

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра  
інформатики та програмної інженерії  
(повна назва кафедри, циклової комісії)

**КУРСОВА РОБОТА**

З Основ програмування  
(назва дисципліни)  
на тему: гра «Змійка»

Студента 1 курсу, групи ІІІ-11  
Лошака Віктора Івановича

Спеціальності 121 «Інженерія програмного забезпечення»

Керівник Головченко Максим Миколайович  
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: \_\_\_\_\_  
Національна оцінка \_\_\_\_\_

Члени комісії

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_  
(підпис)

\_\_\_\_\_  
(вчене звання, науковий ступінь, прізвище та ініціали)

Київ- 2022 рік

# КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

(назва вищого навчального закладу)

Кафедра інформатики та програмної інженерії

Дисципліна Основи програмування

Напрямок "ІПЗ"

Курс 1 Група ІП-11

Семестр 2

## ЗАВДАННЯ на курсову роботу студента Лошака Віктора Івановича

(прізвище, ім'я, по батькові)

1. Тема роботи Гра «Змійка»

2. Строк здачі студентом закінченої роботи 12.06.2022

3. Вихідні дані до роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

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

6. Дата видачі завдання 10.02.2022

# КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	10.02.2022	
2.	Підготовка ТЗ	02.05.2022	
3.	Пошук та вивчення літератури з питань курсової роботи	03.05.2022	
4.	Розробка сценарію роботи програми	04.05.2022	
6.	Узгодження сценарію роботи програми з керівником	04.05.2022	
5.	Розробка (вибір) алгоритму рішення задачі	04.05.2022	
6.	Узгодження алгоритму з керівником	04.05.2022	
7.	Узгодження з керівником інтерфейсу користувача	05.05.2022	
8.	Розробка програмного забезпечення	06.05.2022	
9.	Налагодження розрахункової частини програми	06.05.2022	
10.	Розробка та налагодження інтерфейсної частини програми	07.05.2022	
11.	Узгодження з керівником набору тестів для контрольного прикладу	25.05.2022	
12.	Тестування програми	26.05.2022	
13.	Підготовка пояснювальної записки	05.06.2022	
14.	Здача курсової роботи на перевірку	12.06.2022	
15.	Захист курсової роботи	15.06.2022	

Студент \_\_\_\_\_  
(підпис)

Керівник \_\_\_\_\_  
(підпис)

Головченко Максим Миколайович  
(прізвище, ім'я, по батькові)

" " \_\_\_\_\_ 2022 р.

## АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 100 сторінок, 5 рисунків, 16 таблиць, 1 посилання.

Об'єкт дослідження: гра «Змійка»

Мета роботи: дослідження методів розробки програмного забезпечення, дослідження методів обрах.

Вивчено метод розробки програмного забезпечення з використанням принципів ООП. Приведені змістовні постановки задач, їх індивідуальні логічні моделі, а також описано детальний процес розв'язання кожної з них.

Виконана програмна реалізація гри «Змійка».

## ЗМІСТ

<b>1 ПОСТАНОВКА ЗАДАЧІ .....</b>	<b>8</b>
<b>ВСТУП .....</b>	<b>9</b>
<b>2 ТЕОРЕТИЧНІ ВІДОМОСТІ .....</b>	<b>10</b>
2.1 Елементи гри «Змійка» .....	10
2.2 Правила гри «Змійка» .....	10
<b>3 ОПИС АЛГОРИТМІВ .....</b>	<b>12</b>
3.1 Загальний алгоритм .....	13
3.2 Алгоритм генерації поля .....	13
3.3 Обробка зміни стану гри .....	14
3.4 Алгоритм дій при виклику події Timer.Tick .....	14
3.5 Алгоритм оновлення ігрового поля .....	14
3.6 Алгоритм оновлення змійки .....	14
3.7 Алгоритм оновлення положення приза .....	15
3.8 Алгоритм додавання клітинок до змійки .....	16
<b>4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ .....</b>	<b>17</b>
4.1 Діаграма класів програмного забезпечення .....	17
4.2 Опис методів частин програмного забезпечення .....	17
4.2.1 Стандартні методи .....	17
4.2.2 Користувацькі методи .....	22

	6
<b>5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ</b>	42
5.1 План тестування	42
5.2 Приклади тестування	43
<b>6 ІНСТРУКЦІЯ КОРИСТУВАЧА</b>	50
6.1 Робота з програмою	50
6.2 Системні вимоги	52
<b>ВИСНОВОК</b>	53
<b>ПЕРЕЛІК ПОСИЛАНЬ</b>	54
<b>ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ</b>	55
<b>ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ</b>	58
DateTimeConverter.cs	59
SnakeFieldConverter.cs	60
Cell.cs	62
Field.cs	64
Food.cs	67
ScoreCounter.cs	69
Snake.cs	71
SnakeGameFileManager.cs	75
StatsItem.cs	77
DifficultyWindow.xaml	79

	7
<b>DifficultyWindow.xaml.cs</b> .....	81
<b>MainWindow.xaml</b> .....	83
<b>MainWindow.xaml.cs</b> .....	85
<b>SnakeGameWindow.xaml</b> .....	87
<b>SnakeGameWindow.xaml.cs</b> .....	90
<b>StatsWindow.xaml</b> .....	92
<b>StatsWindow.xaml.cs</b> .....	94
<b>ViewModel.cs</b> .....	96
<b>App.xaml</b> .....	99
<b>App.xaml.cs</b> .....	100

## 1 ПОСТАНОВКА ЗАДАЧІ

Розробити програмне забезпечення, що буде давати змогу переключатися між меню «гра», «статистика» та «складність», грати в гру «Змійка» згідно з правилами (2.1) та записувати статистику ігрових сесій в файл, з можливістю перегляду в меню статистики, а також змінювати складність гри в однойменному меню.

Кожна клітинка змійки так само як і клітинка призу має відповідати клітинці поля. Початкова довжина змійки має складати 4 клітинки, в ході гри довжина змійки повинна збільшуватись за допомогою додавання клітинок в кінець змійки. Клітинки змійки мають розташовуватись в ряд що досягається за допомогою алгоритму додавання клітинок. Вони можуть пересуватись тільки по прямій лінії (горизонтально або вертикально) головою вперед. Розмір призу в клітинках є сталим і складає 1 клітинку поля. Цінність призу в очках змінна залежно від режиму складності гри. Алгоритм дій програми при перетині ігрового кордону залежить від встановленого режиму складності. Зверху від поля мають відображатись кількість очків зароблених в ході поточної ігрової сесії та складність поточної ігрової сесії. Складність гри під час роботи програми може змінюватись але вона незмінна під час ігрової сесії.

В ході ігрової сесії користувач повинен мати можливість керувати змійкою на ігровому полі та здійснювати ігровий процес, призупиняти та відновлювати ігрову сесію після паузи, закінчувати ігрову сесію. Автоматично повинно задаватись початкове розміщення змійки та призу, оновлюватись кількість зароблених гравцем очків під час гри, зберігатись результати гри при її закінченні. Вхідними даними для ігрової сесії є режим складності а вихідними— кількість очок набраних в ході сесії. Ігрові об'єкти повинні бути реалізовані через класи.



## **ВСТУП**

Дана робота присвячена вивченню розробки програмного забезпечення з використанням парадигми ООП, і стосується написання комп'ютерної гри «Змійка». Задача полягає у графічному представленні даної гри та реалізації партії.

## 2 ТЕОРЕТИЧНІ ВІДОМОСТІ

### 2.1 Елементи гри «Змійка»

Ігрове поле – квадрат  $20 \times 20$ , на якому розміщені елементи гри змійки.

Змійка -- в момент гри  $m$  може складатися з  $n$  елементів. Змійка має голову і хвіст і рухається лише в сторону голови. Ребро змійки може бути спільним не більш ніж для двох клітинок змійки. З однієї клітинки змійки прохід в іншу можливий лише через спільне ребро. Клітинки змійки можуть бути розташовані лише так, щоб кожна мала хоча б одне, але не більше двох спільних ребер з будь-якою із сусідніх клітинок змійки. Нехай  $F_i$  – квадратна одинична матриця переходів з однієї клітинки змійки в іншу, де 1—означає «можна пройти» а 0— «немає проходу», тоді  $i = i_1, i_2 \dots i_n$  – номери клітинок змійки починаючи з голови. Матриця має властивості симетричності і є діагональною. Змійка вважається «мертвою» якщо при будь який рядок/стовбець матриці містить більше трьох одиниць. «Головою змійки» називають клітинку що в матриці переходів займає перше місце в головній діагоналі. «Хвостом змійки» називають клітинку що в матриці переходів займає останнє місце в головній діагоналі

$$F_i = \begin{pmatrix} 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & \dots & 0 \\ 0 & 1 & 1 & 1 & \ddots & \vdots \\ 0 & 0 & 1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & 0 & \dots & \dots & 1 & 1 \end{pmatrix}$$

Приз – займає одну клітинку поля.

### 2.2 Правила гри «Змійка»

Після початку гри і закінчення генерації ігрового поля та його елементів змійка розпочинає рух за вказаним напрямком. Зміна положення змійки відбувається зі сталою частотою (один раз на тік таймера), яка залежить від режиму складності для конкретно взятої ігрової сесії.

При кожному оновленні положення змійки гравець може змінити напрямок її руху. Якщо новий напрямок руху співпадає або протилежний поточному, рух змійки продовжується без зміни напрямку, в іншому випадку

відбувається зміна напрямку руху змійки на вказаний гравцем, ця зміна враховується при подальшому оновленні позиції змійки.

Наступною клітинкою для руху змійки вважається найближча клітинка до голови змійки, яка розміщена за напрямком руху змійки. Під час кожного оновлення змійки вона з'їдає наступну клітинку, якщо та заповнена призом і переміщається на наступну клітинку поля, якщо та пуста.

Якщо в ході ігрового сеансу наступна клітинка знаходиться за межами ігрового поля змійка або перетинає кордон і з'являється симетрично до осі, що паралельна заданому кордону і проходить через центр поля, з протилежного боку ігрового поля, або померає. Поведінку змійки в такій ігровій ситуації визначає задана складність гри.

Якщо змійка з'їдає приз то довжина змійки збільшується на одиницю, кількість очків зароблених в ігровому сеансі збільшується на цінність призу і на полі з'являється новий приз.

Якщо змійка померає, оновлення стану поля і його елементів зупиняється без можливості продовження. Гравець може закінчити ігрову сесію.

Мета гри —набрати якомога більшу кількість ігрових очок.

### 3 ОПИС АЛГОРИТМІВ

Перелік всіх основних змінних та їхнє призначення наведено в таблиці

Таблиця 3.1 – Основні змінні та їхні призначення

Змінна	Призначення
gameDifficulty	Складність гри прочитана з файлу
GameState	Стан поточної ігрової сесії
Field	Зберігає об'єкт класу Field
ScoreCounter	Зберігає об'єкт класу ScoreCounter
Timer	Внутрішній таймер програми, зберігає об'єкт класу DispatcherTimer
Score	Рахунок ігрової сесії
NotifySnakeIsDead	Подія що викликається при зміні значення поля IsAlive змійки на значення false
NotifyFoodIsEaten	Подія що викликається при зміні значення поля IsEaten призу на true
NotifyGameStateChanged	Подія що викликається при зміні GameState
Timer.Tick	Подія що викликається через певний інтервал часу
FieldSnake	Зберігає об'єкт змійки
FieldFood	Зберігає об'єкт їжі
NextCell	Наступна клітинка на шляху змійки
CellType	Тип переліку CellTypes що визначає тип клітинки
SnakeDirection	Змінна типу SnakeDirections зберігаєж поточний напрямок руху змійки
RowCoord	Нова координата рядка

## Продовження таблиці 3.1

ColCoord	Нова координата колонки
----------	-------------------------

## 3.1 Загальний алгоритм

- 1 Зчитати складність ігрової сесії з вхідного файлу
  - 1.1 ЯКЩО файл не пустий встановлюємо gameDifficulty,
  - 1.2 ІНАКШЕ встановлюємо gameDifficulty = Easy
- 2 Згенерувати поле в залежності від складності (3.2) і присвоїти його внутрішній змінній Field
- 3 Згенерувати рахівник очків і присвоїти його внутрішній змінній ScoreCounter
- 4 Згенерувати таймер в залежності від складності і присвоїти внутрішній змінній Timer
- 5 Встановити обробник що встановлює GameState=NotInGame для події Field.Snake.NotifySnakeIsDead
- 6 Встановити обробник що викликає метод додавання очків Рахівника очків для події Field.FieldFood.NotifyFoodIsEaten
- 7 Встановити обробник(3.3) для події NotifyGameStateChanged
- 8 Встановити обробник для події(3.4) Timer.Tick
- 9 КІНЕЦЬ

## 3.2 Алгоритм генерації поля

- 1 ПОЧАТОК
- 2 Встановлення значення внутрішньої змінної difficulty класу Ігрового поля
- 3 Цикл проходження по всіх клітинках поля
  - 3.1 Встановлення їх координат та типу
- 4 Генерація об'єкта змійки і присвоєння його в внутрішню змінну FieldSnake класу Field
- 5 Генерація об'єкта їжі і присвоєння його у внутрішню змінну FieldFood класу Field

## 6 КІНЕЦЬ

### 3.3 Обробка зміни стану гри

#### 1 ПОЧАТОК

#### 2 Перевірка переданого стану гри

##### 2.1 Якщо стан дорівнює значенню переліку GameStates.Paused

###### 2.1.1 Зупинка таймера

##### 2.2 Якщо стан дорівнює значенню переліку GameStates.InGame

###### 2.2.1 Старт таймера

##### 2.3 Якщо стан дорівнює значенню переліку GameStates.NotInGame

###### 2.3.1 Стоп таймера

###### 2.3.2 Збереження статистики( gameDifficulty, поточний час, ScoreCounter.Score)

#### 3 КІНЕЦЬ

### 3.4 Алгоритм дій при виклику події Timer.Tick

#### 1 ПОЧАТОК

#### 2 ЯКЩО GameState==InGame

##### 2.1 Виклик методу для оновлення ігрового поля(3.5)

#### 3 КІНЕЦЬ

### 3.5 Алгоритм оновлення ігрового поля

#### 1 ПОЧАТОК

#### 2 Оновлення положення FieldSnake (3.6)

#### 3 Оновлення положення FieldFood (3.7)

#### 4 КІНЕЦЬ

### 3.6 Алгоритм оновлення змійки

#### 1 ПОЧАТОК

#### 2 Присвоєння NextCell об'єкт наступної клітинки(3.8)

#### 3 ЯКЩО змійка не мертва

3.1 Якщо тип `NextCell.CellType==EmptyCell`

3.1.1 Переміститись в наступну клітинку

3.2 Якщо тип `NextCell.CellType==FoodCell`

3.2.1 З'їсти наступну клітинку

3.3 Якщо тип `NextCell.CellType==SnakeCell`

3.3.1 Змійка мертва

## 1 КІНЕЦЬ

### 3.7 Алгоритм оновлення положення приза

1 ЯКЩО приз з'їдено

1.1 Згенерувати новий приз

1.2 Алгоритм пошуку наступної частини змійки

2 Встановити змінні для позначення зміщення наступної клітинки відносно голови як 0

3 ЯКЩО `SnakeDirection==Up`

3.1 Зміна в рядку на -1

4 ЯКЩО `SnakeDirection==Right`

4.1 Зміна в колонці на 1

5 ЯКЩО `SnakeDirection==Down`

5.1 Зміна в рядку на 1

6 ЯКЩО `SnakeDirection==Left`

6.1 Зміна в Колонці на -1

7 Присвоєння `RowCoord` старого значення координати рядка голови змійки + зміна в рядку

8 Присвоєння `ColCoord` старого значення координати стовпця голови змійки + зміна в стовпчику

9 ЯКЩО `Field.Difficulty==GameDifficulties.Hard` і  
(`RowCoord>Field.FieldSize` або `RowCoord<0` або `ColCoord>FieldSize`  
або `ColCoord<0`)

9.1 Змійка мертва

- 10 ЯКЩО RowCoord $\geq$ 0
  - 10.1 RowCoord=RowCoord%20
- 11 ІНАКШЕ
  - 11.1 RowCoord=FieldSize+RowCoord
- 12 ЯКЩО ColCoord $\geq$ 0
  - 12.1 ColCoord=ColCoord %20
- 13 ІНАКШЕ
  - 13.1 ColCoord=FieldSize+ColCoord
- 14 return Field[RowCoord, ColCoord]
- 15 КІНЕЦЬ

### 3.8 Алгоритм додавання клітинок до змійки

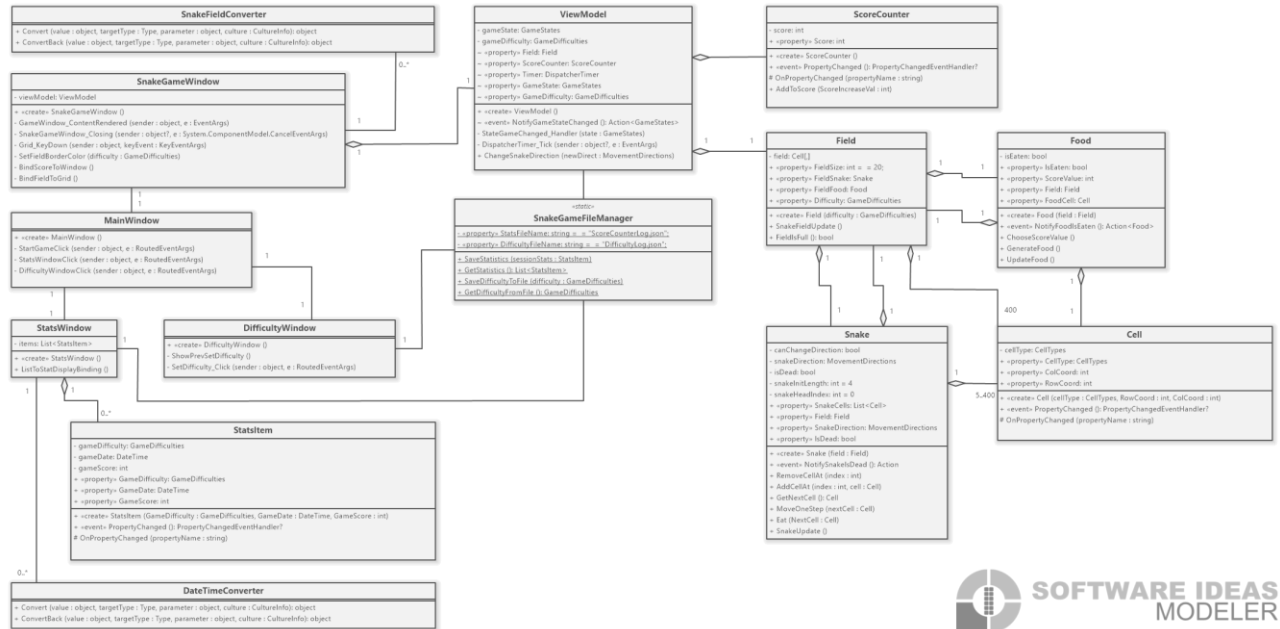
- 1 Якщо місце де додається клітинка це початок або кінець змійки
  - 1.1 ЯКЩО індекс додавання клітинки це голова змійки
    - 1.1.1 Додаємо клітинку в голову змійки
  - 1.2 ІНАКШЕ ЯКЩО індекс додавання клітинки це хвіст змійки
    - 1.2.1 Додаємо клітинку до хвоста змійки
- 2 КІНЕЦЬ



## 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 4.1 Діаграма класів програмного забезпечення

На рисунку 4.1 наведено загальний вигляд діаграми класів.



SOFTWARE IDEAS  
MODELER

Рисунок 4.1 – Діаграма класів

### 4.2 Опис методів частин програмного забезпечення

#### 4.2.1 Стандартні методи

У таблиці 4.1 наведено стандартні методи, використані при розробці програмного забезпечення.

Таблиця 4.1 – Стандартні методи

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
1	List<T>	Add	Adds item to list	Item, T	-
2	List<T>	Count	Gets number of items in list	-	Count, int

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
3	List<T>	Reverse	Reverses the order of elements in list	-	-
4	DateTime	ToString	Converts DateTime to string of given format	Format of outpur string, string	DateTime as string, string
5	File	Open	Opens the filestream	Name of file, string;  Mode in which file should be opened, FileMode	Filestram on the specified path, FileStream
6	StreamReader	ReadLine	Reads a line of file	-	Line, string?
7	StreamWriter	WriteLine	Writes line to file followed by line termaintor	Line, string?	-
8	StreamWriter	Write	Writes line to file	Line, string?	-
9	String	ToLower	Makes string lower case	-	New string, string

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
10	String	Trim	Removes all whitespaces from end and start of string	-	Trimmed string, string
11	JsonSerializer	Serialize<T>	Converts the parameter of generic type into JSON string	Object to be serialized, T	String of JSON, string
12	JsonSerializer	Deserialize< T>	Converts string of JSON format to object of specified generic type parameter	String of JSON, string	Deserialized Object, T
13	Window	ShowDialog	Opens a window and returns only when the newly opened window is closed	-	Indicator of successful window opening, bool

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
14	Panel	Children	Gets panel children	-	Children, UIElementColle ction
15	UIElementCo llection	Add	Adds new element to collection	New element, UIElement	Position of inserted element, int
16	Dependency Object	SetValue	Sets the value of dependencyP roperty of an object	Specified Property, Dependency Property; value to be assigned, object	void
17	FrameWorkE lement	SetBinding	Sets binding for objects property	Property to bind, Dependency property;  Binding to apply, Binding	Conditions of binding, BindingExpressi onBase
18	Binding	Source	Sets binding source	Source, object	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
19	Binding	Path	Sets the path to field which should be binded in the source object	Path, string	-
20	Binding	Converter	Sets the converter for binding	Converter instance, IValueConv erter	-
21	Binding	ConverterPar ameter	Sets the parameter of converter for binding	parameter, object	-
22	Binding	UpdateSourc eTrigger	Sets the value that determines the timing of binding source updates	Value of enumerable type, UpdateSourc eTrigger	-
23	TextBlock	Text	Sets the inner text of the text block	New text, string	-

Продовження таблиці 4.1

№ п/п	Назва класу	Назва функції	Призначенн я функції	Опис вхідних параметрів	Опис вихідних параметрів
24	Window	Owner	Sets the owner of the window	Window owner, Window	-
25	UIElement	Effect	Sets the effect applied to ui element	Effect for element, Effect	-

#### 4.2.2 Користувацькі методи

У таблиці 4.2 наведено користувацькі методи, створені при розробці програмного забезпечення.

Таблиця 4.2 — користувацькі методи

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
1	DateTimeCo nverter	Convert	Convert date to output format	Given date, object; Type of input, Type; parameter of converter, object; Context of culture, CultureInfo	Output date format , string

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
2	DateTimeCo nverter	ConvertBack	-	Given object to convert back, object; Type of input, Type; parameter of converter, object; Context of culture, CultureInfo	-
3	SnakeFieldC onverter	Convert	Convert cell type of cell in field to color that can be displayed on screen	Given color, object; Type of input, Type; parameter of converter, object; Context of culture, CultureInfo	Color , object

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
4	SnakeFieldC onverter	ConvertBack	-	Given object to convert back, object; Type of input, Type; parameter of converter, object; Context of culture, CultureInfo	-
5	ViewModel	ViewModel	Creation of ViewModel instance	-	-
6	ViewModel	Field	Gets the Field of the view model	-	Instance of Field, Field
7	ViewModel	Field	Sets the Field of the viewmodel	Instance of Field, Field	-



Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
8	ViewModel	ScoreCounter	Gets the ScoreCounter instance of the view model	-	Instance of ScoreCounter, ScoreCounter
9	ViewModel	ScoreCounter	Sets the ScoreCounter of the viewmodel	Instance of ScoreCount er, ScoreCount er	-
10	ViewModel	Timer	Gets the Timer instance of the view model	-	Instance of Timer, DispathcherTim er
11	ViewModel	Timer	Sets the Timer of the viewmodel	Instance of Timer, DispatcherT imer	-
12	ViewModel	GameState	Gets the GameState of the view model	-	Value of GameState, GameStates

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
13	ViewModel	GameState	Sets the GameState of the viewmodel	Value of GameState, GameStates	-
14	ViewModel	GameDifficult y	Gets the GameDifficul ty of the view model	-	Value of GameDifficulty, GameDifficultie s
15	ViewModel	GameDifficult y	Sets the GameDifficul ty of the viewmodel	Value of GameDiffic ulty, GameDiffic ulties	-
16	ViewModel	StateGameCh anged_Handle r	Handles the change of GameState	New GameState value, GameStates	-
17	ViewModel	DispatcherTi mer_Tick	Handles the tick of the timer	Sender of event, object; context of event, EventArgs	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
18	ViewModel	ChangeSnake Direction	Changes the direction of the snake	New direction , MovementD irections	-
19	Cell	Cell	Constructor	Type of cell, Celltypes; coordinate of row, RowCoord; coordinate of column, ColCoord	-
20	Cell	CellType	Gets the type of cell	-	Type of current cell, CellTypes
21	Cell	CellType	Sets type of cell	Type of cell, CellTypes	-
22	Cell	ColCoord	Gets column in which cell is located	-	ColCoord in which cell is located, int
23	Cell	ColCoord	Sets column in which cell is located	ColCoord in which cell is located, int	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
24	Cell	RowCoord	Gets row in which cell is located	-	RowCoord in which cell is located, int
25	Cell	RowCoord	Sets row in which cell is located	RowCoord in which cell is located, int	-
26	Cell	OnPropertyCh anged	Notifies subscribed instances that the property has been changed	Name of changed property, string	-
27	Field	Field	Constructor	Difficulty , GameDiffic uties	-
28	Field	FieldSize	Gets the size of the field	-	Size of the field, int
29	Field	FieldSnake	Gets instance of snake that belongs to this field	-	Current Snake, Snake
30	Field	FieldSnake	Sets instance of snake that belongs to this field	New Snake instance, Snake	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
31	Field	FieldFood	Gets instance of food that belongs to this field	-	Current prize instance, Food
32	Field	FieldFood	Sets instance of food that belongs to this field	New Food instance, Food	-
33	Field	Difficulty	Gets the Difficulty of the field	-	Value of Difficulty, GameDifficultie s
34	Field	Difficulty	Sets the Difficulty of the field	Value of Difficulty, GameDiffic ulties	-
35	Field	this	indexer	Row Coordinate of selected cell in field, int; Column Coordinate of selected cell in field, int	Selected Cell from field, Cell

## Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
36	Field	SnakeFieldUp date	Updates field and everything in it	-	-
37	Field	FieldIsFull	Determines if field is full	-	Variable indicating fullness of Field, bool
38	Food	Food	Constructor	Field in which food is generated, Field	-
39	Food	IsEaten	Gets the value of IsEaten	-	Boolean that defines if food is eaten, bool
40	Food	IsEaten	Sets the value of IsEaten	New value of IsEaten flag	-
41	Food	ScoreValue	Gets the value of ScoreValue	-	Value of food in points, int
42	Food	ScoreValue	Sets the value of ScoreValue	New value of food, int	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
43	Food	Field	Gets the instance of Field	-	Inner Field instance of Food, Field
44	Food	Field	Sets the new instance of Field	New Field instance, Field	-
45	Food	FoodCell	Gets the instance of FoodCell	-	Cell in which food is contained, Cell
46	Food	FoodCell	Sets the new instance of FoodCell	New Cell in which food is contained, Cell	-
47	Food	ChooseScore Value	Sets the value of food depending on game difficulty	-	-
48	Food	GenerateFood	Generates fruit in new position on map, if any is available	-	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
49	Food	UpdateFood	Updates position of fruit if previous was eaten	-	-
50	ScoreCounter	ScoreCounter	Constructor	-	-
51	ScoreCounter	Score	Gets the value of inner score variable	-	Score, int
52	ScoreCounter	Score	Sets the value of inner score variable	Score, int	-
53	ScoreCounter	OnPropertyChanged	Notifies subscribed instances that the property has been changed	Name of changed property, string	-
54	ScoreCounter	AddToScore	Adds value to score	Value that should be added, int	-
55	Snake	Snake	Constructor	Instance of field, Field	-



Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
56	Snake	SnakeCells	Returns the list of cells that make the snake up	-	List of snake cells, List<Cell>
57	Snake	SnakeCells	Sets the list of cells that make the snake up	List of snake cells, List<Cell>	-
58	Snake	Field	Returns the inner Field instance of Snake	-	instance of Field in which snake is located, Field
59	Snake	Field	Sets the list of cells that make the snake up	New instance of Field in which snake is located, Field	-
60	Snake	SnakeDirection	Returns the snake direction	-	Direction, SnakeDirections
61	Snake	SnakeDirection	Sets the snake direction	New Direction of snake, SnakeDirections	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
62	Snake	IsDead	Returns the indicator of snake`s death	-	Indicator of snake`s death, bool
63	Snake	IsDead	Sets the indicator of snake`s death	Indicator of snake`s death, bool	
64	Snake	RemoveCellAt	Removes one of snake`s sells (on given index)	Index on which we want to remove cell in snake, int	-
65	Snake	AddCellAt	Adds cell to snake at given index	Index at which we want to add cell to snake, int	-
66	Snake	GetNextCell	Returns the next cell on the snake`s route	-	Next cell to which snake is going to travel, Cell
67	Snake	MoveOneStep	Moves snake one step to the specified cell	Next cell to which snake is going to travel, Cell	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
68	Snake	Eat	Eats the next cell on the snake route	Next cell to which snake is going to travel, Cell	-
69	Snake	SnakeUpdate	Updates the position and the state of the snake on the field	-	-
70	SnakeGame FileManager	SaneStatistics	Save statistics to the correspondin g file	Statistics of the current session	-
71	SnakeGame FileManager	GetStatistics	Get all statistics history from correspondin g file	-	Stats history, List<StatsItem>
72	SnakeGame FileManager	SnakeGameFi leManager	Saves difficulty level to correspondin g file	Difficulty level, GameDiffic ulties	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
73	SnakeGame FileManager	GetDifficulty FromFile	Gets difficulty from correspondin g file	-	Difficulty level, GameDifficultie s
74	StatsItem	StatsItem	Constructor	Game difficulty, GameDiffic ulties; date of game, DateTime; score of the game, int	-
75	StatsItem	GameDifficult y	Gets the value of inner difficulty field	-	Difficulty, GameDifficultie s
76	StatsItem	GameDifficult y	Sets the value of inner difficulty field	New Difficulty, GameDiffic ulties	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
77	StatsItem	GameDate	Gets the instance of inner date field	-	Date, DateTime
78	StatsItem	GameDate	Sets the instance of inner date field	New Difficulty, DateTime	-
79	StatsItem	GameScore	Gets the value of Score inner field	-	Score, int
80	StatsItem	GameScore	Sets the value of inner Score field	New Score, int	-
81	ScoreCounter	OnPropertyChanged	Notifies subscribed instances that the property has been changed	Name of changed property, string	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
82	DifficultyWi ndow	ShowPrevSet Difficulty	Reads previously set difficulty from file and displays it in the window	-	-
83	DifficultyWi ndow	SetDifficulty_ Click	Processes the user choice in difficulty window	Sender, object; Context in which handler was invoked, RoutedEven tArgs	-
84	DifficultyWi ndow	DifficultyWin dow	Constructor	-	-
85	MainWindo w	MainWindow	Constructor	-	-
86	MainWindo w	StartGameCli ck	Handler for click of the “start game” button, opens SnakeGame Window	Sender, object; Useful options for handler handler, RoutedEven tArgs	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
87	MainWindow	StatsWindow_Click	Handler for click of the “statistics” button, opens StatsWindow	Sender, object; Useful options for handler handler, RoutedEventArgs	-
88	MainWindow	DifficultyWindow_Click	Handler for click of the “Difficulty” button, opens DifficultyWindow	Sender, object; Useful options for handler handler, RoutedEventArgs	-
89	SnakeGameWindow	SnakeGameWindow	Constructor	-	-
90	SnakeGameWindow	GameWindow_ContentRendered	Calls methods that must be called as soon as window is rendered	Sender, object; Useful options for handler handler, RoutedEventArgs	-

Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
91	SnakeGame Window	SnakeGameW indow_Closin g	Changes the state of the game to “NotInGame” when window is closed, if it was in other state before	Sender, object; Useful options for handler handler, RoutedEven tArgs	-
92	SnakeGame Window	Grid_KeyDo wn	Handles the event of key being pressed in snake window	Sender, object; Useful options for handler handler, RoutedEven tArgs	-
93	SnakeGame Window	SetFieldBorde rColor	Sets the color of field border depending on difficulty	Difficulty of game session, GameDiffic ulties	-
94	SnakeGame Window	BindScoreTo Window	Binds score of viewmodel to its display on the screen	-	-



Продовження таблиці 4.2

№ п/п	Назва класу	Назва функції	Призначення функції	Опис вхідних параметрів	Опис вихідних параметрів
95	SnakeGame Window	BindFieldToGrid	Binds every cell of viewmodel's field to its display on the screen	-	-
96	StatsWindow	StatsWindow	Constructor, initializes window and everything in it	-	-
97	StatsWindow	ListToStatDisplayBinding	Binds the list of StatsItem to the display list in the window	-	-

## 5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

### 5.1 План тестування

Складемо план тестування програмного забезпечення, за допомогою якого протестуємо весь основний функціонал та реакцію на виключні ситуації

- а) Тестування головного меню
  - 1) Тестування підсвічування кнопок
  - 2) Тестування функцій, прив'язаних, до кнопок при натисканні на них
  - 3) Тестування доступу до головного меню при відкритих вікнах гри, складності та статистики
- б) Тестування меню складності
  - 1) Тестування переключення станів радіо-кнопок при натисканні
  - 2) Тестування збереження складності при натисканні кнопки «встановити»
  - 3) Тестування не збереження змін складності при натисканні кнопки «відміна»
- в) Тестування меню статистики
  - 1) Тестування прокрутки статистики при кількості зіграних ігор від 0 до 7
  - 2) Тестування прокрутки статистики при 7 і більше зіграних ігрових сесіях
- г) Тестування процесу гри
  - 1) Тестування правильного початкового положення змійки та приза на полі
  - 2) Тестування правильного руху змійки через кожен конкретно визначений період часу
  - 3) Тестування правильного руху змійки в залежності від натиснутих користувачем клавіш
  - 4) Тестування механіки тимчасової зупинки і продовження гри
  - 5) Тестування механіки з'їдання призу
  - 6) Тестування механіки смерті змійки

Проведемо тестування( таблиці 5.1 – 5.12 )

## 5.2 Приклади тестування

Таблиця 5.1 - Тестування підсвічування кнопок

Мета тесту	Перевірити підсвічуваність кнопок головного меню
Початковий стан програми	Відкрите вікно програми
Вхідні дані	-
Схема проведення тесту	Навести курсор на кнопки
Очікуваний результат	Підсвічення кнопки, на яку навели
Стан програми після проведення випробувань	Поточна кнопка підсвічується

Таблиця 5.2 - Тестування функцій, прив'язаних, до кнопок при натисканні на них

Мета тесту	Перевірити функцію, прив'язану до кнопки, при її натисканні
Початковий стан програми	Відкрите вікно програми
Вхідні дані	-
Схема проведення тесту	Натиснути на кнопки
Очікуваний результат	Перехід до меню складності при натисканні “difficulty”, перехід до гри при “play”, перехід до статистики при натисканні “statistics”
Стан програми після проведення випробувань	Перейшло до меню складності, статистики, та гри при натисканні відповідних кнопок

Таблиця 5.3 - Тестування доступу до головного меню при відкритих вікнах гри, складності та статистики

Мета тесту	Перевірити неможливість внесення змін в налаштування сесії та неможливість нагромадження вікон
Початковий стан програми	Відкрите вікно налаштування
Вхідні дані	-
Схема проведення тесту	Натиснути кнопку грати в головному вікні
Очікуваний результат	Відсутність реакції програми

## Продовження таблиці 5,3

Стан програми після проведення випробувань	Основне вікно не реагує на натискання і доступ до нього відновлюється лише після закриття усіх інших вікон
--	--

Таблиця 5.4 - Тестування переключення станів радіо-кнопок при натисканні

Мета тесту	Перевірити функціонування переключення радіокнопок при натисканні
Початковий стан програми	Відкрите вікно складності
Вхідні дані	-
Схема проведення тесту	Настинути на кнопки-тумблери та подивитися, чи вони перемикаються
Очікуваний результат	При натисканні кнопки, що позначає певний рівень складності, вона помічається як відмічена.
Стан програми після проведення випробувань	Вибрана кнопка відмічена як вибрана, вікно складності не закрито.

Таблиця 5.5 - Тестування збереження складності при натисканні кнопки «встановити»

Мета тесту	Перевірити зберігання вибраного рівня складності після натискання кнопки «встановити»
Початковий стан програми	Відкрите вікно складності
Вхідні дані	-
Схема проведення тесту	Вибрати новий режим складності і натиснути кнопку “set”
Очікуваний результат	Після початку наступної ігрової сесії відображатиметься вибрана складність. При повторному вході в меню складності відображатиметься нова ігрова складність як попередньо вибрана
Стан програми після проведення випробувань	Складність збережена і відображається правильно в усіх вікнах

Таблиця 5.6 - Тестування не збереження змін складності при натисканні кнопки «відміна»

Мета тесту	Перевірити не зберігання вибраного рівня складності після натискання кнопки «відміна»
Початковий стан програми	Відкрите вікно складності
Вхідні дані	-
Схема проведення тесту	Вибрати новий режим складності і натиснути кнопку “cancel”
Очікуваний результат	Після початку наступної ігрової сесії відображатиметься стара складність. При повторному вході в меню складності відображатиметься стара ігрова складність як попередньо вибрана
Стан програми після проведення випробувань	Складність не встановлена і відображається коректно в усіх ігрових вікнах

Таблиця 5.7 - тестування прокрутки статистики при кількості зіграних ігор від 0 до 7

Мета тесту	Перевірити функціонування прокрутки статистики
Початковий стан програми	Відкрите вікно статистики і файл статистики попередньо стертий
Вхідні дані	-
Схема проведення тесту	Спроба прокрутки статистики
Очікуваний результат	Файл пустий отже в списку нічого не відображається, повзунок для прокрутки відсутній
Стан програми після проведення випробувань	Вікно статистики може бути закритим, після чого гравець може продовжити гру

Таблиця 5.8 - Тестування прокрутки статистики при 7 і більше зіграних ігрових сесіях

Мета тесту	Перевірити функціонування прокрутки статистики
Початковий стан програми	Відкрите вікно статистики, зіграно більше шести ігор
Вхідні дані	-
Схема проведення тесту	Проскролити вікно статистики

Продовження таблиці 5.8

Очікуваний результат	При скролі вдається побачити всю історію попередньо зіграних сесій
Стан програми після проведення випробувань	Вікно статистики залишається відкритим. Гравець може його закрити і продовжити гру

Таблиця 5.9 - Тестування правильного початкового положення змійки та приза на полі

Мета тесту	Перевірити правильність початкової розстановки об'єктів на ігровому полі
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру
Очікуваний результат	Початкове положення змійки буде горизонтальним і голова буде направлена в ліву сторону, початковий напрямок руху змійки вліво. Початкове положення приза- випадкова клітинка поля
Стан програми після проведення випробувань	Гра не зупиняється, положення об'єктів на полі оновлюється

Таблиця 5.10 - Тестування правильного руху змійки через кожен конкретно визначений період часу

Мета тесту	Перевірити правильність руху клітинок змійки
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру

Продовження таблиці 5.10

Очікуваний результат	Голова переміщається в заадному напрямку руху змійки, кожна з клітинок змійки переміщується на місце сусідньої клітинки що стоїть ближче до голови, розмір змійки не змінюється. Переміщення відбувається з частотою що відповідає параметру складності для конкретної ігрової сесії.
Стан програми після проведення випробувань	Гра не зупиняється, положення об'єктів на полі оновлюється

Таблиця 5.11 - Тестування правильного руху змійки в залежності від натиснутих користувачем клавіш

Мета тесту	Перевірити правильність руху клітинок змійки в залежності від натиснутих користувачем клавіш
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру
Очікуваний результат	Голова переміщається в заданому користувачем з клавіатури напрямку руху , кожна з клітинок змійки переміщується на місце сусідньої клітинки що стоїть ближче до голови, розмір змійки не змінюється. Переміщення відбувається з частотою що відповідає параметру складності для конкретної ігрової сесії.
Стан програми після проведення випробувань	Гра не зупиняється, положення об'єктів на полі оновлюється

Таблиця 5.12 - Тестування механіки тимчасової зупинки і продовження гри

Мета тесту	Перевірити правильність реакції гри на натискання клавіш для призупинення і продовження гри
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру
Очікуваний результат	При натисканні клавіші пробіл оновлення об'єктів на полі призупиняється, якщо гра до цього знаходилася в активному стані та поновлюється якщо гра була призупинена.
Стан програми після проведення випробувань	Гра продовжується або залишається в призупиненому стані

Таблиця 5.13 - Тестування механіки з'їдання призу

Мета тесту	Перевірити механіку з'їдання призу
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру. З'їсти приз
Очікуваний результат	Старий приз зникає, змійка збільшує свою довжину на 1, до рахунку очків додається вартість приза, на полі з'являється новий приз
Стан програми після проведення випробувань	Гра не зупиняється, положення об'єктів на полі оновлюється

Таблиця 5.14 - Тестування механіки смерті змійки

Мета тесту	Перевірити алгоритм дій при смерті змійки
Початковий стан програми	Відкрите вікно програми і розпочата гра
Вхідні дані	-
Схема проведення тесту	Грати в гру. Померти



Продовження таблиці 5.14

Очікуваний результат	Оновлення поля зупиняється, управління змійкою не можливе, кількість очків не змінюється гравець може закрити вікно
Стан програми після проведення випробувань	Графець може перейти в головне меню закривши вікно гри

## 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

### 6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe, відкривається головне вікно програми.

На рисунку 6.1 наведено зображення головного вікна програми

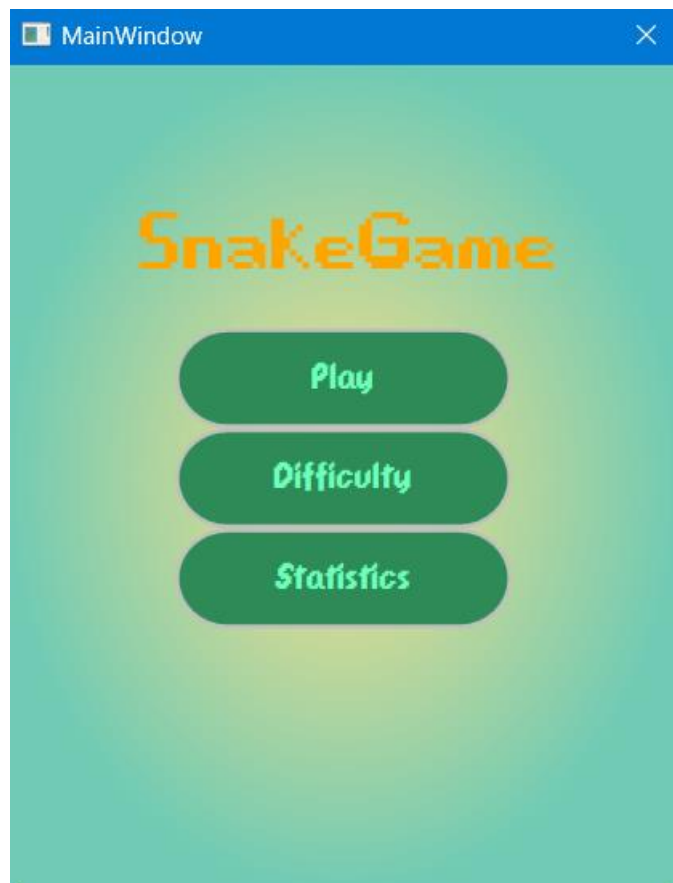


Рисунок 6.1 – Головне вікно програми

Далі у вкладці “difficulty” можемо вибрати складність гри(рисунок 6.2).

На рисунку 6.2 наведено зображення вікна складності програми.

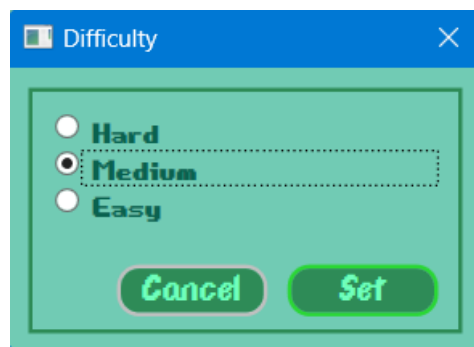


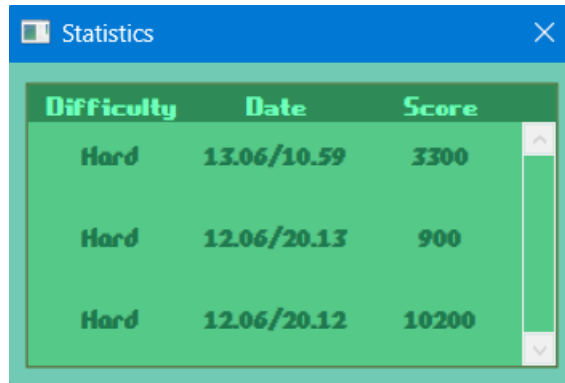
Рисунок 6.2 – Вибір складності

Після цього натискаємо або Set або Cancel або закриваємо вікно якщо ми

передумали встановлювати складність

Після цього можемо перейти в меню статистики натиснувши кнопку “Statistics” щоб подивитись історію попередніх ігрових сесій(рисунок 6.3).

На рисунку 6.3 наведено зображення вікна статистики програми.



The screenshot shows a window titled "Statistics" with a table of game sessions. The table has three columns: Difficulty, Date, and Score. There are three rows of data, all with a difficulty of "Hard".

Difficulty	Date	Score
Hard	13.06/10.59	3300
Hard	12.06/20.13	900
Hard	12.06/20.12	10200

6.3 – перегляд статистики

Після цього можемо закрити меню статистики і перейти до гри натиснувши кнопку “Play” (рисунок 6.4).

На рисунку 6.4 наведено зображення вікна гри програми.

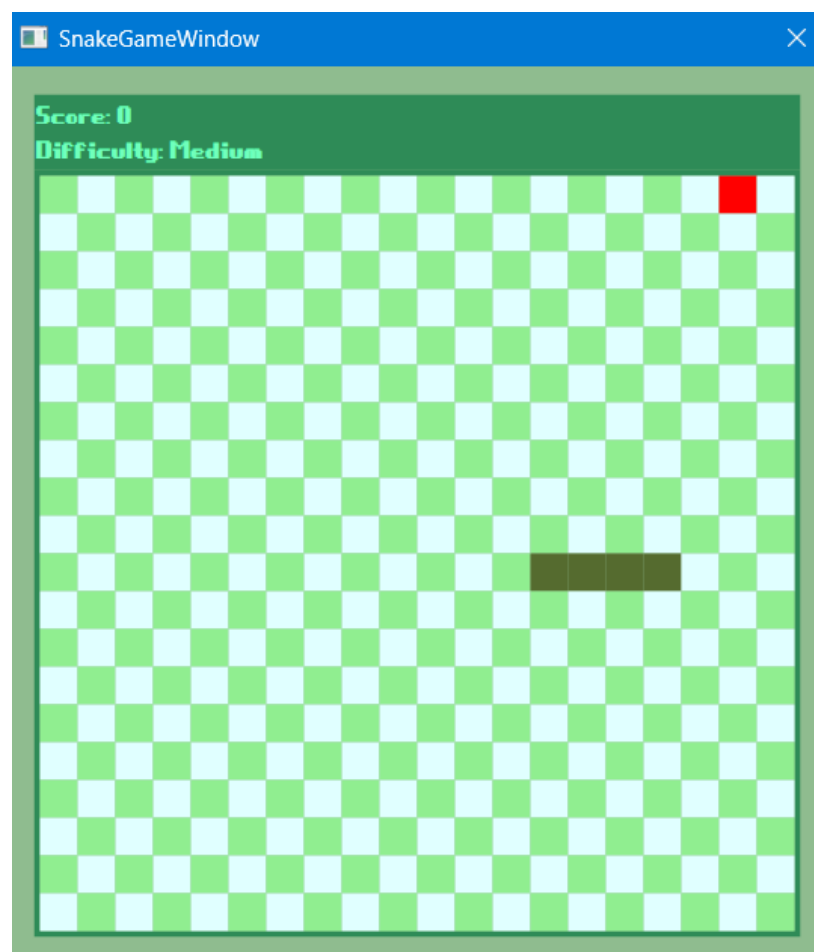


Рисунок 6.4 – Почати гру

Після закінчення гри можемо закрити віно змійки і перейти назад до головного меню. Кроки на рисунках 6.4, 6.3, 6.2 можуть бути виконані ц будь якій послідовності при запуску програми.

## 6.2 Системні вимоги

Системні вимоги до програмного забезпечення наведені в таблиці 10.1.

Таблиця 10.1 – Системні вимоги програмного забезпечення

	Мінімальні	Рекомендовані
Операційна система	Windows® XP/Windows Vista/Windows 7/ Windows 8/Windows 10/Windows 11 (з останніми оновленнями)	Windows 7/ Windows 8/Windows 10/Windows 11  (з останніми оновленнями)
Процесор	Intel® Pentium® III 1.0 GHz або AMD Athlon™ 1.0 GHz	Intel® Pentium® D або AMD Athlon™ 64 X2
Оперативна пам'ять	256 MB RAM (для Windows® XP) / 1 GB RAM (для Windows Vista/Windows 7/  Windows 8/Windows 10)	2 GB RAM
Відеоадаптер	Intel GMA 950 з відеопам'яттю об'ємом не менше 64 МБ (або сумісний аналог)	
Дисплей	800x600	1024x768 або краще

## ВИСНОВОК

Під час курсової роботи було вивчено метод розробки програмного забезпечення з використанням ООП і патерну проєктування ПЗ MVVM на прикладі гри «Змійка».

В розділі «Постановка задачі» були сформульовані основні вимоги до функціоналу вікон програми а також описані ігрові механіки, що потребують реалізації в застосунку

В розділі «Теоретичні відомості» були наведені роз'яснення, що пояснюють правила гри і можливі алгоритми дій гравця під час гри ; описано алгоритм за яким здійснюється ігровий процес, проводиться аналіз математичної моделі змійки та розглядаються різні ігрові ситуації;

У розділі «Опис алгоритмів» наведено опис основних алгоритмів програми таких як оновлення ігрового поля, рух змійки та поїдання призу.

В четвертому розділі записки наведено діаграму класів що наочно відображає реалізацію патернів застосованих при розробці ПЗ. Також в другому підрозділі задано короткий опис функцій що були використані для реалізації вище наведених алгоритмів.

В п'ятому розділі проведено тестування основних механік ігрового процесу та механізмів програми пов'язаних з графічним інтерфейсом та роботою з файлами

Шостий розділ призначений для полегшення ознайомлення користувача з графічним інтерфейсом додатку та надає інформацію про можливі варіанти взаємодії з програмним забезпеченням.

В ході виконання курсової роботи мною було опановано навички проєктування ПЗ в об'єктно орієнтованому стилі та розробки застосунків що використовують графічний інтерфейс для взаємодії з користувачем.

## ПЕРЕЛІК ПОСИЛАНЬ

- 1 Гра змійка: [https://en.wikipedia.org/wiki/Snake\\_\(video\\_game\\_genre\)](https://en.wikipedia.org/wiki/Snake_(video_game_genre))
- 2 Шрифти для головного меню та вкладок програми: <https://www.dafont.com/>
- 3 Корисні матеріали: <https://habr.com/ru/company/geekbrains/blog/268741/>

**ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ**

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра  
інформатики та програмної інженерії

Затвердив

Керівник \_\_\_\_\_

«\_\_\_» \_\_\_\_\_ 201\_ р.

Виконавець:

Студент: Лошак Віктор Іванович

«\_\_\_» \_\_\_\_\_ 201\_ р.

**ТЕХНІЧНЕ ЗАВДАННЯ**

на виконання курсової роботи

на тему: «Створення ігрового застосунку «Змійка»»

з дисципліни:

«Основи програмування»

Київ 2022

1. *Мета:* Метою курсової роботи є розробка ігрового застосунку «Змійка»
2. *Дата початку роботи:* «\_02\_» \_травня\_ 2022 р.
3. *Дата закінчення роботи:* «\_\_\_» \_\_\_\_\_ 202\_ р.
4. *Вимоги до програмного забезпечення.*

1) Функціональні вимоги:

- Можливість відображати на екрані елементи гри «Змійка» після запуску ігрового процесу
- Можливість запуску ігрового процесу
- Можливість переключатися між вкладками головного меню(статистика, режим складності)
- Можливість вибору режиму складності в головному меню
- Можливість збереження і оновлення статистики після закінчення ігрової сесії
- Можливість змінювати напрямок руху змійки за допомогою натискання відповідних клавіш на клавіатурі(w, a, s, d)
- Можливість «з'їдати» фрукти на ігровому полі
- Можливість збільшувати довжину змійки після «з'їдання» фруктів
- Можливість призупинення гри зі збереженням поточного стану ігрового поля

2) Нефункціональні вимоги:

- Можливість запуску додатку на платформі зі встановленою версією Windows (XP/Vista SP2/ 7 SPI/ 8 / 8.1 /10/ 11) версією .Net Framework не нижче 3.5
- Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:  
ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.  
ГОСТ 19.106 - 78 - Вимоги до програмної документації.



ГОСТ 7.1 - 84 та ДСТУ 3008 - 2015 - Розробка технічної документації.

*5. Стадії та етапи розробки:*

- 1) Об'єктно-орієнтований аналіз предметної області задачі (до \_\_.\_\_.202\_\_р.)
- 2) Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.202\_\_р.)
- 3) Розробка програмного забезпечення (до \_\_.\_\_.202\_\_р.)
- 4) Тестування розробленої програми (до \_\_.\_\_.202\_\_р.)
- 5) Розробка пояснювальної записки (до \_\_.\_\_.202\_\_р.).
- 6) Захист курсової роботи (до \_\_.\_\_.202\_\_р.).

*6. Порядок контролю та приймання.* Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

## ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

*Тексти програмного коду «SnakeGame»*

(Найменування програми(документа))

---

*Електронний носій*

(Вид носія даних)

---

*39 арк, 146 Кб*

(Обсяг програми (документа), арк., Кб)

*студента групи ПП-11 І курсу*

*Лошака В.І,*

**DateTimeConverter.cs**

```

using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Data;

namespace SnakeGame
{
    internal class DateTimeConverter : IValueConverter
    {
        #region Fields
        #endregion

        #region Constructors
        #endregion

        #region Methods

        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            DateTime dateTime = (DateTime)value;
            return dateTime.ToString("dd.MM/HH.mm");
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            throw new NotImplementedException();
        }
        #endregion
    }
}

```

**SnakeFieldConverter.cs**

```

using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Data;
using System.Windows.Media;

namespace SnakeGame
{
    internal class SnakeFieldConverter : IValueConverter
    {
        #region Methods
        public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            if (value == null) throw new ArgumentNullException();
            Tuple<int, int> coords = parameter as Tuple<int, int>;
            switch (value)
            {
                case CellTypes.EmptyCell:
                {
                    if ((coords.Item1+coords.Item2) % 2 == 0)
                        return Brushes.LightGreen/*YellowGreen*/;
                    else if ((coords.Item1 + coords.Item2) % 2 == 1)
                        return Brushes.LightCyan/*MediumSeaGreen*/;
                    else
                        return Brushes.AliceBlue;
                }
                case CellTypes.FoodCell:
                    return Brushes.Red;
                case CellTypes.SnakeCell:
                    return Brushes.DarkOliveGreen;
                default: return DependencyProperty.UnsetValue;
            }
        }

        public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
        {
            return DependencyProperty.UnsetValue;
        }
    }
}

```

```
    }  
    #endregion  
  }  
}
```

**Cell.cs**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class Cell : INotifyPropertyChanged
    {
        #region Fields
        private CellTypes cellType;
        #endregion

        #region Constructors
        public Cell(CellTypes cellType, int RowCoord, int ColCoord)
        {
            CellType = cellType;
            this.RowCoord = RowCoord;
            this.ColCoord = ColCoord;
        }
        #endregion

        #region Properties
        public CellTypes CellType
        {
            get => cellType;
            set
            {
                cellType = value;
                OnPropertyChanged(nameof(CellType));
            }
        }

        public int ColCoord { get; set; }
        public int RowCoord { get; set; }
        #endregion

        #region Events
        public event PropertyChangedEventHandler? PropertyChanged;
        #endregion

        #region Handlers
        protected void OnPropertyChanged(string propertyName)

```

```
{
    PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
}
#endregion

#region Methods
#endregion
}
enum CellTypes
{
    EmptyCell,
    FoodCell,
    SnakeCell,
}
}
```

**Field.cs**

```

using SnakeGame.f_ViewModel;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class Field
    {
        #region Fields
        private Cell[,] field;
        #endregion

        #region Constructors
        public Field(GameDifficulties difficulty)
        {
            //setting difficulty
            Difficulty = difficulty;
            //emptying field
            field = new Cell[FieldSize, FieldSize];
            for (int i = 0; i < FieldSize; i++)
            {
                for (int j = 0; j < FieldSize; j++)
                {
                    field[i, j] = new Cell(CellTypes.EmptyCell, i, j);
                }
            }
            //initializing snake
            FieldSnake = new Snake(this);
            //initializing food
            FieldFood = new Food(this);
        }
        #endregion

        #region Properties
        public int FieldSize { get; } = 20;
        public Snake FieldSnake { get; set; }
        public Food FieldFood { get; set; }
        public GameDifficulties Difficulty { get; set; }
        #endregion
    }
}

```



```

#region Methods
public Cell this[int rowInd, int colInd]
{
    get { return field[rowInd, colInd]; }
}

public void SnakeFieldUpdate()
{
    FieldSnake.SnakeUpdate();
    FieldFood.UpdateFood();
}

public bool FieldIsFull()
{
    bool result= true;
    for (int i = 0; i < FieldSize; i++)
    {
        for (int j = 0; j < FieldSize; j++)
        {
            if (field[i, j].CellType == CellTypes.EmptyCell)
                result = false;
        }
    }
    return result;
}
//public void ResetField()
//{
//    for (int i = 0; i < FieldSize; i++)
//    {
//        for (int j = 0; j < FieldSize; j++)
//        {
//            Field[i, j].CellType = Cell.CellTypes.EmptyCell;
//        }
//    }
//    //initializing snake
//    MySnake = new Snake(this);
//    //initializing food
//    MyFood = new Food(this);
//}
#endregion

}

enum GameDifficulties
{
    Easy,

```

Medium,  
Hard,  
}  
  
}

**Food.cs**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class Food
    {
        #region Fields
        private bool isEaten;
        #endregion

        #region Constructors
        public Food(Field field)
        {
            Field = field;
            ChooseScoreValue();
            GenerateFood();
        }
        #endregion

        #region Properties
        public bool IsEaten
        {
            get => isEaten;
            set
            {
                isEaten = value;
                if (IsEaten)
                    NotifyFoodIsEaten?.Invoke(this);
            }
        }
        public int ScoreValue { get; set; }
        public Field Field { get; init; }
        public Cell FoodCell { get; set; }
        #endregion

        #region Events
        public event Action<Food> NotifyFoodIsEaten;
        #endregion

        #region Methods
        public void ChooseScoreValue()

```

```

{
    if (Field.Difficulty == GameDifficulties.Easy)
        ScoreValue = 100;
    else if (Field.Difficulty == GameDifficulties.Medium)
        ScoreValue = 200;
    else if (Field.Difficulty == GameDifficulties.Hard)
        ScoreValue = 300;
}
public void GenerateFood()
{
    if (!Field.FieldIsFull())
    {
        Random random = new Random();
        Cell RandomCellOnField;

        do
        {
            RandomCellOnField = Field[random.Next(0, Field.FieldSize),
random.Next(0, Field.FieldSize)];
        } while (RandomCellOnField.CellType != CellTypes.EmptyCell);
        FoodCell = RandomCellOnField;
        FoodCell.CellType = CellTypes.FoodCell;
        IsEaten = false;
    }
}
public void UpdateFood()
{
    if (IsEaten)
    {
        GenerateFood();
    }
}
#endregion
}
}

```

**ScoreCounter.cs**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class ScoreCounter : INotifyPropertyChanged
    {
        #region Fields
        private int score;
        #endregion

        #region Constructors
        public ScoreCounter()
        {
            Score = 0;
        }
        #endregion

        #region Properties
        public int Score
        {
            get => score; private set
            {
                score = value;
                OnPropertyChanged(nameof(Score));
            }
        }
        #endregion

        #region Events
        public event PropertyChangedEventHandler? PropertyChanged;
        #endregion

        #region Handlers
        protected void OnPropertyChanged(string propertyName)
        {
            PropertyChanged?.Invoke(this,
Property
            PropertyChangedEventArgs(propertyName));
        }
        #endregion
    }
}

```

new

```
#region Methods
public void AddToScore(int ScoreIncreaseVal)
{
    Score += ScoreIncreaseVal;
}
#endregion
}
}
```

**Snake.cs**

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class Snake
    {
        #region Fields
        private bool canChangeDirection; //fixes bug of changing direction twice per
update        private MovementDirections snakeDirection;
        private bool isDead;
        private int snakeInitLength = 4;
        private const int snakeHeadIndex = 0;
        #endregion

        #region Constructors
        public Snake(Field field)
        {
            //reference to the field we create snake in
            Field = field;
            if (field.FieldSize < snakeInitLength)
            {
                throw new ArgumentException("Field is too samll. Snake can't fit in
it");
            }

            //initializing snake in the middle of the map
            SnakeCells = new List<Cell>();
            for (int i = 0; i < snakeInitLength; i++)
            {
                AddCellAt(SnakeCells.Count, Field[Field.FieldSize / 2, Field.FieldSize
/ 2 + i]);
            }
            //initializing snake direction
            canChangeDirection = true;
            SnakeDirection = MovementDirections.Left;
            //initializing other fields
            IsDead = false;
        }
        #endregion

        #region Properties

```

```

public List<Cell> SnakeCells { get; set; }
public Field Field { get; init; }
public MovementDirections SnakeDirection //needs refactoring to make logic
more transparent
{
    get { return snakeDirection; }
    set
    {
        if (Convert.ToBoolean(((int)value + (int)SnakeDirection) % 2) &&
canChangeDirection)
        {
            snakeDirection = value;
            canChangeDirection = false;
        }
    }
}
public bool IsDead
{
    get => isDead;
    set
    {
        isDead = value;
        if (value == true)
            NotifySnakeIsDead?.Invoke();
    }
}
#endregion

#region Events
public event Action NotifySnakeIsDead;
#endregion

#region Methods
public void RemoveCellAt(int index)
{
    SnakeCells[index].CellType = CellTypes.EmptyCell;
    SnakeCells.RemoveAt(index);
}

public void AddCellAt(int index, Cell cell)
{
    int SnakeTailIndex = SnakeCells.Count;
    if (index != snakeHeadIndex && index != SnakeTailIndex)
        throw new Exception("Can't add cell to the middle of the snake");

    if (index == snakeHeadIndex)

```



```

        SnakeCells.Insert(snakeHeadIndex, cell);
    else if (index == SnakeTailIndex)
        SnakeCells.Insert(SnakeTailIndex, cell);
    cell.CellType = CellTypes.SnakeCell;
}

public Cell GetNextCell()
{
    int deltaRow = 0, deltaCol = 0;
    switch (SnakeDirection)
    {
        case MovementDirections.Up:
            deltaRow = -1;
            break;
        case MovementDirections.Right:
            deltaCol = 1;
            break;
        case MovementDirections.Down:
            deltaRow = 1;
            break;
        case MovementDirections.Left:
            deltaCol = -1;
            break;
    }
    //refactoring needed here
    //the trick is that we divide final coordinates by twenty to take into account
    //the fact that we can cross the border and coordinated will drop
    int RowCoord = SnakeCells[snakeHeadIndex].RowCoord + deltaRow;
    int ColCoord = SnakeCells[snakeHeadIndex].ColCoord + deltaCol;

    //this conditional doesn't need to be here(nEEDED refactoring)
    if (Field.Difficulty == GameDifficulties.Hard &&
        (RowCoord > Field.FieldSize-1 || RowCoord < 0 || ColCoord >
        Field.FieldSize-1 || ColCoord < 0))
    {
        IsDead = true;
    }

    RowCoord = RowCoord >= 0 ? (RowCoord % 20) : (RowCoord +
    Field.FieldSize);
    ColCoord = ColCoord >= 0 ? (ColCoord % 20) : (ColCoord +
    Field.FieldSize);

    return Field[RowCoord, ColCoord];
}

```

```

public void MoveOneStep(Cell nextCell)
{
    RemoveCellAt(SnakeCells.Count - 1);
    AddCellAt(snakeHeadIndex, nextCell);
}

public void Eat(Cell NextCell)
{
    Cell GrowingCell = SnakeCells[SnakeCells.Count - 1];
    MoveOneStep(NextCell);
    AddCellAt(SnakeCells.Count, GrowingCell);
    Field.FieldFood.IsEaten = true;
}

public void SnakeUpdate()
{
    Cell NextCell = GetNextCell();
    if (!IsDead)
    {
        switch (NextCell.CellType)
        {
            case CellTypes.EmptyCell:
                MoveOneStep(NextCell);
                break;
            case CellTypes.FoodCell:
                Eat(NextCell);
                break;
            case CellTypes.SnakeCell:
                IsDead = true;
                break;
        }
    }
    canChangeDirection = true;
}

#endregion
}

public enum MovementDirections
{
    Left = 0,
    Up = 1,
    Right = 2,
    Down = 3,
}
}

```

**SnakeGameFileManager.cs**

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.Json;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal static class SnakeGameFileManager
    {
        #region Fields
        private static string StatsFileName { get; set; } = "ScoreCounterLog.json";
        private static string DifficultyFileName { get; set; } = "DifficultyLog.json";
        #endregion

        #region Constructors
        #endregion

        #region Methods

        public static void SaveStatistics(StatsItem sessionStats)
        {
            using (StreamWriter writer = new StreamWriter(File.Open(StatsFileName,
FileMode.Append)))
            {
                writer.WriteLine(JsonSerializer.Serialize(sessionStats));
            }
        }

        public static List<StatsItem> GetStatistics()
        {
            List<StatsItem> statistics = new List<StatsItem>();
            using (StreamReader reader = new
StreamReader(File.Open(StatsFileName, FileMode.Open)))
            {
                while (!reader.EndOfStream)
                {
                    string readItem = reader.ReadLine() ?? string.Empty;
                    if (readItem != string.Empty)
                        statistics.Add(JsonSerializer.Deserialize<StatsItem>(readItem));
                }
            }
        }
    }
}

```

```

        statistics.Reverse();
        return statistics;
    }

    public static void SaveDifficultyToFile(GameDifficulties difficulty)
    {
        using (StreamWriter writer = new
StreamWriter(File.Open(DifficultyFileName, FileMode.Create)))
        {
            writer.Write(JsonSerializer.Serialize(difficulty));
        }
    }

    public static GameDifficulties GetDifficultyFromFile()
    {
        GameDifficulties difficulty;
        using (StreamReader reader = new
StreamReader(File.Open(DifficultyFileName, FileMode.Open)))
        {
            string jsonDiff = reader.ReadLine();
            if (jsonDiff != null)
                difficulty = JsonSerializer.Deserialize<GameDifficulties>(jsonDiff);
            else
                throw new Exception("Difficulty file is empty.");
        }
        return difficulty;
    }
}
#endregion
}

```

**StatsItem.cs**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Text.Json.Serialization;
using System.Threading.Tasks;

namespace SnakeGame.Model
{
    internal class StatsItem:INotifyPropertyChanged
    {
        #region Fields
        private GameDifficulties gameDifficulty;
        private DateTime gameDate;
        private int gameScore;
        #endregion

        #region Constructors

        public StatsItem(GameDifficulties GameDifficulty, DateTime GameDate, int
GameScore )
        {
            this.GameDifficulty = GameDifficulty;
            this.GameDate = GameDate;
            this.GameScore = GameScore;

        }
        #endregion

        #region Properties
        public GameDifficulties GameDifficulty
        {
            get => gameDifficulty;
            set
            {
                gameDifficulty = value;
                OnPropertyChanged(nameof(GameDifficulty));
            }
        }
        public DateTime GameDate
        {
            get => gameDate;
            set
            {

```

```

        gameDate = value;
        OnPropertyChanged(nameof(GameDate));
    }
}
public int GameScore
{
    get => gameScore;
    set
    {
        gameScore = value;
        OnPropertyChanged(nameof(GameScore));
    }
}

#endregion

#region Events
public event PropertyChangedEventHandler? PropertyChanged;
#endregion

#region Handlers
protected void OnPropertyChanged(string propertyName)
{
    PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
}
#endregion

#region Methods
#endregion
}
}

```

**DifficultyWindow.xaml**

```

<Window x:Class="SnakeGame.DifficultyWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SnakeGame"
    mc:Ignorable="d"
    Title="Difficulty" Height="180" Width="250"
    WindowStartupLocation="CenterOwner"
    Background="#FF71CBB4"
    ResizeMode="NoResize">

    <Window.Resources>
        <Style TargetType="RadioButton">
            <Setter Property="FontWeight" Value="Bold"></Setter>
            <Setter Property="FontSize" Value="35"></Setter>
            <Setter Property="Foreground" Value="#FF09624F"/>
            <Setter Property="FontFamily" Value="Game Over"/>
        </Style>

        <Style TargetType="Button">
            <Setter Property="FontFamily" Value="DPComic"/>
            <Setter Property="FontSize" Value="20"/>
            <Setter Property="Foreground" Value="#FF70FFBC"/>
            <Setter Property="BorderBrush" Value="Silver"/>
        </Style>
        <ControlTemplate TargetType="Button" x:Key="MyButton">
            <Border CornerRadius="10"
                BorderBrush="{TemplateBinding BorderBrush}"
                BorderThickness="2"
                Background="SeaGreen"
                Height="Auto"
                >
                <ContentControl Margin="{TemplateBinding Padding}"
                    HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                    VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"
                    Content="{TemplateBinding Content}" />
            </Border>
            <ControlTemplate.Triggers>
                <Trigger Property="IsMouseOver" Value="True">
                    <Setter Property="BorderBrush" Value="#FF2AD735"/>

```

```

        </Trigger>
    </ControlTemplate.Triggers>
</ControlTemplate>
</Window.Resources>
<Border Margin="10" BorderBrush="SeaGreen" BorderThickness="2">
    <Grid >
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
        </Grid.RowDefinitions>

        <StackPanel          Name="DifficultyOptions"          Margin="10"
Orientation="Vertical" Grid.Row="0">
            <RadioButton Content="Hard"></RadioButton>
            <RadioButton Content="Medium"></RadioButton>
            <RadioButton Content="Easy"></RadioButton>
        </StackPanel>
        <WrapPanel          Grid.Row="1"          Orientation="Horizontal"
HorizontalAlignment="Right" VerticalAlignment="Center" >
            <Button IsCancel="True" MinWidth="75" Margin="0,10,10,0"
Template="{StaticResource MyButton}">Cancel</Button>
            <Button          IsDefault="True"          Click="SetDifficulty_Click"
Margin="0,10,10,0"          MinWidth="75"          Template="{StaticResource
MyButton}">Set</Button>
        </WrapPanel>
    </Grid>
</Border>

</Window>

```



**DifficultyWindow.xaml.cs**

```

using SnakeGame.f_ViewModel;
using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace SnakeGame
{
    /// <summary>
    /// Interaction logic for DifficultyWindow.xaml
    /// </summary>
    public partial class DifficultyWindow : Window
    {

        public DifficultyWindow()
        {
            InitializeComponent();
            ShowPrevSetDifficulty();
        }

        private void ShowPrevSetDifficulty()
        {
            GameDifficulties prevSetDif;
            try
            {
                prevSetDif = SnakeGameFileManager.GetDifficultyFromFile();
            }
            catch (Exception)
            {
                prevSetDif = GameDifficulties.Easy;
            }
            foreach (var option in DifficultyOptions.Children)
            {
                if (option is RadioButton choice && choice.Content.ToString().Trim()

```

```

== prevSetDif.ToString())
    {
        choice.SetValue(RadioButton.IsCheckedProperty, true);
    }
}

private void SetDifficulty_Click(object sender, RoutedEventArgs e)
{
    this.DialogResult = true;
    string difficulty = null;
    foreach (var option in DifficultyOptions.Children)
    {
        if (option is RadioButton choice && choice.IsChecked == true)
        {
            difficulty = choice.Content.ToString().Trim().ToLower();
        }
    }
    if (difficulty != null)
    {
        if (difficulty == "hard")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Hard);
        if (difficulty == "medium")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Medium);
        if (difficulty == "easy")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Easy);
    }
}
}
}

```

**MainWindow.xaml**

```

<Window x:Class="SnakeGame.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SnakeGame"
    mc:Ignorable="d"
    Title="MainWindow" Height="450" Width="350"
    WindowStartupLocation="CenterScreen"
    ResizeMode="NoResize"
>

<Grid x:Name="MenuGrid" >
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"></ColumnDefinition>
        <ColumnDefinition Width="2*"></ColumnDefinition>
        <ColumnDefinition Width="*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="*"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="*"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.Background>
        <RadialGradientBrush GradientOrigin="0.5,0.5">
            <GradientStop Color="#FF71CBB4" Offset="1" />
            <GradientStop Color="#FFFCE087" Offset="0" />
        </RadialGradientBrush>
    </Grid.Background>

    <StackPanel Grid.Row="1" Grid.Column="1">
        <StackPanel.Resources>
            <Style TargetType="Button">
                <Setter Property="FontFamily" Value="DPComic"/>
                <Setter Property="FontSize" Value="20"/>
                <Setter Property="Foreground" Value="#FF70FFBC"/>
                <Setter Property="BorderBrush" Value="Silver"/>
            </Style>
            <ControlTemplate TargetType="Button" x:Key="MyButton">
                <Border CornerRadius="25"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="2"

```

```

        Background="SeaGreen"
        Height="50"
    >
        <ContentControl Margin="{TemplateBinding Padding}"
            HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
            VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"
            Content="{TemplateBinding Content}" />

    </Border>
    <ControlTemplate.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
            <Setter Property="BorderBrush" Value="#FF2AD735"/>
        </Trigger>
    </ControlTemplate.Triggers>
</ControlTemplate>
</StackPanel.Resources>
<Button Content="Play"
    Click="StartGameClick"
    Template="{StaticResource MyButton}" />
<Button Content="Difficulty"
    Click="DifficultyWindowClick"
    Template="{StaticResource MyButton}" />
<Button Content="Statistics"
    Click="StatsWindowClick"
    Template="{StaticResource MyButton}" />
</StackPanel>
<TextBlock Grid.Row="0"
    Text="SnakeGame"
    FontSize="100"
    FontFamily="Game Over"
    Foreground="Orange"
    VerticalAlignment="Top"
    HorizontalAlignment="Center"
    Margin="0,55,0,0"/>
    Grid.ColumnSpan="3"

</Grid>
</Window>

```

**MainWindow.xaml.cs**

```

using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace SnakeGame
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {

        public MainWindow()
        {
            InitializeComponent();
        }

        private void StartGameClick(object sender, RoutedEventArgs e)
        {
            SnakeGameWindow game = new SnakeGameWindow();
            game.Owner = this;
            game.ShowDialog();
        }

        private void StatsWindowClick(object sender, RoutedEventArgs e)
        {
            StatsWindow stats = new StatsWindow();
            stats.Owner = this;
            stats.ShowDialog();
        }
    }
}

```

```
private void DifficultyWindowClick(object sender, RoutedEventArgs e)
{
    DifficultyWindow difficulty = new DifficultyWindow();
    difficulty.Owner = this;
    difficulty.ShowDialog();
}
}
enum AppWindows
{
    MainWindow,
    SnakeGameWindow,
    DifficultyWindow,
    StatsWindow
}
}
```

**SnakeGameWindow.xaml**

```

<Window x:Class="SnakeGame.SnakeGameWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SnakeGame"
    mc:Ignorable="d"
    Title="SnakeGameWindow"
    SizeToContent="WidthAndHeight"
    WindowStartupLocation="CenterOwner"
    KeyDown="Grid_KeyDown"
    ContentRendered="GameWindow_ContentRendered"
    x:Name="GameWindow"
    ResizeMode="NoResize">
<Grid Background="DarkSeaGreen">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="15"></ColumnDefinition>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition Width="15"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="15"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="15"></RowDefinition>
    </Grid.RowDefinitions>

    <Border BorderThickness="3"
        x:Name="FieldBorder"
        Grid.Column="1" Grid.Row="2">

        <Grid Height="400" Width="400"
            ShowGridLines="False"
            Name="SnakeField">
            <Grid.Resources>
                <Style TargetType="ColumnDefinition">
                    <Setter Property="Width" Value="*"></Setter>
                </Style>
                <Style TargetType="RowDefinition">
                    <Setter Property="Height" Value="*"></Setter>
                </Style>
            </Grid.Resources>
            <Grid.ColumnDefinitions>
                <ColumnDefinition></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
            </Grid.ColumnDefinitions>

```





```

        <StackPanel          Grid.Row="1"      Grid.Column="1"      Height="40"
Background="SeaGreen" >
    <StackPanel.Resources>
        <Style TargetType="TextBlock">
            <Setter Property="FontWeight" Value="Bold"></Setter>
            <Setter Property="FontSize" Value="33"></Setter>
            <Setter Property="Foreground" Value="#FF70FFBC"/>
            <Setter Property="FontFamily" Value="Game Over"/>
        </Style>
    </StackPanel.Resources>
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Score: " />
        <TextBlock Name="ScoreDisplayer" ></TextBlock>
    </StackPanel>
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Difficulty: " />
        <TextBlock x:Name="DifficultyDisplayer"></TextBlock>
    </StackPanel>
</StackPanel>

</Grid>
</Window>

```

**SnakeGameWindow.xaml.cs**

```

using SnakeGame.f_ViewModel;
using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace SnakeGame
{
    /// <summary>
    /// Interaction logic for DifficultyWindow.xaml
    /// </summary>
    public partial class DifficultyWindow : Window
    {

        public DifficultyWindow()
        {
            InitializeComponent();
            ShowPrevSetDifficulty();
        }

        private void ShowPrevSetDifficulty()
        {
            GameDifficulties prevSetDif;
            try
            {
                prevSetDif = SnakeGameFileManager.GetDifficultyFromFile();
            }
            catch (Exception)
            {
                prevSetDif = GameDifficulties.Easy;
            }
            foreach (var option in DifficultyOptions.Children)
            {
                if (option is RadioButton choice && choice.Content.ToString().Trim()

```

```

== prevSetDif.ToString())
    {
        choice.SetValue(RadioButton.IsCheckedProperty, true);
    }
}

private void SetDifficulty_Click(object sender, RoutedEventArgs e)
{
    this.DialogResult = true;
    string difficulty = null;
    foreach (var option in DifficultyOptions.Children)
    {
        if (option is RadioButton choice && choice.IsChecked == true)
        {
            difficulty = choice.Content.ToString().Trim().ToLower();
        }
    }
    if (difficulty != null)
    {
        if (difficulty == "hard")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Hard);
        if (difficulty == "medium")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Medium);
        if (difficulty == "easy")

SnakeGameFileManager.SaveDifficultyToFile(GameDifficulties.Easy);
    }
}
}
}

```

**StatsWindow.xaml**

```

<Window x:Class="SnakeGame.StatsWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:SnakeGame"
    mc:Ignorable="d"
    Title="Statistics" Height="Auto" MinHeight="100" MaxHeight="200"
Width="300"
    ResizeMode="NoResize"
    WindowStartupLocation="CenterOwner"
    Background="#FF71CBB4">
<Window.Resources>
    <Style TargetType="ScrollBar">
        <Setter Property="Background" Value="#FF55C987"/>
    </Style>
</Window.Resources>

    <Border Margin="10" Grid.RowSpan="2" BorderThickness="1"
BorderBrush="DarkOliveGreen" >
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"></ColumnDefinition>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.Resources>
            <Style TargetType="TextBlock">
                <Setter Property="FontWeight" Value="Bold"></Setter>
                <Setter Property="FontSize" Value="33"></Setter>
                <Setter Property="Foreground" Value="#FF70FFBC"/>
                <Setter Property="FontFamily" Value="Game Over"/>
            </Style>
        </Grid.Resources>

        <Border Grid.Row="0" Background="SeaGreen" Padding="0 0 17 0">
            <Grid Name="StatsHeaderRow" ShowGridLines="false"
Grid.Row="0" >
                <Grid.ColumnDefinitions>
                    <ColumnDefinition ></ColumnDefinition>
                    <ColumnDefinition></ColumnDefinition>

```

```

        <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>

    <TextBlock Grid.Column="0" Grid.Row="0" Text="Difficulty"
HorizontalAlignment="Center"></TextBlock>
    <TextBlock Grid.Column="1" Grid.Row="0" Text="Date"
HorizontalAlignment="Center"></TextBlock>
    <TextBlock Grid.Column="2" Grid.Row="0" Text="Score"
HorizontalAlignment="Center"></TextBlock>
</Grid>
</Border>

<ScrollView Grid.Row="1" Grid.ColumnSpan="2"
VerticalScrollBarVisibility="Visible">

    <Grid ClipToBounds="True" Name="StatsDisplay"
ShowGridLines="False" Background="#FF55C987">
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>

        <Grid.Resources>
            <Style TargetType="TextBlock">
                <Setter Property="FontWeight" Value="Bold"></Setter>
                <Setter Property="FontSize" Value="15"></Setter>
                <Setter Property="Foreground" Value="#FF1E794E"/>
                <Setter Property="FontFamily" Value="DPcomic"/>
                <Setter Property="Height" Value="20"/>
            </Style>
        </Grid.Resources>
    </Grid>

</ScrollView>

</Grid>
</Border>

</Window>

```

**StatsWindow.xaml.cs**

```

using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace SnakeGame
{
    /// <summary>
    /// Interaction logic for StatsWindow.xaml
    /// </summary>
    public partial class StatsWindow : Window
    {
        private List<StatsItem> items;
        public StatsWindow()
        {
            InitializeComponent();
            items = SnakeGameFileManager.GetStatistics();
            ListToStatDisplayBinding();
        }

        public void ListToStatDisplayBinding()
        {
            for (int i = 0; i < items.Count; i++)
            {
                StatsDisplay.RowDefinitions.Add(new RowDefinition());
                for (int j = 0; j < StatsDisplay.ColumnDefinitions.Count; j++)
                {
                    TextBlock innerCellText = new TextBlock();
                    innerCellText.HorizontalAlignment=HorizontalAlignment.Center;
                    innerCellText.SetValue(Grid.RowProperty, i);
                    innerCellText.SetValue(Grid.ColumnProperty, j);
                    Binding cellBind = new Binding()
                    {
                        Source = items[i],

```

```
        UpdateSourceTrigger = UpdateSourceTrigger.PropertyChanged,
    };
    if (j == 0)
        cellBind.Path = new PropertyPath("GameDifficulty");
    if(j == 1)
    {
        cellBind.Path = new PropertyPath("GameDate");
        cellBind.Converter = new DateTimeConverter();
    }
    if(j==2)
        cellBind.Path = new PropertyPath("GameScore");
    innerCellText.SetBinding(TextBlock.TextProperty, cellBind);
    StatsDisplay.Children.Add(innerCellText);
}

}

}

}
```

**ViewModel.cs**

```

using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Threading;

namespace SnakeGame.f_ViewModel
{
    internal class ViewModel
    {

        #region Fields
        private GameStates gameState;
        private GameDifficulties gameDifficulty;
        #endregion

        #region Constructors
        public ViewModel()
        {
            try
            {
                gameDifficulty = SnakeGameFileManager.GetDifficultyFromFile();
            }
            catch(Exception)
            {
                gameDifficulty = GameDifficulties.Easy;
            }

            Field = new Field(gameDifficulty);
            ScoreCounter = new ScoreCounter();
            Timer = new DispatcherTimer() { Interval = TimeSpan.FromSeconds(0.25
- 0.0750 * (int)gameDifficulty) };

            Field.FieldSnake.NotifySnakeIsDead += () => { GameState =
GameStates.NotInGame; };
            Field.FieldFood.NotifyFoodIsEaten += (Food eatenFood) => {
ScoreCounter.AddToScore(eatenFood.ScoreValue); };
            NotifyGameStateChanged += StateGameChanged_Handler;
            Timer.Tick += DispatcherTimer_Tick;
        }
    }
}

```



```
#endregion
```

```
#region Properties
```

```
internal Field Field { get; set; }
internal ScoreCounter ScoreCounter { get; set; }
internal DispatcherTimer Timer { get; set; }
internal GameStates GameState
{
    get => gameState;
    set
    {
        gameState = value;
        NotifyGameStateChanged?.Invoke(value);
    }
}
internal GameDifficulties GameDifficulty
{
    get => gameDifficulty;
    set
    {
        gameDifficulty = value;
    }
}
#endregion
```

```
#region Events
```

```
internal event Action<GameStates> NotifyGameStateChanged;
#endregion
```

```
#region Handlers
```

```
private void StateGameChanged_Handler(GameStates state)
{
    switch (state)
    {
        case GameStates.Paused:
            Timer.Stop();
            break;
        case GameStates.InGame:
            Timer.Start();
            break;
        case GameStates.NotInGame:
            Timer.Stop();
            SnakeGameFileManager.SaveStatistics(new
StatsItem(gameDifficulty, DateTime.Now, ScoreCounter.Score));
            break;
    }
}
```

```

    }
}

private void DispatcherTimer_Tick(object? sender, EventArgs e)
{
    if (GameState == GameStates.InGame)
    {
        Field.SnakeFieldUpdate();
    }
}
#endregion

#region Methods

public void ChangeSnakeDirection(MovementDirections newDirect)
{
    Field.FieldSnake.SnakeDirection = newDirect;
}
#endregion
}
enum GameStates
{
    InGame, NotInGame, Paused
}
}

```

**App.xaml**

```
<Application x:Class="SnakeGame.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:SnakeGame"
    Startup="Application_Startup">
    <Application.Resources>

        </Application.Resources>
</Application>
```

**App.xaml.cs**

```
using SnakeGame.Model;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;

namespace SnakeGame
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {

        private void Application_Startup(object sender, StartupEventArgs e)
        {
            MainWindow window = new MainWindow();
            window.Show();
        }
    }
}
```