НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматики та управляння в технічних системах

Реалізація програми «Конструктор рівней гри “Battle City”»

Курсова робота

З дисципліни «Об’єктно-орієнтоване програмування»

|  |  |
| --- | --- |
| Керівник  Хмелюк В.С.  «Допущений до захисту»    (Особистий підпис керівника)  « \_\_\_\_ » 2017 р.  Захищений з оцінкою    (оцінка)  Члени комісії:    (Особистий підпис)    (Особистий підпис) | Виконавець  ст. Макіян С.А.  зал. Книжка № ЗПІ-зп6115  гр. ЗПІ-ЗП 61    (Особистий підпис виконавця)  « \_\_\_\_ » 2017 р.    (Розшифровка підпису)    (Розшифровка підпису) |

Київ - 2017

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра АВТОМАТИКИ ТА УПРАВЛЯННЯ В ТЕХНІЧНИХ СИСТЕМАХ

Дисципліна «Об’єктно-орієнтоване програмування»

Курс 2 Група ЗПІ-зп61 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента**

|  |
| --- |
| Макіян Смбат Артурович |
| (прізвище, ім’я, по батькові) |

|  |  |
| --- | --- |
| 1. Тема роботи | Програма конструктор рівней гри “Battle City” |
|  | |
|  | |
|  | |

|  |  |
| --- | --- |
| 2. Строк здачі студентом закінченої роботи |  |

|  |
| --- |
| 3. Вихідні дані до роботи: |
| Операційна система Windows 8.1 та вище, платформа x86\_64 |
|  |
|  |

|  |
| --- |
| 4. Зміст розрахунково-пояснювальної записку (перелік питань, які підлягають розробці) |
| *1.Вступ. 2. Постановка задачі. 3. Огляд існуючих рішень та опис моделі.* |
| *4. Режим роботи та опис діалогу. 5. Опис програми. 6. Інструкція програміста.* |
| *7. Керівництво користувача 8. Висновки.* |
| *9. Список використаних джерел.* |
| ***Додатки:*** |
| *10.Текст програми. 11. Результати працездатності програми.* |

|  |
| --- |
| 5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 6. Дата видачі завдання |  |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Назва етапів виконання курсової роботи** | **Строк виконання етапів роботи** | **Підписи або примітки** |
| 1. | Отримання та узгодження теми курсової роботи |  |  |
| 2. | Огляд існуючих рішень з тематики роботи |  |  |
| 3. | Розробка моделі та структури програми |  |  |
| 4. | Вибір алгоритмів розв’язання задачі |  |  |
| 5. | Розробка діалогового інтерфейсу програми |  |  |
| 6. | Кодування програми |  |  |
| 7. | Налагодження та перевірка програми |  |  |
| 8. | Оформлення пояснювальної записки |  |  |
| 9. | Захист курсової роботи |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Керівник** |  |  |  |
|  | (підпис) |  | (прізвище, ім’я, по батькові) |

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2017 р.

# АНОТАЦІЯ

В курсовій роботі розглянуто та описано роботу конструктора рівней для гри “Battle city” з точки зору об’єктно-орієнтованого програмування. Основні сутності дослідженої предметної області описані як класи, між якими встановлені відповідні зв’язки. Програма розроблена об’єктно-орієнтованою мовою програмування C# в середовищі розробки Microsoft Visual Studio 2017. Використано шаблон Windows Forms Application, що дало змогу розробити дружній інтерфейс інформаційної системи. Вхідні та вихідні дані зберігаються в XML-файл.

Пояснювальна записка виконана на 41-х сторінках, містить 2 додатки, 9 рисунків.

ЗМІСТ

[ВСТУП 7](#_Toc484478325)

[1. ПОСТАНОВКА ЗАДАЧІ 8](#_Toc484478326)

[2. МАТЕМАТИЧНАЯ МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ 9](#_Toc484478327)

[3. ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ 11](#_Toc484478328)

[4. СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ 13](#_Toc484478329)

[4.1 Клас Level. 13](#_Toc484478330)

[4.2 Клас BaseGameElement. 13](#_Toc484478331)

[4.3 Клас BaseMovingGameElement 14](#_Toc484478332)

[4.4 Клас Brick 14](#_Toc484478333)

[4.5 Клас User 14](#_Toc484478334)

[4.6 Клас Enemy 14](#_Toc484478335)

[4.7 Глобальні об’єкти 15](#_Toc484478336)

[5. ОПИС ПРОГРАМИ 16](#_Toc484478337)

[5.1 Модульна структура 16](#_Toc484478338)

[5.2 Виклик і завантаження 16](#_Toc484478339)

[5.3 Повідомлення 17](#_Toc484478340)

[6. ІНСТРУКЦІЯ АДМІНІСТРАТОРА 18](#_Toc484478341)

[6.1 Загальні відомості 18](#_Toc484478342)

[6.2 Умови проектування програми 18](#_Toc484478343)

[6.3 Керівництво розробника програми 18](#_Toc484478344)

[6.4 Повідомлення підчас розробки 19](#_Toc484478345)

[6.4.1 Група повідомлень: 19](#_Toc484478346)

[6.4.2 Повідомлення: 20](#_Toc484478347)

[6.4.3 Повідомення: 20](#_Toc484478348)

[7. КЕРІВНИЦТВО КОРИСТУВАЧА 21](#_Toc484478349)

[7.1 Загальні відомості 21](#_Toc484478350)

[7.2 Умови застосування 21](#_Toc484478351)

[7.3 Характеристики програми 21](#_Toc484478352)

[7.4 Послідовність дій користувача програмою 21](#_Toc484478353)

[ВИСНОВКИ 27](#_Toc484478354)

[ПЕРЕЛІК ВИКОРИСТОВАНОЇ ЛІТЕРАТУРИ 28](#_Toc484478355)

[ДОДАТОК 29](#_Toc484478356)

[ДОДАТОК А. ТЕКСТ ПРОГРАМИ 29](#_Toc484478357)

[ДОДАТОК Б. РЕЗУЛЬТАТ ПРАЦЕЗДАТНОСТІ ПРОГРАМИ 39](#_Toc484478358)

# ВСТУП

Підчас розробки ігор в котрих не використовується процедурна генерація рівней, треба мати гнучкий та простий інструмент для ручної генерації рівней, в тому числи в грі “Battle city” (також звісної як «Танчики»).

Мета роботи – зробити прототип програми для генерації рівней, котра би мала графічній інтерфейс та дозволяла додавати, редагувати та видаляти рівні з подальшим збереженням цих рівней до xml файлу. Рівень представляє з себе матрицю 16х16, в кожній комірці котрого може знаходитись певний елемент, або нічого. На кожному рівні можуть буди 3 різних типи елементів: цегляні блоки, вороги та користувач. З кожен елемент мусить мати унікальні стартові координати для цього рівня.

При програмуванні застосовувалися основні методи об’єктно орієнтовного програмування [1]. Для збереження результатів в xml доцільно серіалізувати/десеріалізувати рівні через System.Runtime.Serialization[2]. Також програма має обробляти різні сітуації, в котрих користувач може зробити помилку: пусте ім’я рівня, невірні координати для елементу. Програма має бути написана на мові C# та використовувати фреймворка .Net версії 4.0. Для цієї мови доцільно вести розробку в середовищі Visual Studio (остання версія на момент написання 2017)[3].

# ПОСТАНОВКА ЗАДАЧІ

Для створення графічного інтерфейсу користувача потрібно використати framework Windows Forms. Програма має складатись заставки; головного вікна, в котрому можна додавати, видаляти та редагувати рівні; вікна для редагування рівня, в котрому можна задавати ім’я рівня, а також додавати, видаляти, редагувати та бачити статистику для елементів рівня; вікна для редагування елементу, в котрому можна задавати тип елементу та його координати; вікна для перегляду статистики елементу: тип, координати, та елементи, з котрими треба передивлятися перетин на оновленні.

Також доцільно розділити програму на UI та бізнес логіку: мати окрему бібліотеку з класами для елементів, а графічний інтерфейс скомпілювати до виконавчого файлу.

# МАТЕМАТИЧНАЯ МОДЕЛЬ ТА СТРУКТУРА ПРОГРАМИ

Результатом роботи програми має бути перелік рівней. Для цього найкращим образом підійде контейнер List, котра добре росте з початку, та наприкінці, і з котрої дуже легко видалити елемент з середини.

Рівні зберігають в собі елементи, котрі мають унікальні стартові координати. Контейнер Dictionary, котрий представляє з себе особу реалізацію бінарного дерева пошуку, створене з двох компонентів: ключ та значення, це дозволяє нам гарантувати унікальність стартових позицій елементів в рівні, про те кожний елемент гри має в собі стартову позицію, тому перелік елементів рівня доцільно зберігати також в контейнері List. Але сам елемент гри зберігає в собі свої стартові координати, та використовувати Dictionary не є доцільним, тому зберігатися елементи будуть в контейнері List, а доступ до певних координат та їх унікальність буде забезпечена методами самого рівня (див. Рисунок 2-1).

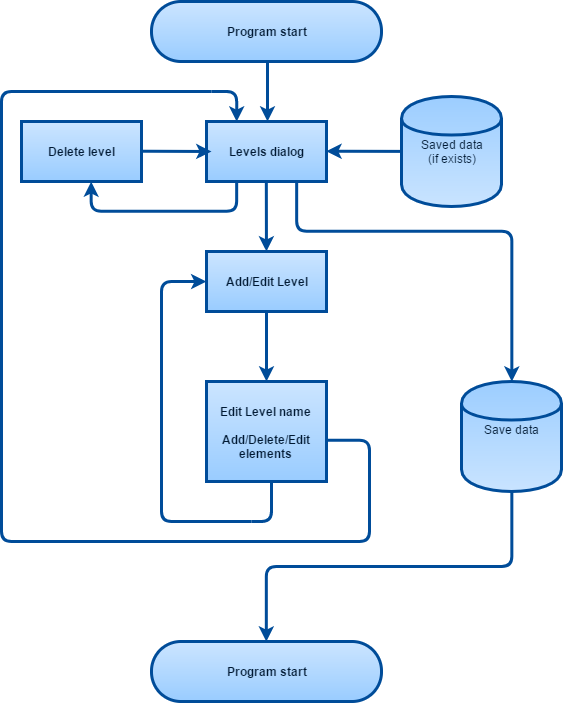


Рисунок .

# ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ

Програма починаетється з заставки, після котрої користувач бачить  діалогу редагування переліку рівней. В цьому діалозі має бути наступний функціонал:

* переглянути перелік доступних рівней (ListBox);
* додати рівень (Button для відкриття діалогу конструктора рівней);
* видалити рівень (Button для видалення вибраного рівня з переліку рівней);
* редагувати рівень (Button для відкриття діалогу конструктора рівней).
* Додавання або редагування рівня відбувається в діалозі конструктора рівня. В цьому діалозі користувач має:
* задати ім’я рівня (в текстовому полі);
* переглянути додані елементи (ListBox)
* додати елемент (Button для відкриття конструктора елемента);
* видалити елемент (Button для видалення вибранного елемента з ListBox);
* редагувати елемент (Button для відкриття конструктора елемента);
* відкрити статистику відповідно до елемента (Button для відкриття діалога статистики елемента);
* можливість відхилити зміни в рівні (Button);
* можливість зберегти зміни в рівні (Button).

Редагування та додавання елементу відбувається в конструкторі елементів. В цьому діалозі користувач може:

* вибрати тип елементу (з ListBox типів);
* вибрати координати елементу (два поля NumericUpDown);
* зберегти зміни (Button);
* відхилити зміни (Button).

Перегляд статистики елементу має бути в окремому діалозі статистики, де користувач нічого не може редагувати. В цьому діалозі користувачу має бути доступна наступна інформація:

* тип елементу (label);
* координати елементу ось X (label);
* координати елементу ось Y (label);
* список елементів, для Collision Detector (Listbox).

# СТРУКТУРА ДАНИХ ТА РЕСУРСІВ ПРОГРАМИ

Дані програми можна розділити на два класи: дані для Рівня та дані для Елементу. За дані для рівня відповідає клас Level. За дані для елементів відповідає клас BaseGameElement та його нащадки.

## Клас Level.

Призначен для зберігання даних про рівень. Зберігає ім’я рівня (string Name), його унікальний ідентифікатор (Guid Id), та список елементів (List< BaseGameElement> elements), також має статичну змінну static Level tempLevel для збереження тимчасових рівней.

Має наступні методи:

* конструктор;
* публічний getter та setter для поля Name;
* приватний setter та публічний getter для Id;
* приватний setter та публічний getter для elements;
* публічні методи void SetGameElement(BaseGameElement), BaseGameElement GetGameElement(Point), void RemoveElement(Point) для більш зручної роботи з полем elements;
* конструктор копіювання.

## Клас BaseGameElement.

Призначен для збереження даних про елементи. Є базовим класом для всіх елементів (BaseMovingGameElement, Brick, User, Enemy). Зберігає в собі ім’я елементу (string Name), тип статичності (Boolean IsStatic), начальну позицію (Point InitialPoint), та ідентифікатор рівня (Guid LevelId).

Має наступні методи:

* конструктор BaseGameElement(string name, Boolean stat, Point initPoint);
* публічний getter та приватний setter для поля Name;
* публічний setter та getter для LevelId;
* публічний getter та приватний setter для поля InitialPoint;
* публічний getter та приватний setter для поля IsStatic;
* публічний віртуальній List<BaseGameElement> CollidesWith() для того, щоб отримати список елементів, з котрими треба перевіряти на перетин при оновлені сцени.

## Клас BaseMovingGameElement

Є нащадком BaseGameElement, та батьком для классів User та Enemy. Не має ніяких додаткових полів, але має перегружений метод List<BaseGameElement> CollidesWith().

Має наступні методи:

* конструктор, котрий приймає стартову позицію;
* публічний віртуальній List<BaseGameElement> CollidesWith() для того, щоб отримати список елементів, з котрими треба перевіряти на перетин при апдейті.

## Клас Brick

Є нащадком BaseGameElement. Не має власних методів та полей, окрім конструктора і базових методів.

Має власний конструктор, котрий приймає стартову позицію.

## Клас User

Є нащадком BaseMovingGameElement. Не має власних методів та полей, окрім конструктора і базових методів.

Має власний конструктор, котрий приймає стартову позицію.

## Клас Enemy

Є нащадком BaseMovingGameElement. Не має власних методів та полей, окрім конструктора і базових методів.

Має власний конструктор, котрий приймає стартову позицію.

## Глобальні об’єкти

Єдиним глобальним об’єктом є List<Level> levels, у головній формі, для збереження переліку рівней.

# ОПИС ПРОГРАМИ

## Модульна структура

Програма створена з двох окремих компнентів Makiyan\_Cursovaya\_sem2 та Makiyan\_cursovaya\_sem2.Data. В першому компоненті знаходяться форми, у другому класи та бізнес логіка (див. Рисунок 5.1).

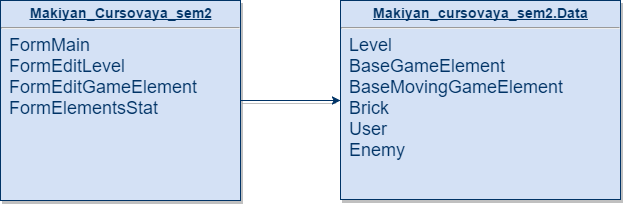


Рисунок .

## Виклик і завантаження

Точкою завантаження програми є функція public FormMain(), з класу FormMain з FormMain.cs. Програма завантажується як люба стандартна програма Windows: по подвійному кліку лівої кнопки миши або поклавіши Enter.

Вхідні дані

Програма може загрузити вхідні дані з файлу “level.xml”, якщо такий існує поряд з виконавчим файлом. Якщо такого файлу нема, а бо дані в ньому неможливо десереалізувати, програма запускається без вхідних даних.

Вихідні дані

Програма на виході генерує файл “level.xml”, котрий є результатом сереалізації List<Level> levels.

Корневим елементом є ArrayOfLevel, в котрому розположені окремі Level.

В Level зебережено наступні ноди: Id, Name, elements в котрому збережені окреми BaseGameElement.

В BaseGameElement збережено наступні ноди: Name, LevelId, InitialPoint (d5p1:x, dp5p1:y), IsStatic.

## Повідомлення

Підчас роботи з програмою користувач може получити наступні повідомлення:

* при спробі зберегти рівень з пустим ім’ям, користувач отримаю повідомлення з наступним текстом "Level name shouldn't be empty";
* при спробі зберегти рівень, з ім’ям котре зайняте іншим рівнем "Another level with same name already exists".

# ІНСТРУКЦІЯ АДМІНІСТРАТОРА

## Загальні відомості

Програма конструктор рівней для гри “Battle City”, назва програми Makiyan\_Cursovaya\_sem2.

## Умови проектування програми

Для написання програми потрібно ПК з наступними компонентами:

* середа програмування Microsoft Visual Studio 2017, з пакетом для роботи з C#;
* Windows 8.1 (або новіша), архітектура х86/x64, мінімальна конфігурація згідно з вимогами середовища VS 2017 (CPU 2x 1.8 GHz, 4 GB Ram, 40 GB hard disk space, video card minimal support resolution 720p and higher);
* стандартна комп’ютерна периферія: миша, клавіатура, монітор від 720p 32 bit.

## Керівництво розробника програми

Встановіть Microsoft Visual Studio 2017; при інсталяції виберіть пакет для роботи Desktop Visual C#.

Запустіть VS2017, та створіть новий проект Visual C#/Windows Forms. Вкажіть ім’я для цього проекту. Створіть в цьому проекті ще один проект Visual C#/Class library .Net Framework [5]. Задайте ім’я цьому проекту. Перейдіть до першого проекту з формами, та додайте reference на проект бібліотеки. Розробку бізнес логіки треба вести у проекті з бібліотекою, а формами в головному проекті. Для додання нової форми, перейдіть до головного проекту, натисніть на його ім’я правою кнопкою миші, виберіть Add/New Item/Visual C#/Windows form. Для додавання нового класу, перейдіть до проекту з класами, натисніть правою кнопкою миші, на його імені, та виберіть Add/New Item/Visual C#/Class file.

Для того щоби додати нові елементи до форми, двічи натисніть на файл форми поки не відкриється редактор форми. Відкрийте Toolbox, знайдіть потрібний вам компонент, та перетягніть його на форму. В Properties ви можете редагувати, ім’я та візуальні характеристики об’єкту. В Properties/Events ви можете редагувати системні події, наприклад, зміну фокусу або натиск кнопками миші.

При розробці програми для збереження результатів, нам знадобиться Serializer, з System.Runtime.Seiralization. Для цього натисніть правою кнопкою миші на імені проекті Add/Reference/Framework/System.Runtime.Serialization. Тим самим образом треба добавити System.Drawing.

Для настройки типу зборки перейдіть до Build/Configuration Manager, та застосуйте стандартні налаштування для створення конфігурації Release x86.

Після написання бізнес логіки, і створення форм, запустіть Build/Build Solution. Якщо сборка пройшла вдало, в папці проекту має з’явитись папка bin/x86/Release, з виконавчим файлом та бібліотекою, та декілька допоміжних файлів (важливі файли BattleCityLevelConstructor.exe та BattleCityLevelConstructor.Data.dll). Якщо скопіювати цю папку в файлами до іншого комп’ютера з Windows 8.1 та новіших версій (зі всіма встановленими оновленнями), то можна користуватися програмою, без будь яких додаткових дій.

## Повідомлення підчас розробки

### Група повідомлень:

* error CS0246: The type or namespace name 'DataContractAttribute' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'DataContract' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'KnownTypeAttribute' could not be found (are you missing a using directive or an assembly reference?)
* error CS0246: The type or namespace name 'KnownType' could not be found (are you missing a using directive or an assembly reference?)

Ця група повідомлень з’являється коли розробник, хоче скористуватись системною серіалізацією, але забув підключити бібліотеку System.Runtime.Serialization.

### Повідомлення:

* 'Type 'Makiyan\_cursovaya\_sem2.Data.Brick' with data contract name 'Brick:http://schemas.datacontract.org/2004/07/Makiyan\_cursovaya\_sem2.Data' is not expected. Consider using a DataContractResolver if you are using DataContractSerializer or add any types not known statically to the list of known types - for example, by using the KnownTypeAttribute attribute or by adding them to the list of known types passed to the serializer.'

Це повідомлення свідчить про те, що система не знає за яким принципом треба серіалізовати нащадка BaseGameElement, тому потрібно додати всіх нащадків до KnownType переліку. Для цього потрібно, під [DataContract] в BaseGameElement додати наступне:

[KnownType(typeof(Brick))]

[KnownType(typeof(BaseMovingGameElement))]

[KnownType(typeof(Enemy))]

[KnownType(typeof(User))]

### Повідомення:

* Error 1 Inconsistent accessibility: return type SOME\_TYPE\_1 is less accessible than method SOME\_METHOD\_2

Це повідомлення свідчить про те, що розробник хочу повернути з більш публічного методу об’єкт з меншим рівнем доступу. Наприклад, в публічного методу об’єкт приватного класу. Для цього або треба зменшити рівень доступу методу, або підвищити рівень доступу об’екта.

# КЕРІВНИЦТВО КОРИСТУВАЧА

## Загальні відомості

Програма конструктор програма BattleCityLevelConstructor. Ця програма є конструктором рівней для гри “Battle city”.

## Умови застосування

Для застосування програми нам знадобиться комп’ютер з операційною системою Windows 8.1 та новіша, та платформа x86 або x64, з усіма встановленими оновленнями та встановленим .Net framework рівня 4.0. Також потрібно вільне місце на жорсткому дисці більше ніж 10 Mb. Також бажано мати роздільну здатність екрана 1280х720 та більш вискоу для комфортної работи.

Комп’ютер має бути обладнаним стандартною комп’ютерною периферією: миша, клавіатура, монітор.

## Характеристики програми

Для коректного запуску програми потрібно, щоб в папці з програмою був файл “leve.xml” згенерований самою програмою, або його відсутність.

Інсталяція та видалення

Програма встановлюється методом копіювання папки з програмою на жорсткий діск (або роботу з флеш накопичувача). Для того щоби видалити програму, достатньо видалити папку програми.

## Послідовність дій користувача програмою

Для початку роботи, нічого не треба налаштовувати. Якщо файл “level.xml”, був пошкоджен, або модифікован із зовні, то для коректної роботи програми може знадобиться його видалити, з папці з програмою.

При відкритті користувач бачить перед собою заставку (зі звуком початку гри Battle city, див рисунок 7.1) після чого відкриється головне вікно (див рисунок 7.2). В котрому йому доступно додати, видалити або редагувати рівень. Якщо користувач хочу видалити рівень, йому потрібно вибрати рівень зі списку, та натиснута кнопку “Remove level”. Якщо додати рівень, натиснути кнопку “Add level”. Для редагування, треба вибрати потрібний рівень зі списку, та натиснути кнопку “Edit”. Редагування та додаток рівня, відкривають нове вікно редактора рівня.

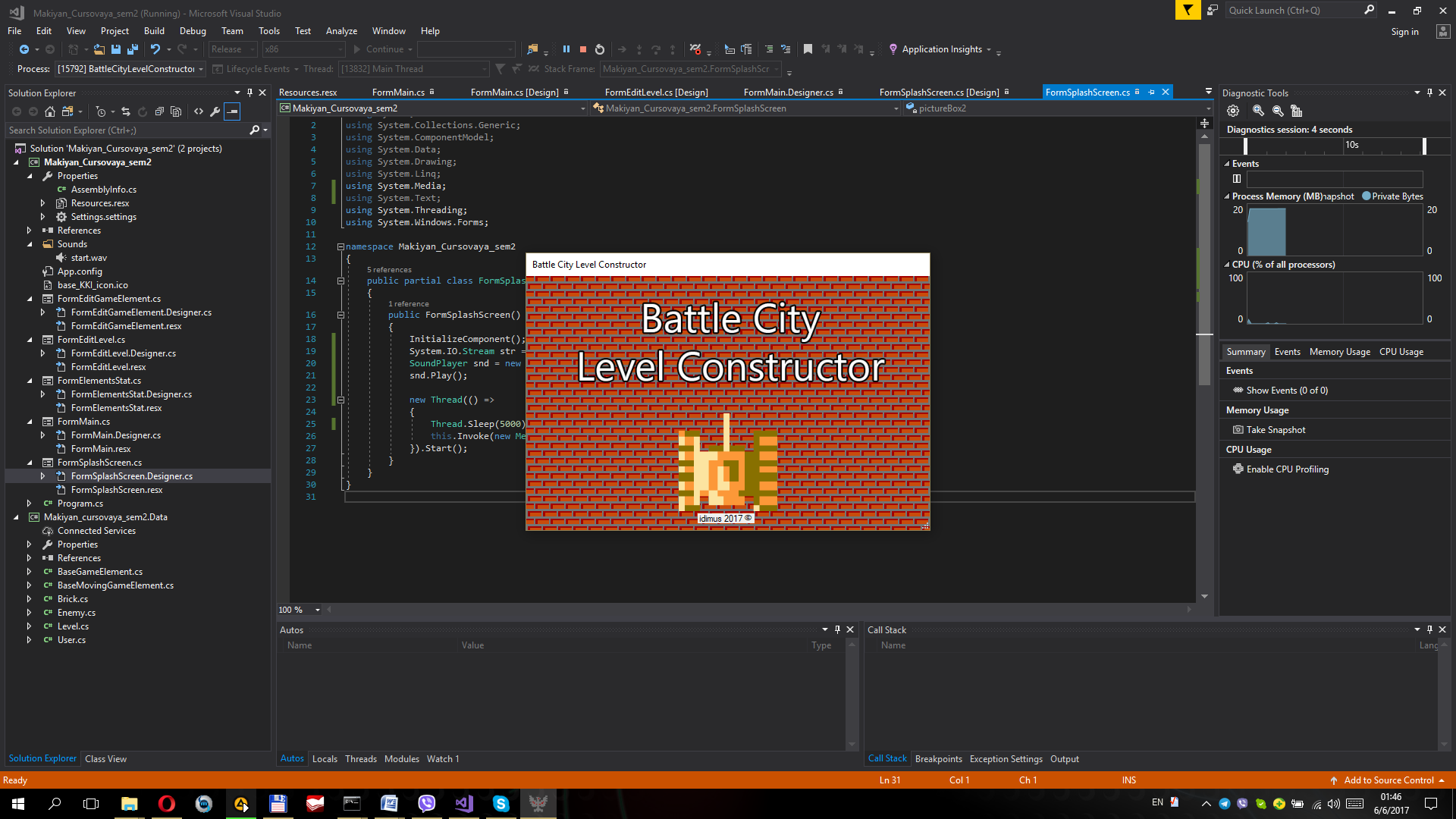


Рисунок .

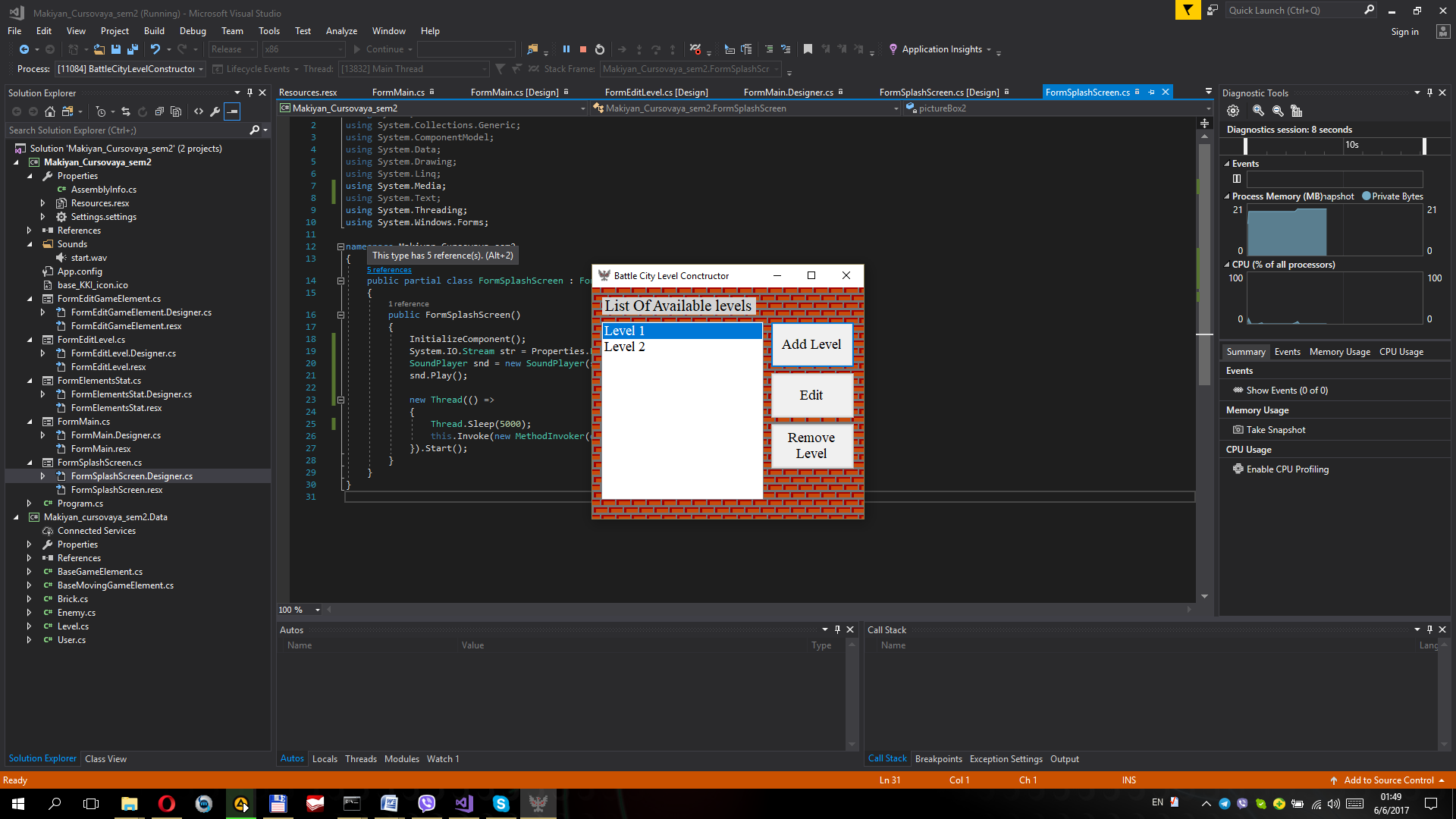


Рисунок .

Вікно редагування рівня, дозволяє нам вибрати ім’я для рівня, та додати, видалити, редагувати елемент, передивитись статистику елементу, та зберегти, чи відхилити наші зміни стосовно для рівня. Для збереження реба натиснути кнопку “Save”, відхилити “Cancel”. Для того щоби задати ім’я рівня, треба змінити текст, в текстовому полі “Level name”. Якщо користувач хочу видалити елемент, йому потрібно вибрати рівень зі списку, та натиснута кнопку “Remove”. Якщо додати елемент, натиснути кнопку “Add”. Для редагування, треба вибрати потрібний елемент зі списку, та натиснути кнопку “Edit”. Редагування та додаток елементу, відкривають нове вікно редактора елементу. Для перегляду статистики елементу треба вибрати лемент зі списку, а натиснути кнопку “Statistics”, після чого відкриється вікно статистики (див. рисунок 7.3).

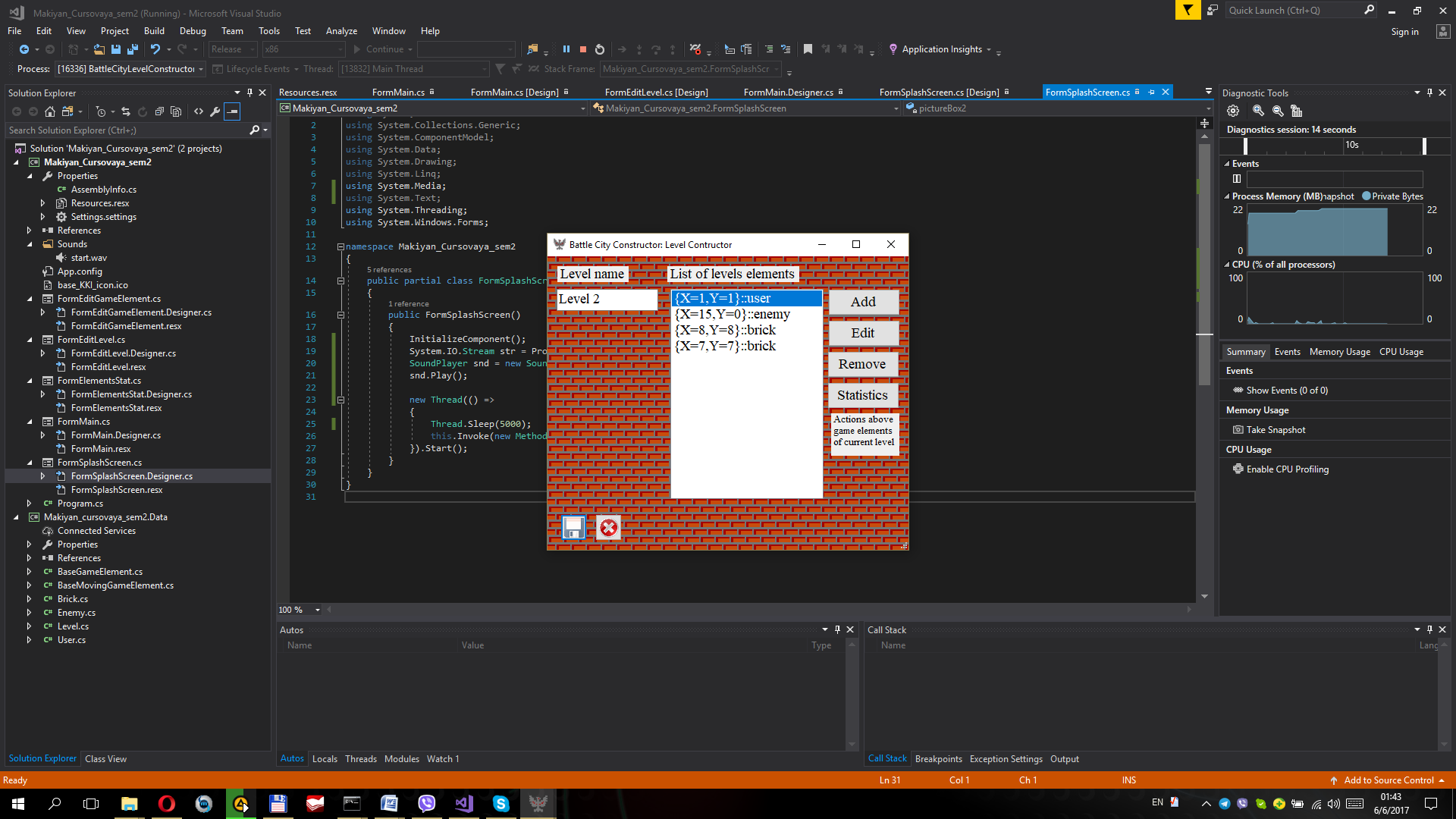


Рисунок .

Вікно редагування елементу, дозволяє користувачу визначити тип елементу, та його початкові координати X, Y. X та Y мають власні поля для вводу цифр від нуля до 16-ті. Не дивлячись на те, що туди можна ввести число з точкою, при збереженні результату, цифри після точки не будуть враховуватись. Для вибору типу, потрібно вибрати один з 3 типів елементів.

Для збереження результату треба натиснути на іконку дискети, для відхилення на круглий червоний хрестик (див. рисунок 7.4).

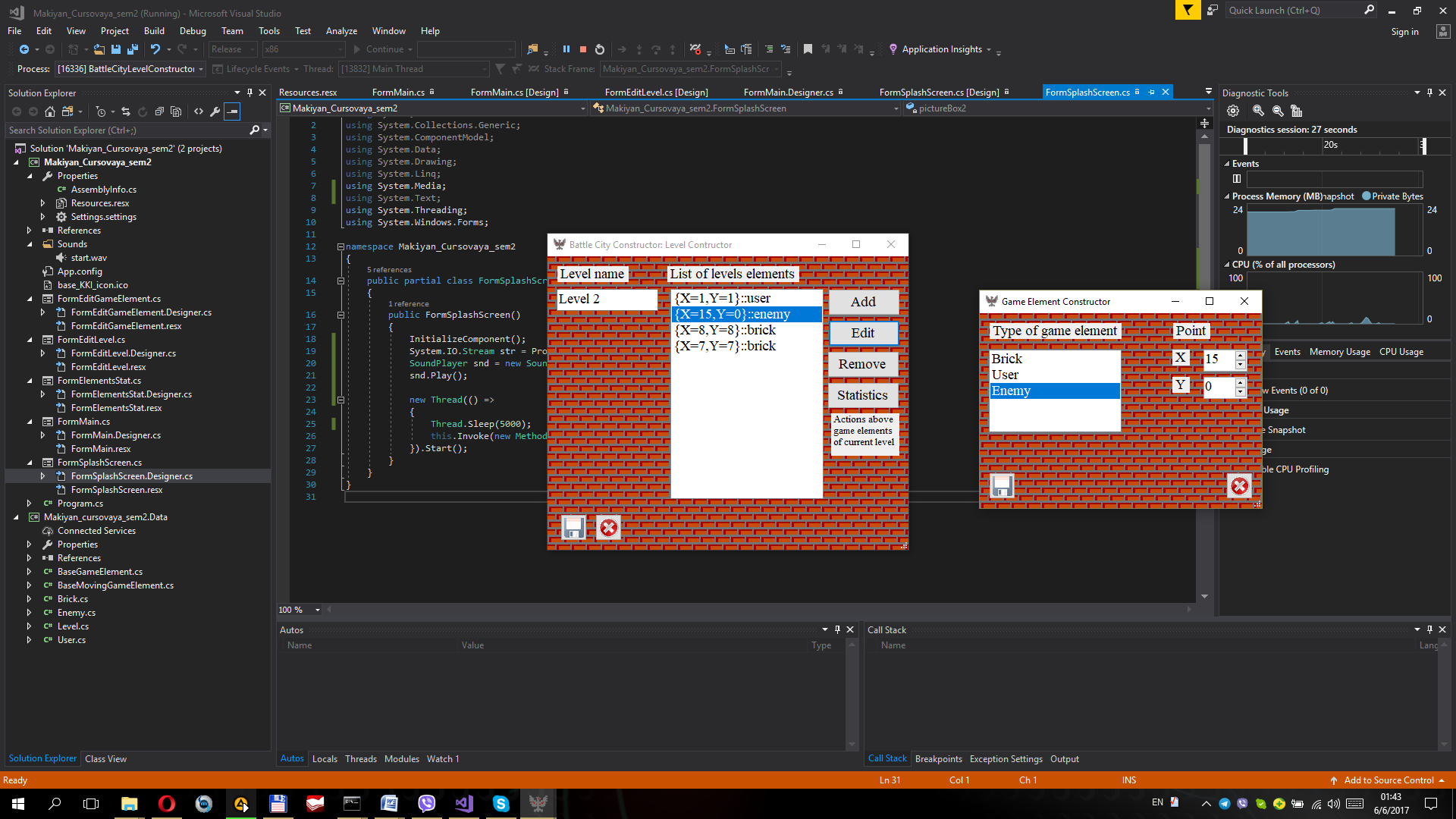


Рисунок .

Вікно статистики не має в собі ні яких контролюючих елементів, і відображає тип рівня, його координати, та список елементів, з якими він може реагувати (див. рисунок 7.5).

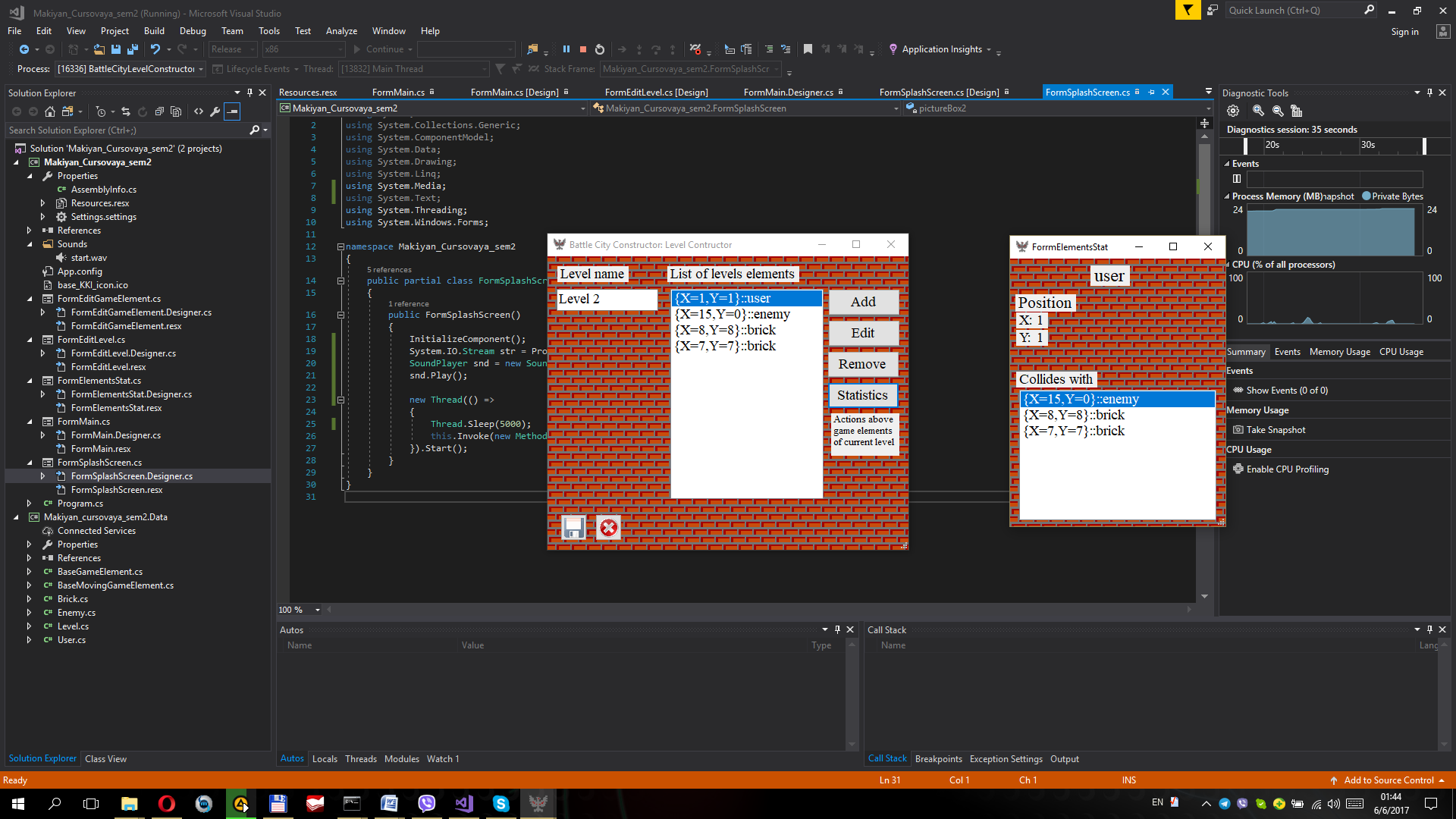


Рисунок .

Підчас роботи з програмою користувач може бачити віконце з повідомленням під час збереження рівня. Це вікно повідомляє про те, що користувач хоче ввести некоректну пусту назву рівня. Для того щоби вікно пропало, достатньо перед збереженням ввести не пусте ім’я рівня (див. Малюнок 7.6).

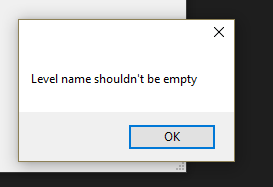


Рисунок .

Також користувач може бачити віконце з повідомленням під час збереження рівня. Це вікно повідомляє про те, що користувач хоче ім’я вже зайнятого рівня. Для цього достатньо змінити ім’я рівня (див. рисунок 7.7).

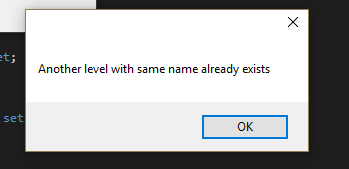


Рисунок .

# ВИСНОВКИ

Під час курсової роботи було отримане вміння писати програми на мові C# з використанням .Net Framework (v. 4.0) та Windows forms. На виході ми отримали повноцінну робочу програму, котра вміє зберігати/редагувати рівні для гри “Battle city”. Перехід від одного вікна до іншого вікна в програмі відбувається інтуїтивно. Результати програми зберігаються до xml фалу напри кінці, та зчитуються з нього в початку роботи програми. Програма реагує на спроби ввести хибні дані, та видає повідомлення згідно до помилки.

При розробці програми використовувалися основні методи об’єктно орієнтованого програмування [1] інкапсуляція, поліморфізм, наслідування [4].

Так загалом створено 1 основний та 4 додаткові класи, які реалізуються шляхом наслідування від основного. Для кожного типу об’єкта забезпечено свій окремий клас.

# ПЕРЕЛІК ВИКОРИСТОВАНОЇ ЛІТЕРАТУРИ

1. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4 [Текст] / Э. Троелсен : Москв, ИЛ, 2011. – 331 с.
2. Ярошенко Ф. Проектное управление. Управление инновационными проектами и программами [Текст] / Ф. Ярошенко : 2011. – 268 с.
3. Microsoft corp. Средства и языки разработки [Електронний ресурс] – Режим доступу <https://msdn.microsoft.com/ru-ru/library/aa187916.aspx>
4. Паппас К. Программирование на С и C++ [Текст] : Издание второе / К. Паппас, У. Мюррей : Киев: «Ирина»; BHV, 2000. – 824 с.
5. Microsoft corp. Документация по Visual Studio [Електронний ресурс] – Режим доступу <https://docs.microsoft.com/ru-ru/visualstudio/>
6. Майо Д. С# Builder. Быстрый старт [Текст] / Д. Майо : Бином-Пресс, 2005. – 384 с.

# ДОДАТОК

## ДОДАТОК А. ТЕКСТ ПРОГРАМИ

**BaseGameElement.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Drawing;

using System.Text;

namespace Makiyan\_cursovaya\_sem2.Data

{

[DataContract]

[KnownType(typeof(Brick))]

[KnownType(typeof(BaseMovingGameElement))]

[KnownType(typeof(Enemy))]

[KnownType(typeof(User))]

public class BaseGameElement

{

[DataMember]

public string Name { get; private set; }

[DataMember]

public Boolean IsStatic { get; private set; }

[DataMember]

public Point InitialPoint { get; private set; }

[DataMember]

public Guid LevelId { get; set; }

public BaseGameElement(string name, Boolean stat, Point initPoint)

{

Name = name;

IsStatic = stat;

InitialPoint = initPoint;

}

public virtual List<BaseGameElement> CollidesWith()

{

return new List<BaseGameElement>();

}

}

}

BaseMovingGameElement.cs

using System.Collections.Generic;

using System.Drawing;

using System.Runtime.Serialization;

namespace Makiyan\_cursovaya\_sem2.Data

{

[DataContract]

public class BaseMovingGameElement : BaseGameElement

{

public BaseMovingGameElement(string name, Point initPoint) : base(name, false, initPoint)

{

}

public override List<BaseGameElement> CollidesWith()

{

List<BaseGameElement> el = new List<BaseGameElement>();

Level lv = Level.tempLevel;

List<BaseGameElement> res = new List< BaseGameElement>();

foreach (BaseGameElement mm in lv.elements)

{

if (mm.InitialPoint != InitialPoint)

{

res.Add(mm);

}

}

return res;

}

}

}

**Brick.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

namespace Makiyan\_cursovaya\_sem2.Data

{

[DataContract(Name ="Brick")]

[KnownType(typeof(BaseGameElement))]

public class Brick : BaseGameElement

{

public Brick(Point initPoint) : base("brick", true, initPoint)

{

}

}

}

**Enemy.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

using System.Runtime.Serialization;

namespace Makiyan\_cursovaya\_sem2.Data

{

[DataContract(Name ="Enemy")]

public class Enemy : BaseMovingGameElement

{

public Enemy(Point initPoint) : base("enemy", initPoint)

{

}

}

}

**User.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

namespace Makiyan\_cursovaya\_sem2.Data

{

[DataContract(Name = "User")]

public class User : BaseMovingGameElement

{

public User(Point initPoint) : base("user", initPoint)

{

}

}

}

**FormMain.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Makiyan\_cursovaya\_sem2.Data;

using System.Runtime.Serialization;

using System.Xml;

using System.IO;

namespace Makiyan\_Cursovaya\_sem2

{

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

public FormMain()

{

FormSplashScreen sp = new FormSplashScreen();

sp.ShowDialog();

InitializeComponent();

try

{

FormMain.levels = (List<Level>)Load("level.xml", typeof(List<Level>));

RefreshLevelsList();

}

catch { }

BringToFront();

}

private void button2\_Click(object sender, EventArgs e)

{

FormMain.levels.Remove((Level)LevelsList.SelectedItem);

RefreshLevelsList();

}

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

{

}

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

{

try

{

Level lv = null;

string name = LevelsList.SelectedItem.ToString();

foreach (Level l in FormMain.levels)

{

if (l.Name == name)

{

lv = l;

break;

}

}

new FormEditLevel(lv).ShowDialog();

RefreshLevelsList();

}

catch { }

}

private void Save(object obj, string filaName)

{

DataContractSerializer dcs = new DataContractSerializer(obj.GetType());

XmlWriter xmlw = XmlWriter.Create(filaName);

dcs.WriteObject(xmlw, obj);

xmlw.Close();

}

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

{

Save(FormMain.levels, "level.xml");

}

private object Load(string filaName, Type t)

{

if (File.Exists("level.xml"))

{

DataContractSerializer dcs = new DataContractSerializer(t);

XmlReader xmlr = XmlReader.Create(filaName);

object res = dcs.ReadObject(xmlr);

xmlr.Close();

return res;

}

return new List<Level>();

}

}

}

**FormEditLevel.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Makiyan\_cursovaya\_sem2.Data;

namespace Makiyan\_Cursovaya\_sem2

{

public partial class FormEditLevel : Form

{

private Level currentLevel;

public FormEditLevel(Level lv = null)

{

InitializeComponent();

if (lv == null)

currentLevel = new Level();

else

{

currentLevel = lv;

name.Text = currentLevel.Name;

}

Level.tempLevel = new Level( currentLevel);

refreshGameElementsList();

}

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

{

}

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

{

if (name.Text.Trim().Length == 0)

{

name.Text = "";

MessageBox.Show("Level name shouldn't be empty");

}

else

{

string sName = name.Text.Trim();

bool shouldAddLevel = true;

bool shouldClose = true;

foreach (Level l in FormMain.levels)

{

if (sName == l.Name)

{

shouldAddLevel = false;

if (currentLevel.Id == l.Id)

{

currentLevel = new Level(Level.tempLevel);

int index = FormMain.levels.IndexOf(l);

FormMain.levels[index] = currentLevel;

}

else

{

shouldClose = false;

MessageBox.Show("Another level with same name already exists");

}

break;

}

else if (currentLevel.Id == l.Id)

{

shouldAddLevel = false;

Level.tempLevel.Name = sName;

currentLevel = new Level(Level.tempLevel);

int index = FormMain.levels.IndexOf(l);

FormMain.levels[index] = currentLevel;

break;

}

}

if (shouldClose)

{

currentLevel.Name = sName;

}

if (shouldAddLevel)

{

Level.tempLevel.Name = sName;

currentLevel = new Level(Level.tempLevel);

FormMain.levels.Add(currentLevel);

}

if (shouldClose)

{

Close();

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

Close();

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

try

{

string pointAndName = listBoxGameObjects.SelectedItem.ToString();

string pointName = pointAndName.Split(':')[0];

foreach (char i in "{XY=}")

{

string ss = "" + i;

pointName = pointName.Replace(ss, "");

}

string[] coords = pointName.Split(',');

Point point = new Point(int.Parse(coords[0]), int.Parse(coords[1]));

Level.tempLevel.RemoveEelement(point);

refreshGameElementsList();

}

catch

{ }

}

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

{

new FormEditGameElement(Level.tempLevel, new Point(-1, -1)).ShowDialog();

refreshGameElementsList();

}

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

{

try

{

string pointAndName = listBoxGameObjects.SelectedItem.ToString();

string pointName = pointAndName.Split(':')[0];

foreach (char i in "{XY=}")

{

string ss = "" + i;

pointName = pointName.Replace(ss, "");

}

string[] coords = pointName.Split(',');

Point point = new Point(int.Parse(coords[0]), int.Parse(coords[1]));

new FormEditGameElement(Level.tempLevel, point).ShowDialog();

refreshGameElementsList();

}

catch

{ }

}

private void refreshGameElementsList()

{

listBoxGameObjects.DataSource = null;

List<string> gameElementsList = new List<string>();

foreach ( BaseGameElement el in Level.tempLevel.elements)

{

gameElementsList.Add(el.InitialPoint.ToString() + "::" + el.Name);

}

listBoxGameObjects.DataSource = gameElementsList;

}

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

{

try

{

string pointAndName = listBoxGameObjects.SelectedItem.ToString();

if (pointAndName.Trim().Length == 0)

{

return;

}

string pointName = pointAndName.Split(':')[0];

foreach (char i in "{XY=}")

{

string ss = "" + i;

pointName = pointName.Replace(ss, "");

}

string[] coords = pointName.Split(',');

Point point = new Point(int.Parse(coords[0]), int.Parse(coords[1]));

new FormElementsStat(Level.tempLevel, point).ShowDialog();

}

catch { }

}

private void name\_TextChanged(object sender, EventArgs e)

{

}

}

}

**FormEditGameElement.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Makiyan\_cursovaya\_sem2.Data;

namespace Makiyan\_Cursovaya\_sem2

{

public partial class FormEditGameElement : Form

{

private Level level;

private Point poinClass;

private List<String> availableTypes = new List < String >{ "Brick", "User", "Enemy" };

public FormEditGameElement(Level crLevel, Point p)

{

InitializeComponent();

listBoxType.DataSource = availableTypes;

level = crLevel;

poinClass = p;

if (poinClass != new Point(-1,-1))

{

BaseGameElement bs = level.GetGameElement(poinClass);

Point dummyPoint = new Point(-1, -1);

if (bs.Name == new Brick(dummyPoint).Name)

{

listBoxType.SetSelected(0, true);

}

else if (bs.Name == new User(dummyPoint).Name)

{

listBoxType.SetSelected(1, true);

}

else

{

listBoxType.SetSelected(2, true);

}

numericUpDownX.Value = poinClass.X;

numericUpDownY.Value = poinClass.Y;

}

}

private void button2\_Click(object sender, EventArgs e)

{

Close();

}

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

{

Point p = new Point((int)numericUpDownX.Value, (int)numericUpDownY.Value);

string type = (string)listBoxType.SelectedItem;

BaseGameElement gameEl = null;

if (type == "Brick")

{ gameEl = new Brick(p); }

else if (type == "User")

{ gameEl = new User(p); }

else if (type == "Enemy")

{ gameEl = new Enemy(p); }

gameEl.LevelId = level.Id;

level.SetGameElement(gameEl);

Close();

}

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

{

}

}

}

**FormElementsStat.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Makiyan\_cursovaya\_sem2.Data;

namespace Makiyan\_Cursovaya\_sem2

{

public partial class FormElementsStat : Form

{

private Level currentLevel;

private Point point;

public FormElementsStat(Level currentLevel, Point point)

{

InitializeComponent();

this.currentLevel = currentLevel;

this.point = point;

BaseGameElement bs = currentLevel.GetGameElement(point);

name.Text = bs.Name;

labelXCoord.Text = this.point.X.ToString();

labelYCoord.Text = this.point.Y.ToString();

listBoxCollide.DataSource = null;

List<string> gameElementsList = new List<string>();

foreach ( BaseGameElement el in bs.CollidesWith())

{

gameElementsList.Add(el.InitialPoint.ToString() + "::" + el.Name);

}

listBoxCollide.DataSource = gameElementsList;

}

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

{

}

}

}

**FormSplashScreen.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Windows.Forms;

namespace Makiyan\_Cursovaya\_sem2

{

public FormSplashScreen()

{

InitializeComponent();

System.IO.Stream str = Properties.Resources.start;

SoundPlayer snd = new SoundPlayer(str);

snd.Play();

new Thread(() =>

{

Thread.Sleep(5000);

this.Invoke(new MethodInvoker(delegate { Close(); }));

}).Start();

}}

## ДОДАТОК Б. РЕЗУЛЬТАТ ПРАЦЕЗДАТНОСТІ ПРОГРАМИ

**Текст файлу “level.xml”**

<?xml version="1.0" encoding="utf-8"?>

<ArrayOfLevel

xmlns:i="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://schemas.datacontract.org/2004/07/Makiyan\_cursovaya\_sem2.Data">

<Level>

<Id>a09da13b-54b7-4cda-b64a-b8d0d346e7b9</Id>

<Name>Level 1</Name>

<elements>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>0</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>a09da13b-54b7-4cda-b64a-b8d0d346e7b9</LevelId>

<Name>brick</Name>

</BaseGameElement>

<BaseGameElement i:type="User">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>3</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>a09da13b-54b7-4cda-b64a-b8d0d346e7b9</LevelId>

<Name>user</Name>

</BaseGameElement>

<BaseGameElement i:type="Enemy">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>12</d5p1:x>

<d5p1:y>12</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>a09da13b-54b7-4cda-b64a-b8d0d346e7b9</LevelId>

<Name>enemy</Name>

</BaseGameElement>

<BaseGameElement i:type="Enemy">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>5</d5p1:x>

<d5p1:y>5</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>a09da13b-54b7-4cda-b64a-b8d0d346e7b9</LevelId>

<Name>enemy</Name>

</BaseGameElement>

</elements>

</Level>

<Level>

<Id>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</Id>

<Name>Level 2</Name>

<elements>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>0</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>brick</Name>

</BaseGameElement>

<BaseGameElement i:type="User">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>1</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>user</Name>

</BaseGameElement>

<BaseGameElement i:type="Enemy">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>0</d5p1:x>

<d5p1:y>1</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>enemy</Name>

</BaseGameElement>

<BaseGameElement i:type="User">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>5</d5p1:x>

<d5p1:y>5</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>user</Name>

</BaseGameElement>

<BaseGameElement i:type="Enemy">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>0</d5p1:x>

<d5p1:y>5</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>enemy</Name>

</BaseGameElement>

<BaseGameElement i:type="Enemy">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>0</d5p1:x>

<d5p1:y>12</d5p1:y>

</InitialPoint>

<IsStatic>false</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>enemy</Name>

</BaseGameElement>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>10</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>brick</Name>

</BaseGameElement>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>11</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>brick</Name>

</BaseGameElement>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>12</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>brick</Name>

</BaseGameElement>

<BaseGameElement i:type="Brick">

<InitialPoint

xmlns:d5p1="http://schemas.datacontract.org/2004/07/System.Drawing">

<d5p1:x>13</d5p1:x>

<d5p1:y>0</d5p1:y>

</InitialPoint>

<IsStatic>true</IsStatic>

<LevelId>70601dd7-6fcd-4df1-9cae-bb60d5a896d6</LevelId>

<Name>brick</Name>

</BaseGameElement>

</elements>

</Level>

</ArrayOfLevel>