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  Жеребко В.А.  «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 |
|  |
|  |

|  |
| --- |
| 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.

# ЗМІСТ

# 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 добавляти, редагувати та видаляти рівні з подальшим збереженням ціх 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.

Программа 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). Для збереження результатів в xml доцільно серіалізувати/десеріалізувати рівні через System.Runtime.Serialization. Також програма має обробляти різні сітуації, в котрих користувач може зробити помилку: пусте ім’я рівня, невірні координати для елементу.

# ПОСТАНОВКА ЗАДАЧІ

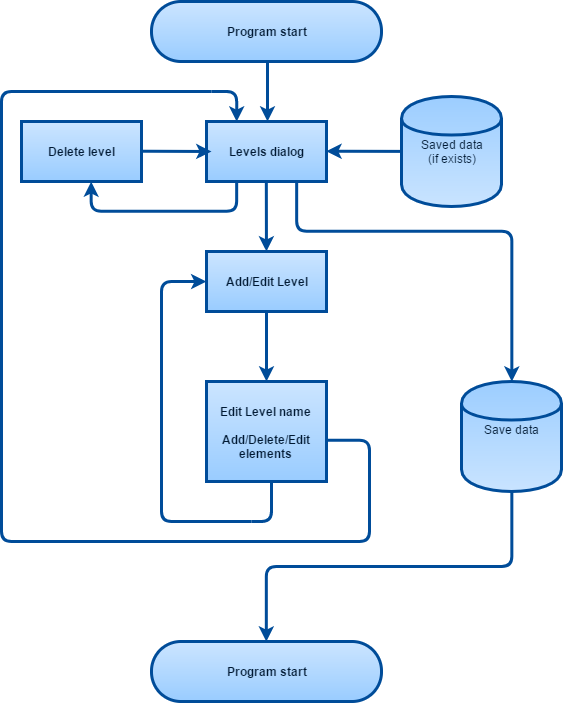
Для створення графічного інтерфейсу користувача потрібно використати framework Windows Forms. Програма має складатись з головного вікна, в котрому можна додавати, видаляти та редагувати рівні; вікна для редагування рівня, в котрому можна задавати ім’я рівня, а також додавати, видаляти, редагувати та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

Також доцільно розділити программу на UI та бізнес логіку: мати окрему бібліотеку з классами 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 на прикінці, 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, звторен 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.



# ПЕРЕЛІК І ПРИЗНАЧЕННЯ РЕЖИМІВ ТА СТРУКТУРА ДІАЛОГУ

Програма має починатись зі стартового 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, 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 апдейті.

**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.

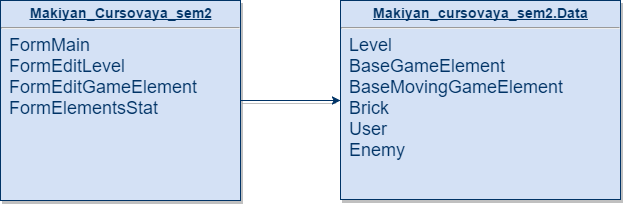
**Глобальні об’єкти**

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.



**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 распологаються окреми 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 ПРОГРАМІСТА

**Загальні відомості**

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

**Умови проектування програми**

Для написання програми потрібно ПК з наступними компонетами:

* 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. 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