмет дослідження:** Реалізація гри «водопровідник» з використанням можливостей Windows Forms

**Метою курсової роботи є** розробка гри «водопровідник» відповідно до технічного завдання.

**Програмний продукт** працює під управлінням операційних систем сімейства Windows. Додаток розроблений за допомогою об’єктно-орієнтованої технології програмування в середовищі програмування Visual Studio на мові C#.

**Основною перевагою** зберігання досягнень гравця та можливість створювати власні рівні.

**Передбачувана область застосування:** гра може буду застосована для розвитку логічного мислення дітей та для проявлення креативного мислення в редакторі рівнів.

**РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ**

**1.1. Аналіз задачі, засобів та методів її вирішення**

Гра «водопровідник» – спрямована на розвиток логічного мислення шляхом розваг.

Завданням курсового проекту є створення гри «водопровідник», яке буде мати наступні функції :

-зберігання поточних досягнень гравця

-котонізацію(вибір шапок) головного персонажа

-створення власних рівнів

-зберігання власних рівнів в файл для обміну

-прогресія у виді можливості відкриття шапок для персонажа

.

Гра повинна мати чітку прогресію та заохочування гравця шляхом розблокування нового контента за прогресію в грі.

Гра була виконана з допомогою Visual Studio 2019 та Windows Forms.

Для зберігання рівнів використовувалась бібліотека newtonsoft для зручного серіалізування/десіріарізування рівнів в Json.

**1.2. Аналіз існуючого програмного забезпечення за тематикою курсового проекту (роботи).**

На даний момент часу найбільш популярні ігри про «водопровідника» є :

Plumber - У найгарячішій пустелі вода - дефіцитний ресурс. Маленькій квітці потрібна вода, щоб вижити. Побудуйте трубопровід і піднесіть воду, щоб врятувати квітку. Розкрийте свої навички водопровідника в цій безкоштовній грі-головоломці.(рис. 1.1)( <https://play.google.com/store/apps/details?id=com.tedrasoft.plumber&hl=uk&gl=US>)



рис. 1.1

Переваги:

-наявність гри у плей маркеті

-більш продуктивний рушій

-власні зображення

Недоліки:

-існує тільки на андроїді

-немає костомізації рівнів

-немає костомізації персонажу

Водопровідник 3 - Здається, що технології стали більш досконалими, ніж будь-коли, і традиційні ігри, в які ми грали кілька десятиліть тому, втрачають свою чарівність і привабливість. Будучи дітьми, ми всі хотіли стати пожежниками, поліцейськими або водопровідниками. Сьогодні навряд чи ви мрієте бути сантехніком - принаймні поки не спробуєте грати в Водопровідник 3 !! .(рис. 1.2)

(<https://play.google.com/store/apps/details?id=com.Appholdings.Plumber3&hl=ru&gl=US>)



рис. 1.2

Переваги:

-наявність гри у плей маркеті

-більш продуктивний рушій

-велика кількість рівнів

-власні зображення

Недоліки:

-існує тільки на андроїді

-немає костомізації персонажу

The Plumber Game - Надміцна витік, і сантехнік повинен усунути її протягом певного часу. Ця гра-головоломка включає в себе таймер цокання, ситуацію з витоком, подібну повені, і окремі частини трубопроводу, які необхідно виправити, щоб зупинити повінь. Скачайте і встановіть цю кумедну маленьку гру і насолоджуйтеся розблокуванням всіх 30 рівнів один за іншим. .(рис. 1.3)

(<https://play.google.com/store/apps/details?id=plumber.pixellicious.game&hl=ru&gl=US>)

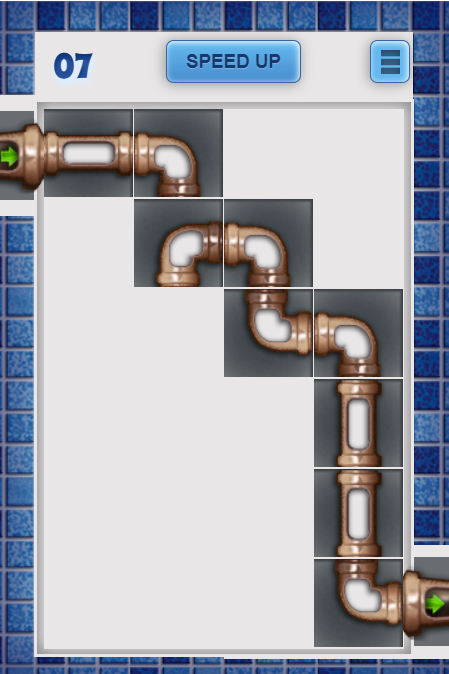


рис. 1.3

Переваги:

-наявність гри у веб-форматі

Недоліки:

-існує тільки в веб-форматі

-маленька кількість рівнів

-низька продуктивність рушія

-немає костомізації персонажу

**Висновки до першого розділу:**

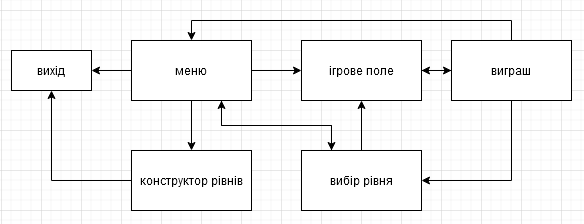
Після аналізу схожих проектів, їх переваг та недоліків, я зробив для себе важливі висновки. Я визначився з кількістю можливостей які важливо реалізувати у своїй грі та чітко зрозумів як має виглядати подібна гра.

**РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**2.1 Проектування загального алгоритму роботи програми**

Після етапу аналізу та чіткого уявлення поставленої задачі, можна перейти то проектування алгоритму роботи програми.

На рис. 2.1. зображена схема програми, то як окремі частини програми будуть взаємодіяти між собою.



Складней алгоритм роботы програми ддає нам можливість оцінити складність та зробити висновки щодо подальшої розробки.

Однією із важливих можливостей користувача буде створювання своїх власних рівнів використовуючи конструктор рівнів. Кастомні рівні, разом із основними, будуть розташовані на формі вибора рівнів. Також особливістю гри повинна стати можливість розблокування нового контенту.

* 1. **Розробка функціональних алгоритмів роботи програми**

На діаграмі класів ми бачимо клас Sprite та його наслідників Character та Hat.

Справа зверху ми бачимо клас GameField та форми які з ним контактують. Зліва знизу знаходяться класи які працюють з рівнями та форма LevelCreator яка створює нові рівні. В центрі знизу знаходяться форми які спрямовані на відображення персонажа та розблокованих шапок. Справа знизу розміщені стандартні класи програми, Resources призначений для збереження ресурсів проекту та Settings який я використовую для збереження прогресу гравця після

закриття програми.



У програми є основних клас (GameField) у якому відбуватися основна логіка гри. Відбувається перевірка кожного шагу гравця та обробляєтеся логіка труб.

Клас Levels, спрямованийй на зручну роботу з рівнями гри, у ньому зберігайтеся список основних рівнів та зберігаються/зчитуються кастомні рівні. Список рівнів являє собою список об’єктів класу Level у яркому описуються параметры рівня. Клас Level може бути серіалізований та має 5 полів(Name – ім’я рівня, LevelArr – масив який описуе знаходження труб та 3 readonly bool флага які відповідають за налаштування рівня).

Є абстрактный клас Sprite від якого унаслідуються клас шапки(Hat) та персонажа(Character).

Клас Hat має int поле height яке выдповыдає за висоту шапки, це значення використовется для правельного розміщення шапки на голові персонажа.

Клас Character має поле headLocation яке є обектом класа Point, це потрібне для правельного відображення шапки на голові персонажа.

* 1. **Розробка програмного забезпечення**

Основним методом програми, який перевіряє чи гравець пройшов рівень, є метод Check в класі GameField, він побудований на циклічній повірці можливості перейти до наступної труби. Спочатку метод шукає стартову трубу від якої можливо почати рухатися далі метод відштовхуючись від напрямку старту перевіряє можливість перейти далі. Якщо такої можливості немає то метот пропустить усі перевірити та поверне false, що буде означати що труба не зійшлася в кінцем. У разі якщо далі перейти можливо, програма переходить далі через оператор continue, і так поки не дійде до кінця або не зможе перейти далі.

Також є метод Update який приймає на вхід об’єкт класа Level у якого він зчитує масив рівня та параметри(наприклад наявність пустих клітинок чи заблокованих труб) та на основі цих параметрів генерує рівень замінюючи клітинки с 0id на випадкову за параметрами рівня.

Метод RollTiles який викликається в методі Update та випадково повертає усі труби, окрім заблокованих, щоб рівень кожен раз мав різний вигляд при цьому не змінюючи рішення рівня.

**Висновки до другого розділу:**

На етапі проектування програми мені довелося визначитися з структурою програми та обдумати усі проблеми з якими я міг би зіштовхнутися під час розробки. Потрібно було зробити структуру достатньо гнучкою, щоб можливо було без великих зусиль модифікувати програму, що дозволило мені після завершення реалізації основного фунціонала додавати нові можливості до гри(такі як персонаж та розблокування шапок для нього).

**РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ**

* 1. **Опис роботи з програмним додатком (Опис інтерфейсу)**

Після запуску гри ми побачимо меню (рис. 3.1) з 4 кнопками.

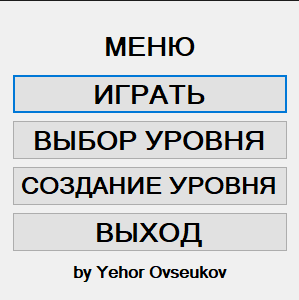


Рис. 3.1. Головне меню програми

Натиснувши на кнопку «Играть» з’являється повідомлення з інформацією (рис. 3.2).

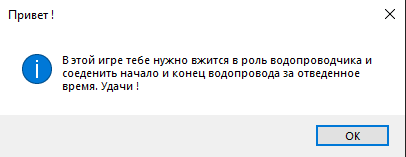


Рис. 3.2. повідомлення

Після натискання на кнопку «Ок» ми попадаємо на ігрове поле (рис 3.3) В центрі екрана ми бачимо саме ігрове поле, з правого – нашого персонажа. Зверху ми можемо побачити поточний рівень та кількість часу що у нас залишилося.

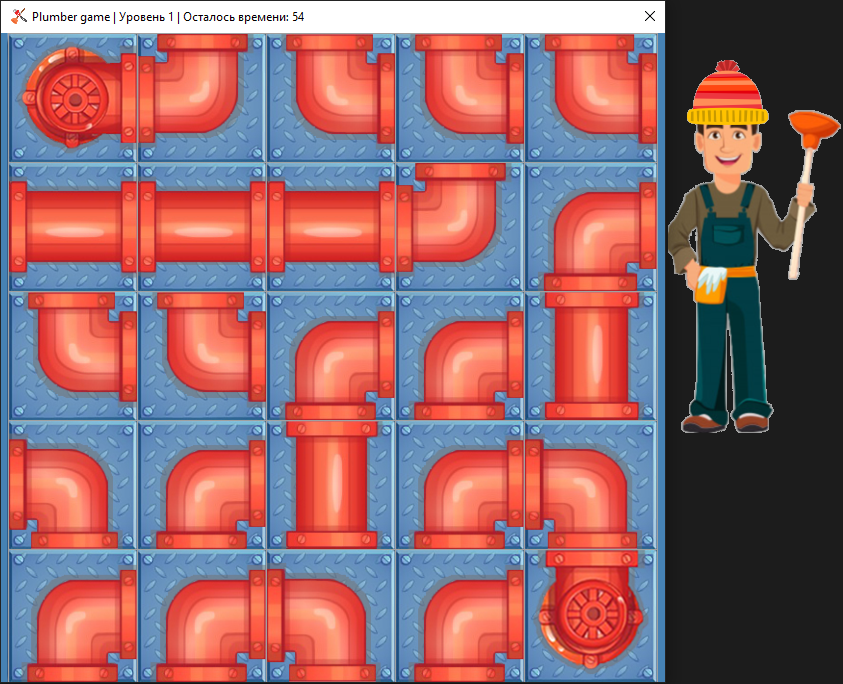


Рис. 3.3. ігрове поле

Якщо ми пройшли рівень вчасно ми побачимо форму с вітанням та 3 кнопками та інформацією про кількість часу що лишилося та кількість задіяних труб (рис. 3.4). У разі якщо час було вичерпано ми побачимо повідомлення (рис. 3.5) з можливістю почати рівень знову або вийти з гри.

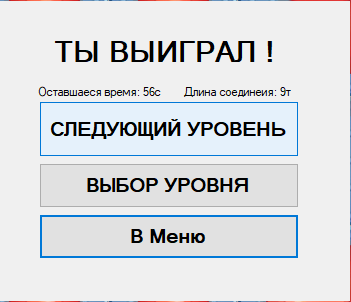


Рис. 3.4. форма перемоги

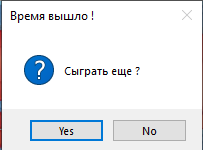


Рис. 3.5. повідомлення при вичерпанні часу

Натиснувши на кнопку «Вибор уровня» ми попадаємо на форму вибора рівня(рис. 3.6) де ми зможемо вибрати якийсь із розблокованих рівнів або із випадаючего меню вибрати наш кастомний рівень. Також тут є кнопка «рандомний» яка при натисканні відкриє нам випадковий рівень із розблокованих. Кнопка «сгенерировать» в свою чергу згенерую зовсім випадковий рівень. Кнопка «в меню» поверне нас до меню.

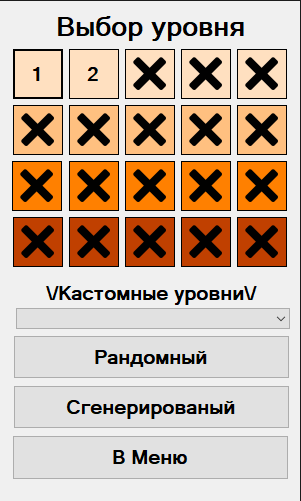


Рис. 3.6. вибір рівня

Натиснувши кнопку «создание уровня» в меню ми попадаємо на форму створення рівня(рис. 3.7) На цій формі ми можемо побачити ігрове поле зліва на якому ми можемо розміщати труби. Труби ми можемо обирати з допомогою колеса миші або натискаючи на кнопки справа зверху, там же ми можемо бачити обрану трубу а також попередню та наступну. Під вибором труби ми можемо побачити настройки рівня. Тут ми можемо дати назву нашому рівню та вибрати деякі параметри. Нижче ми бачимо інструкцію щодо користування конструктором рівня. Справа знизу ми бачимо 4 кнопки: «Сохранить уровень» яка валідує рівень та зберігає його, «заполнить пустие рандомно» яка заповнить усі пусті клітинки випадковими трубами, «очистить поле» яка очистить поле та «В меню» яка, як нам вже знайомо, поверне нас в меню. Як і на формі з ігровим полем справа ми бачимо нашого персонажа.

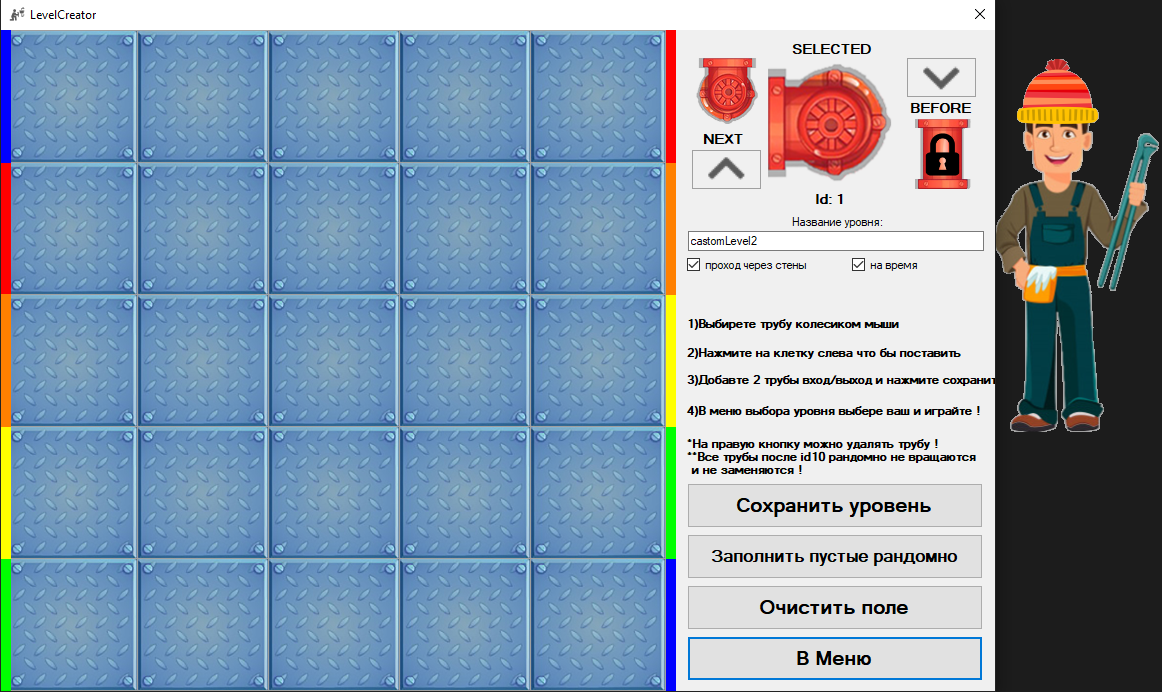


Рис. 3.7. форма створення рівня

* 1. **Тестування роботи програмного забезпечення**

Програма передбачае помилки пов’язані з створюванням рівнів. Наприкад при зберіганні рівня без входів та виходів программа каже що треба щоб їх було 2 (рис. 3.1)

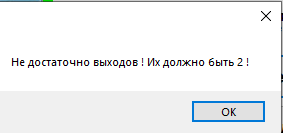


Рис. 3.1. помилка при недостатній кількости входів/виходів

З другого боку, якщо виходів/входів буду надто багато програма знов скаже що требя щоб їх було 2 (рис. 3.2)

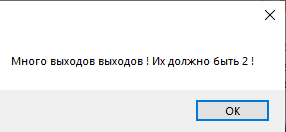


Рис. 3.2. помилка при занадто великій кількості входів/виходів

Також якщо створювати рівень з назвою яке вже існує ми також побачимо помилку (рис. 3.3)

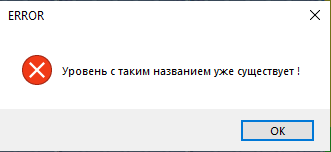


Рис. 3.3. помилка про існування рівня з такою назвою

**Висновки до третього розділу**

Під час розробки інтерфейсу гри мне довелося уявляти як буде бачити мою гру друга людина, не знайома з структурою програми. Треба було зробити максимально очевидні елементи управління та уникнути усіх можливих помилок повідомляючи про них користувачу. Також, потрібно було написати інструкцію до редактора рівнів щоб зменшити поріг входження та дати можливість користувачу відразу створювати власні рівні.

ВИСНОВКИ

Отже, під час виконання даної курсової роботи я закріпив матеріал, який вичався протягом 2 семестру, а саме мова програмуваня С# та проектування додатків використоввуючі матодологію ООП.

Я попрактикувася у аналізі власної роботи основуючись на вже існуючіх аналогів. Також навчився оцінювати переваги та недоліки інших проектів щоб мати уяву про те що можна покращити під час розробки власного продукту.

Отже, дана робота демонструє що з допомогою мови програмування С# та Windows Forms можливо побудувати ігри, особливо ігри в яких не є важливою велика кількість кадрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Эндрю Т. Язык Программирования C# 5.0 и платформа .NET 4.5 / Троелсен Эндрю., 2013. – 1168 с. – (6-е издание).
2. Об'єктно-орієнтоване програмування [Електронний ресурс] / Освітній портал ДУ «Житомирська політехніка» – 2020. – Режим доступу до ресурсу: https://learn.ztu.edu.ua/course/view.php?id=1628
3. Технології програмування. Мова С# : навч. посібн/ В.В. Томашевський. – Житомир: ЖВІ НАУ, 2012. – 484 с.
4. Павловская Т.А. Програмирование на языке высокого уровня / Т.А. Павловская – СПб.: Питер, 2011, – 461с.: ил.
5. Шилдт Герберт C# 4.0: полное руководство.: Пер. с англ. — М.: ООО "И.Д. Вильямс", 2011. — 1056 с.: ил.
6. Докумуентація до бібліотеки newtonsoft[Електронний ресурс] - <https://www.newtonsoft.com/json>
7. Робота з TableLayoutPanel [Електронний ресурс] – Режим доступу до ресурсу: <https://metanit.com/sharp/windowsforms/3.3.php>
8. Спосіб зберігання параметрів програми в Properties.Settings [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/271483/
9. Серіалізація в C# [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/>
10. Діалогові вікна в windows forms [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.form.showdialog?view=net-5.0

Додатки

Лістінг програми:

Класи:

class GameField

{

public static int amountOfConections = 0;

public static bool noClip = false;

public int CorrectLevel = 1;

public static List<Image> TilesIcons = new List<Image>()

{

Resources.randomTile, //0

Resources.Start\_end\_left, //1

Resources.Start\_end\_up, //2

Resources.Start\_end\_right, //3

Resources.Start\_end\_down, //4

Resources.Knee\_left\_up, //5

Resources.Knee\_up\_right, //6

Resources.Knee\_right\_down, //7

Resources.Knee\_down\_left, //8

Resources.Tube\_horizontal, //9

Resources.Tube\_vertical, //10

Resources.emptyTile, //11

Resources.Knee\_left\_up\_disabled, //12

Resources.Knee\_up\_right\_disabled, //13

Resources.Knee\_right\_down\_disabled, //14

Resources.Knee\_down\_left\_disabled, //15

Resources.Tube\_horizontal\_disabled, //16

Resources.Tube\_vertical\_disabled //17

};

Random rnd = new Random();

TableLayoutPanel Field;

public GameField(TableLayoutPanel Field, int CorrectLevel)

{

this.Field = Field;

this.CorrectLevel = CorrectLevel;

}

public void Update(Level Level, int CorrectLevel)

{

PictureBox tile;

do

{

for (int i = 0; i < Field.Controls.Count; i++)

{

tile = (PictureBox)Field.Controls[Field.Controls.Count - 1 - i];

tile.Image = TilesIcons[Level.LevelArr[i]];

if (Level.LevelArr[i] == 0)

{

if (CorrectLevel == 0 || CorrectLevel > 15 || Level.IsCastom)

{

int rand = rnd.Next(0, 101);

if (rand >= 0 && rand <= 80)

tile.Image = TilesIcons[rnd.Next(5, 11)];

else if (rand >= 80 && rand < 90)

tile.Image = TilesIcons[rnd.Next(12, 18)];

else

tile.Image = TilesIcons[11];

}

else if (CorrectLevel > 10 && CorrectLevel <= 15)

{

int rand = rnd.Next(0, 101);

if (rand >= 0 && rand <= 80)

tile.Image = TilesIcons[rnd.Next(5, 11)];

else if (rand >= 80 && rand < 95)

tile.Image = TilesIcons[rnd.Next(12, 18)];

else

tile.Image = TilesIcons[11];

}

else if (CorrectLevel > 5 && CorrectLevel <= 10)

{

int rand = rnd.Next(0, 101);

if (rand <= 90)

tile.Image = TilesIcons[rnd.Next(5, 11)];

else

tile.Image = TilesIcons[11];

}

else if (CorrectLevel <= 5)

tile.Image = TilesIcons[rnd.Next(5, 11)];

}

}

} while (!Check(noClip) && CorrectLevel == 0 && !Level.IsCastom);

RollTiles();

}

private void RollTiles()

{

for (int i = 0; i < Field.Controls.Count; i++)

{

int tileI = Field.Controls.Count - 1 - i;

PictureBox tile = (PictureBox)Field.Controls[tileI];

if (tile.Image == TilesIcons[5] || tile.Image == TilesIcons[6] || tile.Image == TilesIcons[6]

|| tile.Image == TilesIcons[8])

tile.Image = TilesIcons[rnd.Next(5, 9)];

else if (tile.Image == TilesIcons[9] || tile.Image == TilesIcons[10])

tile.Image = TilesIcons[rnd.Next(9, 11)];

}

}

public static void RotateTile(ref PictureBox tile)

{

//knee

if (tile.Image == TilesIcons[5])

tile.Image = TilesIcons[6];

else if (tile.Image == TilesIcons[6])

tile.Image = TilesIcons[7];

else if (tile.Image == TilesIcons[7])

tile.Image = TilesIcons[8];

else if (tile.Image == TilesIcons[8])

tile.Image = TilesIcons[5];

//regular tube

if (tile.Image == TilesIcons[9])

tile.Image = TilesIcons[10];

else if (tile.Image == TilesIcons[10])

tile.Image = TilesIcons[9];

}

public bool Check(bool noClip = false)

{

int previousI = 0;

bool startIsFound = false, firstTileIsStart = true;

amountOfConections = 0;

for (int i = 0; i < Field.Controls.Count; i++)

{

int tileI = Field.Controls.Count - 1 - i;

PictureBox tile = (PictureBox)Field.Controls[tileI];

Image nextRightTile = TilesIcons[0], nextDownTile = TilesIcons[0], nextUpTile = TilesIcons[0], nextLeftTile = TilesIcons[0], lastTile = TilesIcons[0],

firstTile = TilesIcons[0];

if (i != Field.Controls.Count - 1)

nextRightTile = ((PictureBox)Field.Controls[tileI - 1]).Image;

if (i != 0)

nextLeftTile = ((PictureBox)Field.Controls[tileI + 1]).Image;

if (i < Field.Controls.Count - 5)

nextDownTile = ((PictureBox)Field.Controls[tileI - 5]).Image;

if (i >= 5)

nextUpTile = ((PictureBox)Field.Controls[tileI + 5]).Image;

lastTile = ((PictureBox)Field.Controls[0]).Image;

firstTile = ((PictureBox)Field.Controls[Field.Controls.Count - 1]).Image;

int lastTileId = Field.Controls.Count - 1;

int firstTileId = 0;

bool canJumpLast;

bool canJumpFirst;

bool canJumpLeft;

bool canJumpRight;

bool canJumpUp = (nextUpTile == TilesIcons[7] || nextUpTile == TilesIcons[8] || nextUpTile == TilesIcons[10] ||

nextUpTile == TilesIcons[14] || nextUpTile == TilesIcons[15] || nextUpTile == TilesIcons[17]);

bool canJumpDown = (nextDownTile == TilesIcons[6] || nextDownTile == TilesIcons[5] || nextDownTile == TilesIcons[10] ||

nextDownTile == TilesIcons[13] || nextDownTile == TilesIcons[12] || nextDownTile == TilesIcons[17]);

if (!noClip)

{

canJumpLeft = (i % 5 != 0) && (nextLeftTile == TilesIcons[9] || nextLeftTile == TilesIcons[7] || nextLeftTile == TilesIcons[6] ||

nextLeftTile == TilesIcons[16] || nextLeftTile == TilesIcons[14] || nextLeftTile == TilesIcons[13]);

canJumpRight = (i + 1) % 5 != 0 && (nextRightTile == TilesIcons[5] || nextRightTile == TilesIcons[8] || nextRightTile == TilesIcons[9] ||

nextRightTile == TilesIcons[12] || nextRightTile == TilesIcons[15] || nextRightTile == TilesIcons[16]);

canJumpLast = false;

canJumpFirst = false;

}

else

{

canJumpLeft = (nextLeftTile == TilesIcons[9] || nextLeftTile == TilesIcons[7] || nextLeftTile == TilesIcons[6] ||

nextLeftTile == TilesIcons[16] || nextLeftTile == TilesIcons[14] || nextLeftTile == TilesIcons[13]);

canJumpRight = (nextRightTile == TilesIcons[5] || nextRightTile == TilesIcons[8] || nextRightTile == TilesIcons[9] ||

nextRightTile == TilesIcons[12] || nextRightTile == TilesIcons[15] || nextRightTile == TilesIcons[16]);

canJumpLast = (lastTile == TilesIcons[6] || lastTile == TilesIcons[7] || lastTile == TilesIcons[9] ||

lastTile == TilesIcons[13] || lastTile == TilesIcons[14] || lastTile == TilesIcons[16]);

canJumpFirst = (firstTile == TilesIcons[5] || firstTile == TilesIcons[8] || firstTile == TilesIcons[9] ||

firstTile == TilesIcons[12] || firstTile == TilesIcons[15] || firstTile == TilesIcons[16]);

}

//find start

if (!startIsFound && tile.Image != TilesIcons[1] && tile.Image != TilesIcons[2] && tile.Image != TilesIcons[3] && tile.Image != TilesIcons[4])

{

previousI = i;

firstTileIsStart = false;

continue;

}

startIsFound = true;

amountOfConections++;

if (noClip)

{

//first to last jump and last to first

//1

if (tile.Image == TilesIcons[1] && i == firstTileId && canJumpLast)

{

previousI = i;

i = lastTileId - 1;

continue;

}

//3

if (tile.Image == TilesIcons[3] && i == lastTileId && canJumpFirst)

{

previousI = i;

i = firstTileId - 1;

continue;

}

//6

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && i == lastTileId && previousI == firstTileId && canJumpUp)

{

previousI = i;

i -= 6;

continue;

}

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && i == lastTileId && previousI == i - 5 && canJumpFirst)

{

previousI = i;

i = firstTileId - 1;

continue;

}

//8

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && i == firstTileId && previousI == lastTileId && canJumpDown)

{

previousI = i;

i += 4;

continue;

}

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && i == firstTileId && previousI == i + 5 && canJumpLast)

{

previousI = i;

i = lastTileId - 1;

continue;

}

//9

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == firstTileId && previousI == lastTileId && canJumpRight)

{

previousI = i;

continue;

}

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == firstTileId && previousI == i + 1 && canJumpLast)

{

previousI = i;

i = lastTileId - 1;

continue;

}

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == lastTileId && previousI == firstTileId && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == lastTileId && previousI == i - 1 && canJumpFirst)

{

previousI = i;

i = firstTileId - 1;

continue;

}

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == lastTileId && previousI == firstTileId && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

}

//start logic

//1

if (tile.Image == TilesIcons[1] && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

//2

if (tile.Image == TilesIcons[2] && canJumpUp)

{

previousI = i;

i -= 6;

continue;

}

//3

if (tile.Image == TilesIcons[3] && canJumpRight)

{

previousI = i;

continue;

}

//4

if (tile.Image == TilesIcons[4] && canJumpDown)

{

previousI = i;

i += 4;

continue;

}

//tubes logic

//5

if ((tile.Image == TilesIcons[5] || tile.Image == TilesIcons[12]) && previousI == i - 1 && canJumpUp)

{

previousI = i;

i -= 6;

continue;

}

if ((tile.Image == TilesIcons[5] || tile.Image == TilesIcons[12]) && previousI == i - 5 && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

//6

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && previousI == i - 5 && canJumpRight)

{

previousI = i;

continue;

}

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && previousI == i + 1 && canJumpUp)

{

previousI = i;

i -= 6;

continue;

}

//7

if ((tile.Image == TilesIcons[7] || tile.Image == TilesIcons[14]) && previousI == i + 5 && canJumpRight)

{

previousI = i;

continue;

}

if ((tile.Image == TilesIcons[7] || tile.Image == TilesIcons[14]) && previousI == i + 1 && canJumpDown)

{

previousI = i;

i += 4;

continue;

}

//8

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && previousI == i - 1 && canJumpDown)

{

previousI = i;

i += 4;

continue;

}

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && previousI == i + 5 && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

//9

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && previousI == i - 1 && canJumpRight)

{

previousI = i;

continue;

}

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && previousI == i + 1 && canJumpLeft)

{

previousI = i;

i -= 2;

continue;

}

//10

if (previousI == i - 5 && (tile.Image == TilesIcons[10] || tile.Image == TilesIcons[17]) && canJumpDown)

{

previousI = i;

i += 4;

continue;

}

if (previousI == i + 5 && (tile.Image == TilesIcons[10] || tile.Image == TilesIcons[17]) && canJumpUp)

{

previousI = i;

i -= 6;

continue;

}

//when 1 is end

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && nextRightTile == TilesIcons[1] && previousI == i - 1)

break;

if ((tile.Image == TilesIcons[7] || tile.Image == TilesIcons[14]) && nextRightTile == TilesIcons[1] && previousI == i + 5)

break;

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && nextRightTile == TilesIcons[1] && previousI == i - 5)

break;

if (noClip && !firstTileIsStart)

{

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && nextRightTi

le == TilesIcons[1] && i == firstTileId && previousI == lastTileId)

break;

}

//when 2 is end

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && nextDownTile == TilesIcons[2] && previousI == i - 1)

break;

if ((tile.Image == TilesIcons[10] || tile.Image == TilesIcons[17]) && nextDownTile == TilesIcons[2] && previousI == i - 5)

break;

if ((tile.Image == TilesIcons[7] || tile.Image == TilesIcons[14]) && nextDownTile == TilesIcons[2] && previousI == i + 1)

break;

//when 3 is end

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && nextLeftTile == TilesIcons[3] && previousI == i + 1)

break;

if ((tile.Image == TilesIcons[5] || tile.Image == TilesIcons[12]) && nextLeftTile == TilesIcons[3] && previousI == i - 5)

break;

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && nextLeftTile == TilesIcons[3] && previousI == i + 5)

break;

if (noClip && !firstTileIsStart)

{

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && nextLeftTile == TilesIcons[3] && previousI == firstTileId && i == firstTileId)

break;

if ((tile.Image == TilesIcons[8] || tile.Image == TilesIcons[15]) && nextLeftTile == TilesIcons[3] && previousI == firstTileId + 5 && i == firstTileId)

break;

if ((tile.Image == TilesIcons[9] || tile.Image == TilesIcons[16]) && i == firstTileId && lastTile == TilesIcons[3] && previousI == i + 1)

break;

}

//when 4 is end

if ((tile.Image == TilesIcons[5] || tile.Image == TilesIcons[12]) && nextUpTile == TilesIcons[4] && previousI == i - 1)

break;

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && nextUpTile == TilesIcons[4] && previousI == i + 1)

break;

if ((tile.Image == TilesIcons[10] || tile.Image == TilesIcons[17]) && nextUpTile == TilesIcons[4] && previousI == i + 5)

break;

if (noClip && !firstTileIsStart)

{

if ((tile.Image == TilesIcons[5] || tile.Image == TilesIcons[12]) && nextUpTile == TilesIcons[4] && i == lastTileId && previousI == lastTileId - 1)

break;

if ((tile.Image == TilesIcons[6] || tile.Image == TilesIcons[13]) && nextUpTile == TilesIcons[4] && i == lastTileId && previousI == firstTileId)

break;

if ((tile.Image == TilesIcons[10] || tile.Image == TilesIcons[17]) && nextUpTile == TilesIcons[4] && i == lastTileId && previousI == firstTileId + 5)

break;

}

return false;

}

amountOfConections--;

return true;

}

}

public abstract class Sprite

{

Image image;

Size size;

public Image Image

{

get

{

return image;

}

set

{

image = value;

}

}

public Size Size

{

get

{

return size;

}

set

{

size = value;

}

}

public Sprite(Image Image)

{

this.image = Image;

this.size = Image.Size;

}

public Sprite(Image Image, Size Size)

{

this.image = Image;

this.size = Size;

}

}

public class Character : Sprite

{

Point headLocation;

public Point HeadLocation

{

get

{

return headLocation;

}

set

{

headLocation = value;

}

}

public Character(Image Image, Point HeadLocation)

:base(Image)

{

this.headLocation = HeadLocation;

}

}

public class Hat : Sprite

{

int hatHight;

public int HatHight

{

get

{

return hatHight;

}

set

{

hatHight = value;

}

}

public Hat(Image HatImage, int HatHight)

: base(HatImage)

{

this.hatHight = HatHight;

}

}

public class Level

{

[JsonProperty("name")]

public string Name;

[JsonProperty("levelArr")]

public int[] LevelArr;

[JsonProperty("isCastom")]

public readonly bool IsCastom;

[JsonProperty("isNoClip")]

public readonly bool isNoClip;

[JsonProperty("isOnTime")]

public readonly bool isOnTime;

public Level(string Name, int[] LevelArr, bool isCastom = false, bool isNoClip = false, bool isOnTime = true)

{

this.Name = Name;

this.LevelArr = LevelArr;

this.IsCastom = isCastom;

this.isNoClip = isNoClip;

this.isOnTime = isOnTime;

}

}

class Levels

{

public static List<Level> GameLevelsList = new List<Level>()

{

new Level("generated" ,new int[] { 3, 0, 0, 0, 0,

0, 0, 0, 0, 0,

0, 0, 0, 0, 0,

0, 0, 0, 0, 0,

0, 0, 0, 0, 1 },false, true),

new Level("1" ,new int[] { 3, 8, 0, 0, 0,

0, 10, 0, 0, 0,

0, 6, 8, 0, 0,

0, 0, 10, 7, 8,

0, 0, 6, 5, 2 }),

new Level("2" ,new int[] { 3, 9, 9, 8, 0,

0, 7, 9, 5, 0,

0, 6, 9, 8, 0,

6, 0, 7, 5, 0,

0, 0, 6, 9, 1 }),

new Level("3" ,new int[] { 7, 8, 0, 0, 0,

2, 6, 8, 0, 0,

0, 0, 6, 8, 0,

6, 0, 0, 6, 8,

0, 0, 0, 0, 2 }),

new Level("4" ,new int[] { 0, 0, 0, 7, 1,

0, 0, 0, 6, 8,

0, 7, 8, 7, 5,

7, 5, 6, 5, 0,

2, 0, 0, 0, 0 }),

new Level("5" ,new int[] { 3, 9, 5, 0, 0,

0, 0, 5, 9, 5,

5, 9, 9, 9, 5,

9, 4, 0, 0, 0,

5, 5, 0, 0, 0 }),

new Level("6" ,new int[] { 3, 5, 11, 5, 1,

5, 5, 11, 9, 0,

5, 9, 9, 5, 0,

0, 0, 11, 0, 0,

0, 0, 11, 0, 0 }),

new Level("7" ,new int[] { 4, 5, 9, 5, 0,

5, 5, 11, 9, 11,

11, 5, 0, 9, 0,

0, 9, 5, 5, 0,

0, 3, 5, 0, 0 }),

new Level("8" ,new int[] { 5, 9, 9, 9, 1,

5, 5, 5, 11, 14,

11, 9, 11, 11, 17,

5, 5, 5, 11, 13,

5, 9, 9, 9, 1 }),

new Level("9" ,new int[] { 5, 5, 5, 5, 4,

9, 5, 5, 5, 5,

9, 11, 11, 11, 11,

9, 5, 5, 5, 5,

5, 5, 5, 5, 2 }),

new Level("10" ,new int[] { 4, 0, 0, 0, 4,

9, 11, 0, 11, 9,

5, 9, 5, 11, 9,

0, 11, 9, 11, 9,

11, 0, 5, 9, 5 }),

new Level("11" ,new int[] { 14, 12, 0, 13, 15,

12, 0, 5, 5, 13,

3, 15, 9, 13, 1,

15, 5, 5, 0, 14,

13, 15, 0, 14, 12 }),

new Level("12" ,new int[] { 0, 4, 0, 4, 0,

5, 12, 16, 13, 5,

5, 15, 11, 14, 5,

5, 5, 0, 5, 5,

5, 9, 16, 9, 5 }),

new Level("13" ,new int[] { 14, 9, 5, 0, 15,

9, 0, 2, 0, 0,

13, 9, 16, 9, 15,

0, 0, 4, 0, 9,

13, 0, 5, 9, 12 }),

new Level("14" ,new int[] { 4, 0, 16, 16, 5,

5, 5, 16, 5, 5,

0, 17, 11, 17, 9,

5, 12, 16, 9, 9,

5, 9, 9, 5, 2 }),

new Level("15" ,new int[] { 0, 0, 0, 0, 4,

5, 5, 5, 5, 9,

9, 17, 17, 17, 9,

9, 5, 5, 9, 9,

2, 0, 0, 13, 5 }),

new Level("16" ,new int[] { 4, 0, 11, 0, 5,

12, 0, 11, 0, 5,

9, 5, 11, 0, 0,

0, 9, 11, 5, 16,

9, 5, 11, 5, 1 },false, true),

new Level("17" ,new int[] { 1, 5, 5, 0, 3,

9, 5, 9, 0, 0,

0, 0, 9, 0, 0,

5, 5, 9, 5, 9,

12, 5, 5, 5, 16 },false, true),

new Level("18" ,new int[] { 3, 5, 0, 13, 5,

9, 5, 5, 9, 5,

16, 16, 5, 16, 16,

5, 5, 9, 5, 9,

5, 5, 5, 5, 1 },false, true),

new Level("19" ,new int[] { 9, 5, 11, 5, 9,

9, 5, 11, 5, 1,

11, 11, 11, 11, 11,

3, 5, 11, 5, 9,

9, 5, 11, 5, 9},false, true),

new Level("20" ,new int[] { 1, 3, 9, 15, 7,

9, 16, 9, 7, 9,

5, 9, 9, 5, 13,

5, 5, 5, 9, 14,

9, 12, 5, 12, 6 },false, true)

};

public static List<Level> CastomLevelsList = new List<Level>();

public static List<Level> CorrectlevelList = GameLevelsList;

public static int LevelAmount = Levels.GameLevelsList.Count - 1;

public static int CorrectLevelId = 1;

public static int AvailableLevel = Properties.Settings.Default.avalibleLevel;

static Levels()

{

if (!File.Exists("levels.json"))

File.Create("levels.json").Close();

else

CastomLevelsList = GetCustomLevels();

}

public static void ChangeCorrectLevelList(bool isCustomMode)

{

if(isCustomMode)

CorrectlevelList = CastomLevelsList;

else

CorrectlevelList = GameLevelsList;

}

public static void AddLevel(Level level)

{

CastomLevelsList.Add(level);

File.WriteAllText("levels.json", JsonConvert.SerializeObject(CastomLevelsList));

}

public static List<Level> GetCustomLevels()

{

try

{

List<Level> levels = JsonConvert.DeserializeObject<List<Level>>(File.ReadAllText("levels.json"));

return levels != null ? levels : new List<Level>();

}

catch

{

DialogResult userAns = MessageBox.Show("Ошибка при считовании файла ! Очистить содержимое ?", "Ошибка", MessageBoxButtons.YesNo, MessageBoxIcon.Error);

if (userAns == DialogResult.Yes)

{

File.Delete("levels.json");

File.Create("levels.json").Close();

}

return new List<Level>();

}

}

}

Форми:

public partial class CharacterDisplay : Form

{

static List<Hat> hats = new List<Hat> { null, new Hat(Resources.hat\_1, 32), new Hat(Resources.hat\_2, 30), new Hat(Resources.hat\_3, 5), new Hat(Resources.hat\_4, 20),

new Hat(Resources.hat\_5, 20), new Hat(Resources.hat\_6, 42)};

static List<Character> characterPos = new List<Character> { null, new Character(Resources.char\_pos1, new Point(-11, 0)), new Character(Resources.char\_pos2, new Point(23, 0)),

new Character(Resources.char\_pos3, new Point(-9, 0))};

Point HatLocation;

public static int AvalibleHats = Properties.Settings.Default.avalibleHats;

int \_selectedHat = Properties.Settings.Default.selectedHat;

int \_charPosotion = 1;

public CharacterDisplay()

{

InitializeComponent();

HatLocation = pictureBoxHat.Location;

}

public static Hat UnlockHat()

{

if (AvalibleHats != hats.Count - 1)

{

Properties.Settings.Default.avalibleHats = AvalibleHats++;

Properties.Settings.Default.Save();

}

return hats[AvalibleHats];

}

public void ChangeCharPosition(int charID)

{

if (charID <= 0 || charID > 3)

{

this.BackgroundImage = characterPos[1].Image;

\_charPosotion = 1;

}

else

{

\_charPosotion = charID;

HatLocation = characterPos[charID].HeadLocation;

HatLocation.Y -= hats[\_selectedHat].HatHight;

pictureBoxHat.Location = HatLocation;

this.BackgroundImage = characterPos[charID].Image;

}

}

public void ChangeCharsHat()

{

HatLocation = characterPos[\_charPosotion].HeadLocation;

HatLocation.Y -= hats[\_selectedHat].HatHight;

pictureBoxHat.Location = HatLocation;

pictureBoxHat.BackgroundImage = hats[\_selectedHat].Image;

Properties.Settings.Default.selectedHat = \_selectedHat;

Properties.Settings.Default.Save();

}

private void Character\_Load(object sender, EventArgs e)

{

this.BackColor = Color.DarkGray;

this.TransparencyKey = Color.DarkGray;

ChangeCharsHat();

}

public void Attach(Point Formlocation, int FormWidth)

{

Point gameLocation = Formlocation;

gameLocation.X += FormWidth - 7;

gameLocation.Y += 30;

this.Location = gameLocation;

}

private void Character\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void Character\_Click(object sender, EventArgs e)

{

if (\_selectedHat >= AvalibleHats)

{

\_selectedHat = 1;

ChangeCharsHat();

}

else

{

\_selectedHat++;

ChangeCharsHat();

}

}

}

public partial class LevelCreator : Form

{

static List<Image> TilesIcons = GameField.TilesIcons;

private int selectedTileId = 1;

CharacterDisplay character = new CharacterDisplay();

public LevelCreator()

{

InitializeComponent();

}

private void LevelCreator\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void buttonMenu\_Click(object sender, EventArgs e)

{

character.Hide();

new Menu().Show();

this.Hide();

}

private void FandeMouseWeel(object sender, MouseEventArgs e)

{

selectedTileId += e.Delta / Math.Abs(e.Delta);

menuSwaper();

}

private void buttonUp\_Click(object sender, EventArgs e)

{

selectedTileId += 1;

menuSwaper();

}

private void buttonDown\_Click(object sender, EventArgs e)

{

selectedTileId -= 1;

menuSwaper();

}

private void buttonClearPlayground\_Click(object sender, EventArgs e)

{

FillWithEmpty();

}

private void buttonSaveLevel\_Click(object sender, EventArgs e)

{

ValidateLevel();

}

private void pictureBoxTile\_Click(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

((PictureBox)sender).Image = TilesIcons[selectedTileId];

else if (e.Button == MouseButtons.Right)

((PictureBox)sender).Image = TilesIcons[11];

}

private void buttonFillRandom\_Click(object sender, EventArgs e)

{

FillEmptyWithRandom();

}

private void menuSwaper()

{

if (selectedTileId < 0)

selectedTileId = 17;

else if (selectedTileId > 17)

selectedTileId = 0;

pictureBoxSelectetTile.BackgroundImage = TilesIcons[selectedTileId];

if (selectedTileId == 0)

labelTileId.Text = $"RND";

else

labelTileId.Text = $"Id: {selectedTileId}";

if (selectedTileId == 17)

{

pictureBoxNextTile.BackgroundImage = TilesIcons[1];

pictureBoxBeforeTile.BackgroundImage = TilesIcons[selectedTileId - 1];

}

else if (selectedTileId == 0)

{

pictureBoxNextTile.BackgroundImage = TilesIcons[selectedTileId + 1];

pictureBoxBeforeTile.BackgroundImage = TilesIcons[17];

}

else

{

pictureBoxNextTile.BackgroundImage = TilesIcons[selectedTileId + 1];

pictureBoxBeforeTile.BackgroundImage = TilesIcons[selectedTileId - 1];

}

}

private void FillWithEmpty()

{

for (int i = 0; i < tableLayoutPanelPlayground.Controls.Count; i++)

{

PictureBox tile = (PictureBox)tableLayoutPanelPlayground.Controls[tableLayoutPanelPlayground.Controls.Count - 1 - i];

tile.Image = TilesIcons[11];

}

}

private void FillEmptyWithRandom()

{

for (int i = 0; i < tableLayoutPanelPlayground.Controls.Count; i++)

{

PictureBox tile = (PictureBox)tableLayoutPanelPlayground.Controls[tableLayoutPanelPlayground.Controls.Count - 1 - i];

if (tile.Image == TilesIcons[11])

tile.Image = TilesIcons[0];

}

}

private void ValidateLevel()

{

int exitsAmount = 0;

int[] newLevel = new int[tableLayoutPanelPlayground.Controls.Count];

for (int i = 0; i < tableLayoutPanelPlayground.Controls.Count; i++)

{

PictureBox tile = (PictureBox)tableLayoutPanelPlayground.Controls[tableLayoutPanelPlayground.Controls.Count - 1 - i];

for (byte j = 0; j < TilesIcons.Count; j++)

{

if (tile.Image == TilesIcons[j])

newLevel[i] = j;

}

if (tile.Image == TilesIcons[1] || tile.Image == TilesIcons[2] || tile.Image == TilesIcons[3] || tile.Image == TilesIcons[4])

exitsAmount++;

}

if (exitsAmount < 2)

MessageBox.Show("Не достаточно выходов ! Их должно быть 2 !");

else if (exitsAmount > 2)

MessageBox.Show("Много выходов выходов ! Их должно быть 2 !");

else

{

try

{

foreach (Level oldLevel in Levels.CastomLevelsList)

{

if (oldLevel.Name == textBoxLevelName.Text)

throw new Exception("Уровень с таким названием уже существует !");

}

Levels.AddLevel(new Level(textBoxLevelName.Text, newLevel, true, checkBoxNoClip.Checked, checkBoxOnTime.Checked));

character.ChangeCharPosition(2);

MessageBox.Show($"Уровень был успешно сохранен !", "Успех !");

character.ChangeCharPosition(3);

textBoxLevelName.Text = "castomLevel" + Levels.CastomLevelsList.Count;

FillWithEmpty();

}

catch (Exception e)

{

MessageBox.Show(e.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void LevelCreator\_Load(object sender, EventArgs e)

{

textBoxLevelName.Text = "castomLevel" + Levels.CastomLevelsList.Count;

this.MouseWheel += FandeMouseWeel;

FillWithEmpty();

pictureBoxNextTile.BackgroundImage = TilesIcons[selectedTileId + 1];

pictureBoxSelectetTile.BackgroundImage = TilesIcons[selectedTileId];

pictureBoxBeforeTile.BackgroundImage = TilesIcons[17];

character.Show();

character.Attach(this.Location, this.Width);

character.ChangeCharPosition(3);

}

private void LevelCreator\_LocationChanged(object sender, EventArgs e)

{

character.Attach(this.Location, this.Width);

}

private void checkBoxNoClip\_CheckedChanged(object sender, EventArgs e)

{

if (checkBoxNoClip.Checked)

{

pictureBoxUi1.Visible = true;

pictureBoxUi2.Visible = true;

pictureBoxUi3.Visible = true;

pictureBoxUi4.Visible = true;

pictureBoxUi5.Visible = true;

pictureBoxUi6.Visible = true;

pictureBoxUi7.Visible = true;

pictureBoxUi8.Visible = true;

pictureBoxUi9.Visible = true;

pictureBoxUi10.Visible = true;

}

else

{

pictureBoxUi1.Visible = false;

pictureBoxUi2.Visible = false;

pictureBoxUi3.Visible = false;

pictureBoxUi4.Visible = false;

pictureBoxUi5.Visible = false;

pictureBoxUi6.Visible = false;

pictureBoxUi7.Visible = false;

pictureBoxUi8.Visible = false;

pictureBoxUi9.Visible = false;

pictureBoxUi10.Visible = false;

}

}

}

public partial class LevelSelect : Form

{

Random rnd = new Random();

public LevelSelect()

{

InitializeComponent();

}

private void LevelSelect\_Load(object sender, EventArgs e)

{

Button[] Buttons = { null, buttonLevel1, buttonLevel2, buttonLevel3, buttonLevel4, buttonLevel5, buttonLevel6, buttonLevel7, buttonLevel8, buttonLevel9, buttonLevel10,

buttonLevel11, buttonLevel12, buttonLevel13, buttonLevel14, buttonLevel15, buttonLevel16, buttonLevel17, buttonLevel18, buttonLevel19, buttonLevel20};

for(int i = 1; i < Buttons.Length; i++)

{

if(i <= Levels.AvailableLevel)

{

Buttons[i].Enabled = true;

Buttons[i].Text = i.ToString();

Buttons[i].BackgroundImage = null;

}

}

foreach(Level level in Levels.GetCustomLevels())

{

comboBoxCustomLevels.Items.Add(level.Name);

}

}

private void LevelSelect\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void buttonLevelRandom\_Click(object sender, EventArgs e)

{

Levels.CorrectLevelId = 0;

Levels.ChangeCorrectLevelList(false);

new PlumberGame().Show();

this.Hide();

}

private void buttonLevel\_Click(object sender, EventArgs e)

{

Levels.CorrectLevelId = int.Parse(((Button)sender).Text);

Levels.ChangeCorrectLevelList(false);

new PlumberGame().Show();

this.Hide();

}

private void buttonMenu\_Click(object sender, EventArgs e)

{

new Menu().Show();

this.Hide();

}

private void buttonLevelRandom\_Click\_1(object sender, EventArgs e)

{

Levels.CorrectLevelId = rnd.Next(1, Levels.AvailableLevel);

Levels.ChangeCorrectLevelList(false);

new PlumberGame().Show();

this.Hide();

}

private void comboBoxCustomLevels\_SelectedIndexChanged(object sender, EventArgs e)

{

Levels.CorrectLevelId = comboBoxCustomLevels.SelectedIndex;

Levels.ChangeCorrectLevelList(true);

new PlumberGame().Show();

this.Hide();

}

}

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

new PlumberGame().Show();

this.Hide();

}

private void buttonLevelSelect\_Click(object sender, EventArgs e)

{

new LevelSelect().Show();

this.Hide();

}

private void buttonLevelCreation\_Click(object sender, EventArgs e)

{

new LevelCreator().Show();

this.Hide();

}

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Menu\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void Menu\_Load(object sender, EventArgs e)

{

if (Properties.Settings.Default.avalibleLevel > 1)

buttonClearProgress.Visible = true;

else

buttonClearProgress.Visible = false;

}

private void buttonClearProgress\_Click(object sender, EventArgs e)

{

Properties.Settings.Default.avalibleHats = 1;

Properties.Settings.Default.avalibleLevel = 1;

Properties.Settings.Default.selectedHat = 1;

Properties.Settings.Default.Save();

Application.Restart();

buttonClearProgress.Visible = false;

}

private void buttonExit\_KeyPress(object sender, KeyPressEventArgs e)

{

if(e.KeyChar == (char)Keys.P)

if (DialogResult.Yes == MessageBox.Show("Вы хотите заблокировать весь контент ?", "ЧИТЫ !", MessageBoxButtons.YesNo, MessageBoxIcon.Question))

{

Properties.Settings.Default.avalibleHats = 4;

Properties.Settings.Default.avalibleLevel = 18;

Properties.Settings.Default.selectedHat = 1;

Properties.Settings.Default.Save();

Application.Restart();

}

}

}

public partial class PlumberGame : Form

{

public static int time;

public static bool noClip = false;

public static int amountOfConections = 0;

CharacterDisplay character = new CharacterDisplay();

GameField gameField;

public PlumberGame()

{

InitializeComponent();

gameField = new GameField(tableLayoutPanel1, Levels.CorrectLevelId);

}

private void Form1\_Load(object sender, EventArgs e)

{

time = 60;

this.Text = $"Plumber game | Уровень {Levels.CorrectlevelList[Levels.CorrectLevelId].Name}";

noClip = Levels.CorrectlevelList[Levels.CorrectLevelId].isNoClip;

if (!noClip)

{

pictureBoxUi1.Visible = false;

pictureBoxUi2.Visible = false;

pictureBoxUi3.Visible = false;

pictureBoxUi4.Visible = false;

pictureBoxUi5.Visible = false;

pictureBoxUi6.Visible = false;

pictureBoxUi7.Visible = false;

pictureBoxUi8.Visible = false;

pictureBoxUi9.Visible = false;

pictureBoxUi10.Visible = false;

}

else

{

pictureBoxUi1.Visible = true;

pictureBoxUi2.Visible = true;

pictureBoxUi3.Visible = true;

pictureBoxUi4.Visible = true;

pictureBoxUi5.Visible = true;

pictureBoxUi6.Visible = true;

pictureBoxUi7.Visible = true;

pictureBoxUi8.Visible = true;

pictureBoxUi9.Visible = true;

pictureBoxUi10.Visible = true;

}

if (Levels.AvailableLevel == 1 && Levels.CorrectLevelId == 1 && !Levels.CorrectlevelList[Levels.CorrectLevelId].IsCastom)

MessageBox.Show("В этой игре тебе нужно вжится в роль водопроводчика и соеденить начало и конец водопровода за отведенное время. Удачи !", "Привет !",

MessageBoxButtons.OK, MessageBoxIcon.Information);

if (Levels.AvailableLevel == 6 && Levels.CorrectLevelId == 6 && !Levels.CorrectlevelList[Levels.CorrectLevelId].IsCastom)

MessageBox.Show("Начиная с этого уровня тебе будут иногда попадатся пустые клетки без труб. Удачи !", "Поздравляю !",

MessageBoxButtons.OK, MessageBoxIcon.Information);

if (Levels.AvailableLevel == 11 && Levels.CorrectLevelId == 11 && !Levels.CorrectlevelList[Levels.CorrectLevelId].IsCastom)

MessageBox.Show("Ты прошел ещё 1 ряд ! Теперь некоторые из труб вращать нельзя ! Удачи !", "Поздравляю !",

MessageBoxButtons.OK, MessageBoxIcon.Information);

if (Levels.AvailableLevel == 16 && Levels.CorrectLevelId == 16 && !Levels.CorrectlevelList[Levels.CorrectLevelId].IsCastom)

{

MessageBox.Show("Ты подошел к финишной прямой, это последний ряд уровней ! Теперь труба переходит в одинаковые цвета по краям!", "Поздравляю !",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

timer.Enabled = Levels.CorrectlevelList[Levels.CorrectLevelId].isOnTime;

this.Enabled = true;

gameField.Update(Levels.CorrectlevelList[Levels.CorrectLevelId], Levels.CorrectLevelId);

character.Show();

character.Attach(this.Location, this.Width);

character.ChangeCharPosition(1);

}

private void pictureBox7\_MouseClick(object sender, MouseEventArgs e)

{

PictureBox tile = (PictureBox)sender;

GameField.RotateTile(ref tile);

if (gameField.Check(noClip))

{

this.Enabled = false;

timer.Enabled = false;

character.ChangeCharPosition(2);

new WinScreen(Levels.CorrectlevelList[Levels.CorrectLevelId]).ShowDialog();

this.Hide();

character.Hide();

}

}

private void button1\_Click(object sender, EventArgs e)

{

Form1\_Load(sender, e);

}

private void timer\_Tick(object sender, EventArgs e)

{

this.Text = $"Plumber game | Уровень {Levels.CorrectlevelList[Levels.CorrectLevelId].Name} | Осталось времени: {--time}";

if (time == 0)

{

this.Enabled = false;

timer.Enabled = false;

var ans = MessageBox.Show("Сыграть еще ?", "Время вышло !", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (ans == DialogResult.Yes)

Form1\_Load(sender, e);

else

this.Close();

}

}

private void PlumberGame\_FormClosed(object sender, FormClosedEventArgs e)

{

timer.Enabled = false;

new Menu().Show();

this.Hide();

character.Hide();

}

private void PlumberGame\_LocationChanged(object sender, EventArgs e)

{

character.Attach(this.Location, this.Width);

}

}

public partial class UnlocksDisplay : Form

{

public UnlocksDisplay(Hat hat)

{

InitializeComponent();

pictureBox1.Image = hat.Image;

}

public UnlocksDisplay()

{

InitializeComponent();

}

private void UnlocksDisplay\_Load(object sender, EventArgs e)

{

}

}

public partial class WinScreen : Form

{

Level correctLevel;

UnlocksDisplay unlocksDisplay = new UnlocksDisplay();

public WinScreen(Level level)

{

InitializeComponent();

this.correctLevel = level;

}

private void WinScreen\_Load(object sender, EventArgs e)

{

if(correctLevel.isOnTime)

labelTimeLeft.Text = $"Оставшаеся время: {PlumberGame.time}c";

labelAmountOfConections.Text = $"Длина соединеия: {GameField.amountOfConections}т";

if (Levels.CorrectLevelId == Levels.AvailableLevel && !correctLevel.IsCastom)

{

Properties.Settings.Default.avalibleLevel = ++Levels.AvailableLevel;

Properties.Settings.Default.Save();

if (Levels.AvailableLevel % 5 == 0)

{

unlocksDisplay = new UnlocksDisplay(CharacterDisplay.UnlockHat());

unlocksDisplay.ShowDialog();

}

}

if (Levels.CorrectLevelId == Levels.LevelAmount && !correctLevel.IsCastom)

{

labelCompGame.Visible = true;

buttonNextLevel.Enabled = false;

}

else if (Levels.CorrectLevelId == 0 || Levels.CorrectLevelId > 20 && !correctLevel.IsCastom)

{

labelRandomly.Visible = true;

buttonNextLevel.Enabled = false;

}

else

labelWin.Visible = true;

if (Levels.CorrectLevelId == 0 || correctLevel.IsCastom)

buttonAgain.Visible = true;

if (Levels.CorrectLevelId != 0 && !correctLevel.IsCastom)

buttonNextLevel.Visible = true;

}

private void buttonMenu\_Click(object sender, EventArgs e)

{

new Menu().Show();

unlocksDisplay.Hide();

this.Hide();

}

private void WinScreen\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void buttonNextLevel\_Click(object sender, EventArgs e)

{

Levels.CorrectLevelId++;

new PlumberGame().Show();

unlocksDisplay.Hide();

this.Hide();

}

private void buttonLevelSelect\_Click(object sender, EventArgs e)

{

new LevelSelect().Show();

unlocksDisplay.Hide();

this.Hide();

}

private void buttonAgain\_MouseClick(object sender, MouseEventArgs e)

{

new PlumberGame().Show();

unlocksDisplay.Hide();

this.Hide();

}

}