МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ім. Б. ХМЕЛЬНИЦЬКОГО

ІНФОРМАЦІЇНІ ТЕХНОЛОГІЇ

ЗАТВЕРДЖУЮ

Завідувач кафедри _____,

_____

(підпис)

„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студента Поліщука Дениса Сергійовича групи КН-19 першого курсу

ТЕМА « Гра тетріс»

Вихідні дані до курсової роботи:

Зміст Пояснювальної записки до курсової роботи:

Індивідуальне завдання

Вступ

1 Огляд ...

2 Розробка схеми алгоритму ...

3 Розробка програми ...

Висновки

Список літератури

Додатки (за необхідності)

Перелік наочного матеріалу:

Програмний продукт ..., пояснювальна записка, презентація результатів роботи.

Календарний план виконання роботи:

No п/п	**Назва етапу дипломного проекту (роботи)**	**Термін виконання етапу: до ...**	**Примітка**
1	Отримання завдання на курсову роботу.	03.03.2020	
2	Огляд джерел технічної інформації за темою роботи	18.03.2020	
3	Виконання аналізу методів реалізації завдання	24.03.2020	
4	Розробка алгоритму реалізації завдання	30.03.2020	
5	Програмування алгоритму, створення програмного продукту	08.04.2020	
6	Тестування розробленого програмного продукту	15.04.2020	
7	Написання пояснювальної записки	01.05.2020	
8	Подача роботи керівнику для написання відгуку	07.05.2020	
9	Корегування роботи за результатами розгляду керівника. Остаточне оформлення пояснювальної роботи.	14.05.2020	
10	Написання доповіді, створення слайдів.	15.05.2020	
11	Захист курсової роботи	26.05.2020	

Студент _____ / Поліщук Д.С.

Науковий керівник _____/ Царик Т.Ю.

Завдання видане «03» 03.2020 року

Міністерство освіти і науки України

ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ім. Богдана Хмельницького

Факультет Обчислювальної техніки та управляючих систем

Кафедра Інформаційні технології

Курсова робота

з дисципліни « Основи програмування, програмування та алгоритмічні мови, Алгоритмізація та програмування »

На тему: Гра тетріс

Студента 1 курсу, групи КН-19 .

спеціальності «Комп'ютерні науки» .

Поліщук Д.С.

Керівник к.т.н., ст. викл каф ІТ

Царик Т.Ю.

Національна шкала: _____

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії _____ Стабецька Т.А.

_____Веретельник В.В.

_____Царик Т.Ю.

Черкаси, 2020

Зміст

Вступ

Тетріс - це одна з перших комп’ютерних ігор, що являє собою головоломку. Створена та розроблена радянським програмістом Олексієм Пажитновим. Гра була випущена 1984 року.

Правила Гри – випадкові фігурки падають зверху в прямокутний стакан розмірами 20 на 10 клітинок. Під час падіння гравець може рухати фігурку по горизонталі та повертати її на 90°.Також можливо скинути фігурку вниз, вона буде летіти вниз доки не торкнеться нижньої стінки ігрової карти, або доки не торкнеться іншої фігурки. Якщо заповнився горизонтальний ряд з 10 клітинок, він пропадає и все що вище нього рухається вниз на одну клітинку. Додатково гравець може бачити наступну фігурку, що дає змогу планувати наступний хід під час гри. Поступово темп гри зростає, це залежить від кількості знищених рядів. Гра закінчується коли нова фігурка не може помістится в стакан. Гравець отримує додаткові бали за кожний знищений за раз ряд, тому головним завданням гравця знищувати як можна більше рядів за раз, щоб отримати більше балів перед тим як гра прискориться.

Можлива проблема - зазвичай гравець програє у зв’язку з тим, що не може впоратись з занадто швидким темпом гри або тому що гра реагує на клавіші дуже повільно порівняно з ростом темпу падіння фігурок, у зв’язку з чим гравець уже не може зробите необхідну кількість зрушень.

Дана тема є актуальною в плані вивчення програмування та покращення своїх логічних здібностей.

Мета курсової роботи полягає в відтворені особливостей оригінальної гри.

Гра буде розроблена в середовищі Visual Studio за допомогою мови програмування C# та Windows Forms. Звісно написання гри за допомогою Game engine значно спростило би її розробку але я вирішив не відходити від канонів.

Розділ 1. Огляд алгоритмів роботи

При створенні даної гри було поставлено головну умову: відтворити всі механіки оригінальної гри.

Мета курсової роботи полягає в освоєні мови C# за допомогою відтворення та розробки логіки гри.

В результаті кінцевий варіант повинен мати такий функціонал:

- Рух фігурки вліво та вправо;
- Падіння та рух вниз;
- Поворот фігурки на 90 градусів;
- Знищення заповнених рядків;
- Кінець гри;
- Показ наступної фігурки;
- Прискорення гри;
- Нарахування балів та ліній за знищення ряду;
- Пауза;

Таким чином ми маємо повністю відтворити функціонал оригінальної гри та забезпечити зрозуміле керування;

1.1 Вхідні та вихідні дані

Вхідні дані являють собою зчитування натискань на клавіші та на кнопки миші

Для зручності гри у гравця буде зрозумілий інтерфейс та інструкція до нього. При обробці дій гравця вступає ігрова логіка, за допомогою якої відтворюється взаємодія з грою та можливість візуально спостерігати наслідок дій гравця. Ігрове поле має представляти собою так би мовити «Стакан», що складається з клітинок та розмірністю 10 клітинок по горизонталі та 20 клітинок по вертикалі, в результаті

маємо поле розміром в 200 клітинок. Гравець матиме змогу відстежити свій результат у реальному часі за допомогою «Таблиці результату», яка містить інформацію про зароблені бали та знищені ряди. При натисканні паузи ігрове поле буде перекривати панель з надписом «Pause». Коли гравець програє він отримає повідомлення про завершення гри та результати.

1.2 вибір конкретних алгоритмів

Більшість можливостей буде реалізовано за допомогою функцій, циклів та графіки.

Функція в C# - це невелика підпрограма всередині основної програми, завдання якої виконання певної задачі та зменшення обсягу основної програми. Інколи функцію ще називають методом. Оголошення функції має таку структуру:

[Модифікатор Доступу] [тип поверненого значення] [ім’я самої функції](Вхідні дані)

{

Тіло фунції;

}

Цикл використовують для задач які використовують повторення одних і тих самих дій за певної умови. В C# існує чотири типів циклів : do…while, while, та for foreach. Цикл foreach відрізняється тим що його використовують в масивах для проходження по кожному елементу.

Програмування графіки буде реалізована за допомогою Windows Forms. Windows Forms – це інтерфейс програмування додатків, за його допомогою буде створена абсолютно весь інтерфейс та графіка гри. Цей інтерфейс простий в освоєнні та потребує знання однієї мови програмування – C#.

Важливу роль відіграють також оператори вибору такі як switch та if else. Кожна перевірка буде виконуватися з їх допомогою. Алгоритм дії if else дуже простий на вхід йде булева інформація яка має тільки два значення істина або хиба, в залежності від значення виконуються різні інструкції. Switch має аналогічний алгоритм, тільки його виконання йде за допомогою «Кейсів». Булеві значення мають такі оператори (їх ще називають – логічні оператори) в мові C# :

- == рівне;
- && умовне і;
- || умовне або;
- ! заперечення;

За допомогою конструкторів буде реалізовано створення фігурок.

Конструктор – це метод класу, який призначений для ініціалізації об’єктів при їх створені

Ініціалізація – це задання певних початкових параметрів змінних при їх створені.

```
Public[Ім’я_класу]([аргументи])
{
  //тіло_конструктора
}
```

Конструктор також може мати параметри.

1.3 Пошук інформації для розробки гри

Оскільки моєю метою було розробити тетріс саме на C# без використання Game engine, це викликало деякі труднощі але в результаті я створив особисту гру , використовуючи лише знання які я отримав протягом навчання на першому курсі.

Основну інформацію що до реалізації всіх алгоритмів та графіки, які потребувала гра для свого функціювання, я брав з лекцій, які були викладені викладачем у «classroom».

Основною ідеєю для алгоритмів послугувала стаття на тему: «Тетрис в сто строк», в цій статті були реалізовані фундаментальні алгоритми такі як рух фігурок та знищення рядків.

Для створення повноцінної гри з її особливостями необхідно було розробити алгоритми повороту, показу наступної фігурки, оновлення кадру та зміни швидкості гри.

Було прийнято рішення зробити фігурки як матрицю або двовимірний масив для розробки повороту фігурки, поворот реалізований за допомогою транспонування матриці початкової фігурки.

Для створення графіки потребувалися додаткові знання, в результаті за знайденим алгоритмом реалізовано графіку яка змінює картинку за час який рівний тіку таймера.

Розділ 2. Розробка алгоритму для гри

2.1 Графіка

Основний інтерфейс було створено за допомого форми, на якій присутні: інструкція, ігрове поле, кнопка стоп/плей, панель паузи та результати.

Ігрове поле або «Стакан» створено за допомогою сітки та клітинок. Сітка створена за допомогою функції DrawLine(), а клітинки за допомогою функції FillRectangle().Графіка фігурок буде пояснена в логікі ігрового поля, оскільки там це буде більш зрозуміло

2.2 Логіка

Заповнення фігурок та їх відображення буде реалізовано так: ігрове поле буде представляти з себе двовимірний масив розмірністю 20 на 10 елементів та заповненій нулями, в той час як фігурка створена з ненульових значень в власній матриці. При розміщені фігурки на поле буде поступати команда про зміну значення елементів ігрового поля в кінці матимемо схематичну картинку з нулів та одиниць.

```
0;0;0;0;1;0;0;0;0;0;
0;0;0;0;1;0;0;0;0;0;
0;0;0;0;1;1;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
```

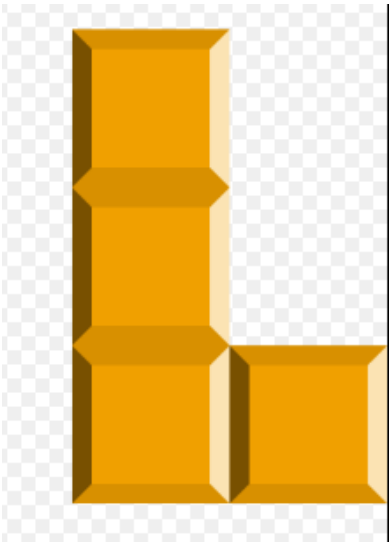


Рис. 1 2.2 Приклад зображення фігурки в матриці порівняно з граф. представленням

Рух фігурок буде залежити від «Tick» таймера, який буде створений в формі на панелі інструментів «Timer». Для фігурок буде створено функцію, яка буде рухати їх в трьох напрямках: ліво, право та низ, це буде реалізовано за допомогою зсунення фігурок у масиві ігрового поля.

Поворот фігурки реалізовано за допомогою математики, адже фігурки в логічному сенсі це матриці , а для того щоб повернути фігурку достатньо лише транспонувати її матрицю. А для повороту фігурки на 90 градусів застосуємо так рівняння : Проміжна матриця[i, j] = основна матриця [j, (розмірність матриці - 1) - i], після чого йде присвоєння основній матриці значення проміжної матриці.

Дотик фігурки реалізовано за допомогою її координат. Фігурка дотикається якщо на своєму шляху наткнеться на іншу фігурку або впаде в кінець ігрової картою.

Випадковість створення фігур виконано за допомогою псевдовипадкової змінної.

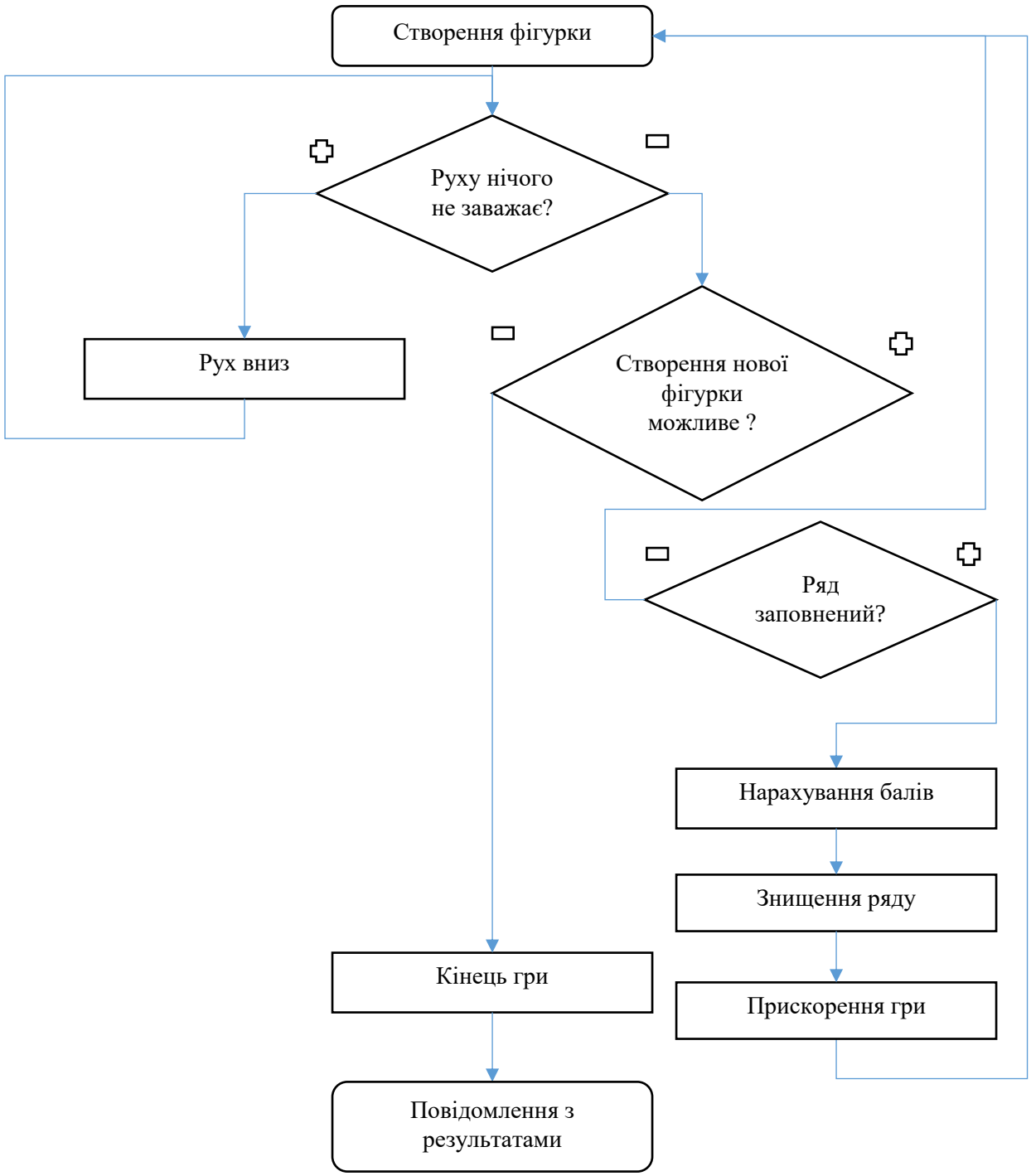
Знищення заповненого ряду відбувається перевіркою кількості ненульових значень в ряду, якщо ця кількість дорівнює довжині ряду то весь ряд прирівнюється до нуля, нараховуються бали та знищені лінії і всі значення здвигаються вниз, як і описано в правилах.

```
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;1;0;0;1;0;
0;0;0;1;0;1;1;1;1;0;
1;1;1;1;1;1;1;1;1;1;
```

```
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;0;0;0;0;0;
0;0;0;0;0;1;0;0;1;0;
0;0;0;1;0;1;1;1;1;0;
```

Рис. 2 2.2 Приклад знищення заповненого ряду на матриці

Загальна схема гри:



Нарахування балів виконується так(I = кількість нарахованих балів):

- Один ряд – I = 10;
- Два ряди – I = 10 + 20;
- Три ряди – I = 10 + 20 + 30;
- Чотири ряди – I = 10 + 20 + 30 + 40;

Отже, даний алгоритм нарахування балів може бути виконаний в виді циклу з початковим показником 1 а кінцевим 4 з формулою:

$$Score += 10 * i$$

Для прискорення гри кожен знищений ряд буде віднімати певне число від інтервалу.

Якщо гравець програє то всі значення масиву ігрової карти прирівнюються до нуля.

2.3 Використані функції

Для створення випадкових фігурок використовується клас Random, який генерує випадкові цілі або дробові числа в певному діапазоні. За «зерно» генерації в конструкторі Random() по замовчуванню береться час на даний момент. Можлива реалізація конструктора через задане «зерно» таким чином: Random(int seed). Int seed – це число, яке використовується для обчислення начального значення послідовності випадкових чисел. Методи цього класу : Next() – повертає випадкове ціле число, яке більше або рівне нулю, але менше за Int32.MaxValue, також можливо повертати випадкове число в певному діапазоні наприклад: Next(2, 5), такий метод поверне випадкове число від двох включно та до чотирьох теж включно, для генерації дробового числа використовують метод NextDouble(), який повертає число від 0.0 до 1.0 .

Для відтворення графіки використано лише функції FillRectangle() та DrawLine(), а всі інші елементи інтерфейсу створені за допомогою елемента Label.

Функція FillRectangle() – малює прямокутники в заданих координатах та з заданими розмірами.

Функція DrawLine() – малює лінію по двом заданим точкам.

Для розробки графіки використаю середовище Visual Studio. Після запуску середовища створюємо графічний проект Windows Forms Application, після чого відкриється.

Після створення проекту відкриється конструктор форми в якому є певний інструментарій та налаштування всіх обраних об’єктів.

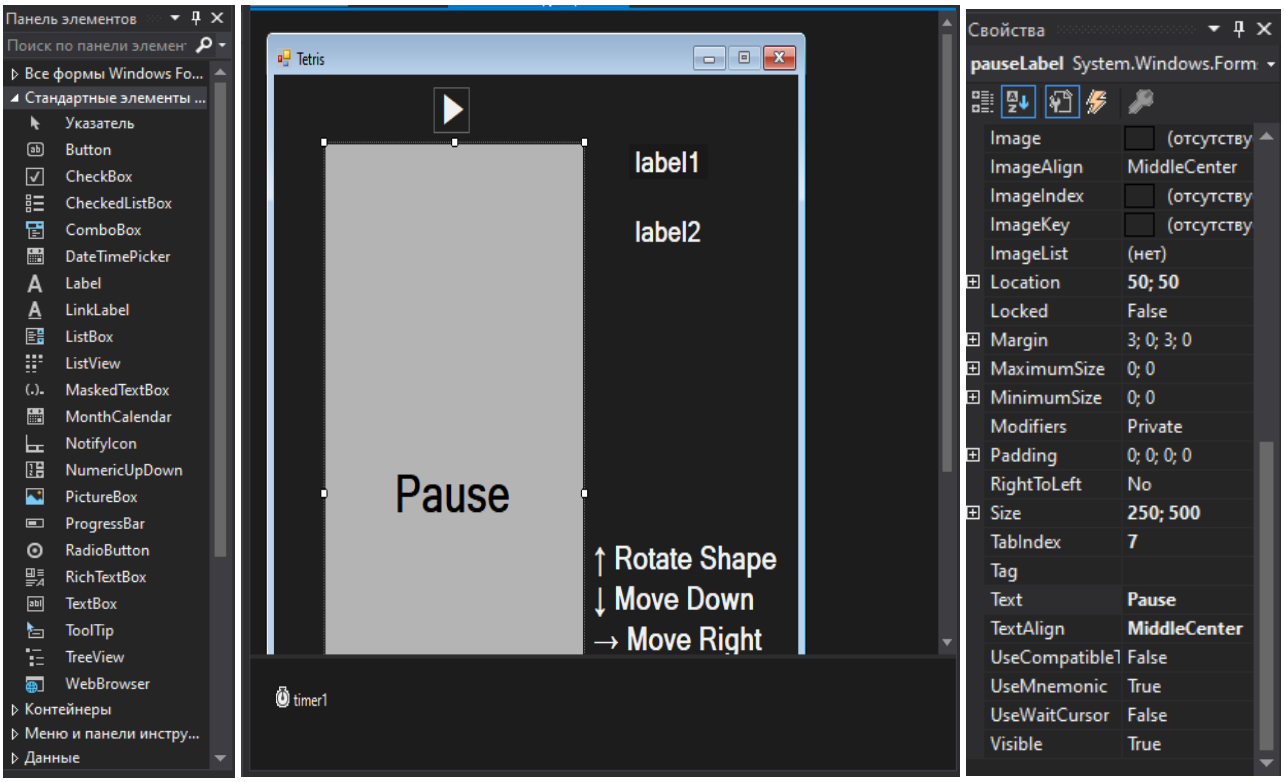


Рис. 3 2.3 приклад панелі інстрементів, вікна конструктора та властивостей обраного елемента.

Вся гра пов’язана з подією яку створює інструмент «Timer».

«Timer» - це один із найважливіших класів, даний клас дозволяє запускати певні дії по закінченню деякого періоду часу. Наприклад: потрібно запускати яку-небудь функцію кожні 3000 мілісекунд, тобто раз в три секунди. Спершу створюється об’єкт делегата TimerCallback, який приймає функцію як параметр. Також даний метод повинен приймати параметр типу object. Після чого створюється таймер, він приймає чотири параметра:

Об’єкт делегата TimerCallback;

Об’єкт, який передається як параметр в метод Count;

Кількість мілісекунд, через які таймер буде запускатися.

Інтервал між викликом метода Count;

Підсумок:

- Реалізація графіки краще всього робити за допомогою графічних функцій та конструктора Windows Forms;
- Все що пов’язано з фігуркою потрібно розмістити в окремому класі, для більшої зручності та спрощення головного коду.
- Оскільки функція Random() повертає псевдовипадкове число, що залежить від часу на момент використання, можуть виникнути деякі друднощі.
- Для повороту фігурки не достатньо лише транспонувати її матрицю
- Для створення приємного інтерфейсу потрібно використовувати конкретну палітру кольорів, які личать один одному.

Розділ 3 Реалізація гри мовою C#

3.1 клас Shape

В класі Shape знаходяться все що стосується фігурок.

Рух фігурок працює за рахунок трьох функцій MoveDown(), MoveRight() та MoveLeft(), їх суть дуже проста вони додають або віднімають по одиниці в просторі X а рух вниз додає до координати одиницю(окільки лівий верхній край стакана має координати (0,0)) .

Приклад руху фігурки в грі:

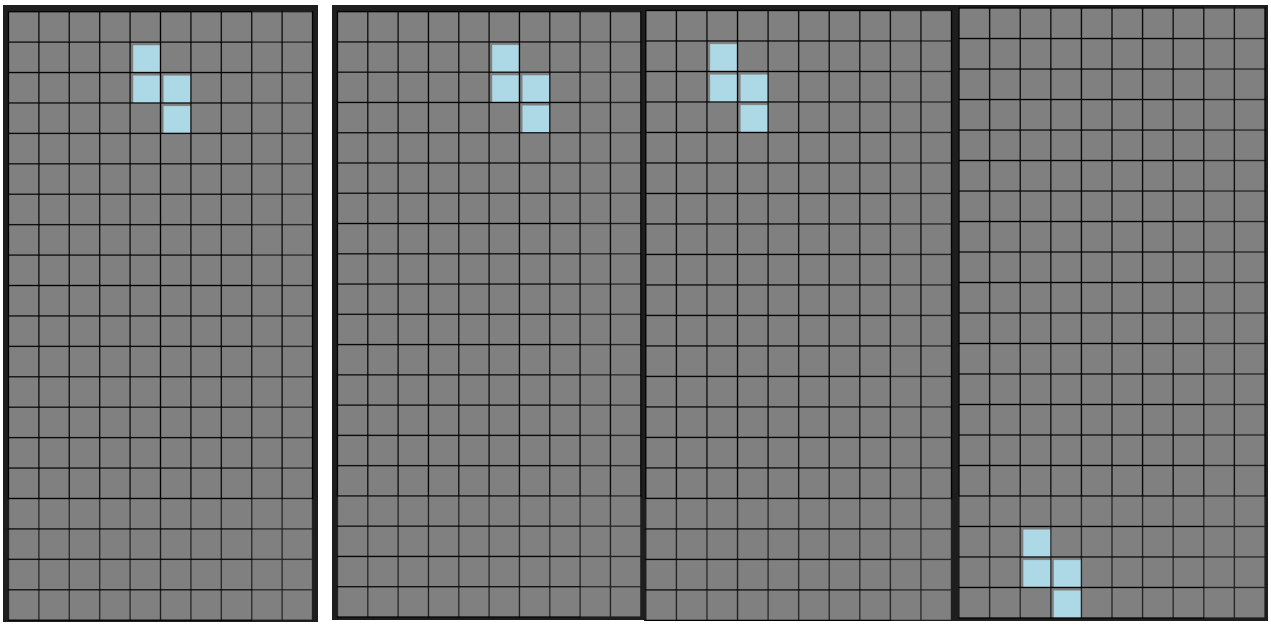


Рис. 4 3.1 демонстрація руху фігурки

Рух фігурки обмежений(щоб не було накладання фігурок та виходу за межі ігрової карти) в основній галці коду.

Генерація матриць відбувається в функції GenerateMatrix(), яка повертає двовимірний масив випадкової фігурки(Їх кількість – 7 , як і в оригіналі, створені вони заздалегідь).

Поворот фігурки зроблений дещо складніше, тому мені потрібно вставити фрагмент коду для розуміння:

```
public void RotateShape()
{
    int[,] tempMatrix = new int[sizeMatrix, sizeMatrix];
    for (int i = 0; i < sizeMatrix; i++)
    {
        for (int j = 0; j < sizeMatrix; j++)
        {
            tempMatrix[i, j] = matrix[j, (sizeMatrix - 1) - i];
        }
    }
    matrix = tempMatrix;
    int offset1 = (8 - (x + sizeMatrix));
    if (offset1 < 0)
    {
        for (int i = 0; i < Math.Abs(offset1); i++)
            MoveLeft();
    }

    if (x < 0)
    {
        for (int i = 0; i < Math.Abs(x) + 1; i++)
            MoveRight();
    }

}
```

Поворт фігурки реалізований за принципом, який я описав в другому розділі.

Перший if перевіряє чи знаходиться фігурка біля правого краю, якщо так, то фігурка при повороті здвигається вліво(в основному коді є обмеження щоб фігурки не накладалися).

Другий if аналогічно перевіряє чи знаходиться фігурка біля лівого краю, якщо так, то фігурка при повороті здвигається вправо(в головному коді: при обмежені що вона не накладається на іншу фігурку).

Конструктор Shape на вхід отримує координати створення фігурки після чого і створює нову фігурку з допомогою функії GenerateMatrix().

Функція ResetShape служить для функціонування показу наступної фігурки, вона присвоює значення наступної фігурки до даної фігурки та оновлює наступну фігурку, Для свого функціоналу також використовує GenerateMatrix().

3.2 Основні функції

Функція Init() ініціалізує всі необхідні змінні для коректної роботи гри, з її допомогою можна швидко налаштувати гру для тестів тощо.

Функція KeyFunc() відповідає за керування під час гри. Вона розпізнає яка саме була натиснута клавіша та виходячи з цього виконує певні дії. Працює за допомогою умовного оператора switch, який приймає будь-яке натиснення клавіші та порівнює його з заданими клавішами, якщо рівності не було знайдено нічого не відбувається.

Функція Update() – слугує оновленням графіки та логіки з плином часу або примусово. В ній викликається функція SliceMap(), яка перевіряє повноту ряду, реалізує їх знищення та оновлює результат.

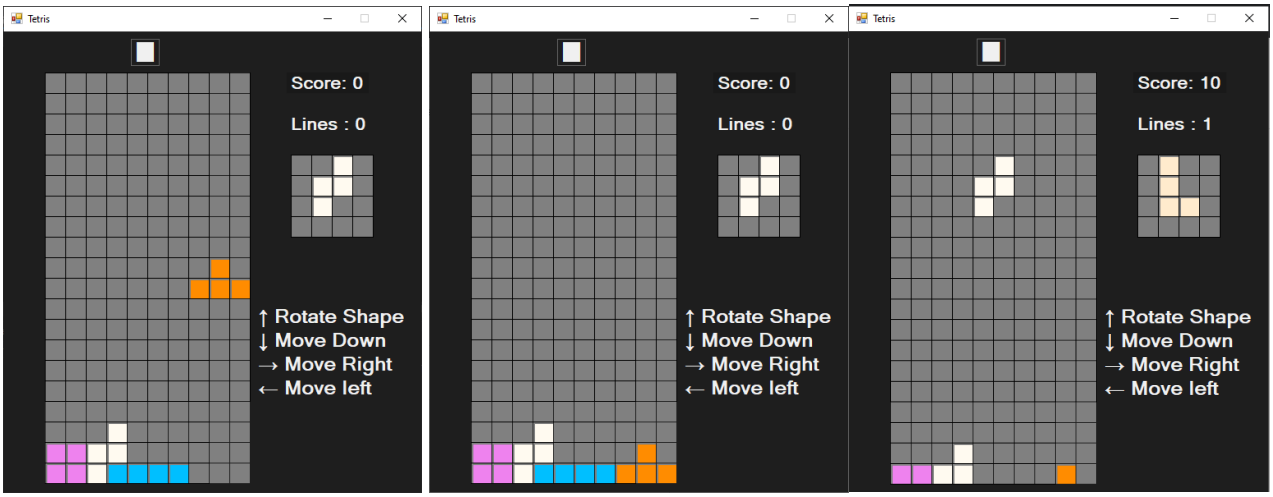


Рис. 5 3.2 Приклад знищення ряду

Також є функції перевірки налягання фігурок виходу за ігрову карту з їх допомогою реалізовано:

- Кінець гри;
- Можливість руху фігурки;
- Можливість повороту фігурки;

Для того щоб фігурки залишалися на ігровій карті та «склеювалися» один з одним була створена така функція як Merge()

```
     for (int i = currentShape.y; i < currentShape.y +
currentShape.sizeMatrix; i++)
{
     for (int j = currentShape.x; j < currentShape.x +
currentShape.sizeMatrix; j++)
     {
     if (currentShape.matrix[i - currentShape.y, j -
currentShape.x] != 0)
     map[i, j] = currentShape.matrix[i - currentShape.y, j -
currentShape.x];
     }
}
```

Її функціонал досить простий та не потребує детального розгляду.

Функція ResetArea() одна з головних функцій. ЇЇ функціонал: вона реалізує графіку та логіку ігрової карти, тому що при кожному русі чи падінні фігурки за нею залишається її слід, який ця функція видаляє з ігрового поля

Функції графіки:

- DrawNextShape() малює наступну фігурку для спрощення та планування гри;

- DrawGridForNextShape() малює місце де буде показ наступної фігурки;

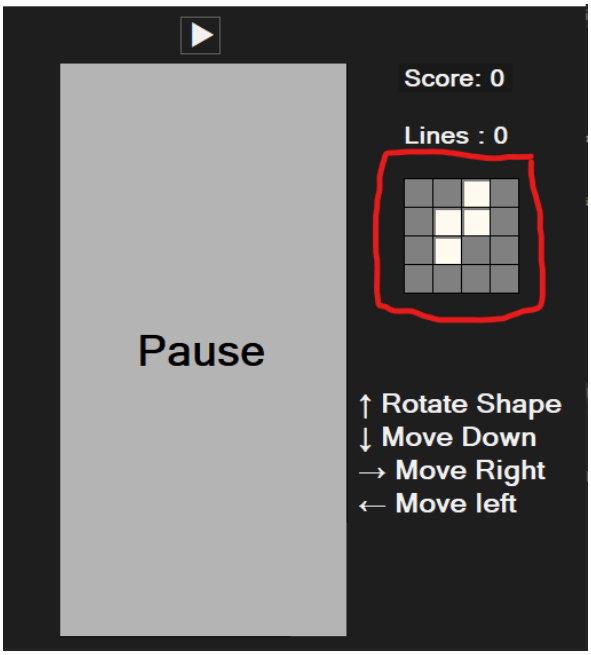


Рис. 6 3.2 Показ наступної фігурки

- DrawGrid() – функція що малює сітку для ігрової карти;
- DrawMap() – функція яка малює фігурку на ігровій карті;
- OnPaint() – функція, яка викликає всі функції графіки;

3.3 Проблеми що виникли при створенні гри

Головною проблемою виявилося створення випадкових фігурок, адже клас Random не є справжним генератором випадкових чисел, він містить генератор псевдовипадкових чисел. Кожен екземпляр класа Random має деякий внутрішній стан і при виклику метода Next (або NextDouble, або NextBytes) метод використовує цей стан для повернення числа, яке буде вважатися випадковим. Після цього внутрішній стан змінюється таким чином щоб, при наступних викликах метода, він повертав число яке б здавалося випадковим, відмінне від повернутого раніше. Якщо взяти декілька екземплярів класу Random з одним і тим внутрішнім станом, який задається параметром Seed (по замовчуванню час на момент виклику), то вкінці ми отримаємо

однаковий результат, що і відбувалося при створені даної фігурки та наступної(перші дві фігурки завжди були однакові).

Для вирішення цієї проблеми – можна використовувати один екземпляр для створювання великої кількості випадкових чисел, але в такого варіанта є свій мінус: при великій кількості створень є ймовірність обнулити внутрішній стан, після чого екземпляр перестане приносити користь, що і сталося. При довготривалій грі в певний момент часу створюються одні й ті самі фігурки (до 7 однакових фігурок, ймовірність чого 8.4*10^(-6)), такий результат спостерігався майже завжди коли було знищено більше 30 ліній.

Іншим і остаточним вирішенням даної проблеми – це задання певного параметра при створені нового екземпляра, який би не залежив від часу на момент виклику.

r = new Random(GetHashCode() + (int)DateTime.Now.Ticks);

За зерно я взяв хешкод об’єкта.

Висновок

У результаті виконанні роботи я повністю відтворив оригінальну гру тетріс, за виключенням нарахування балів. Весь проект було реалізовано за допомогою середовища Visual Studio та мови C#. Гра відповідає всьому функціоналу, який був на меті створення. За рахунок використання правильної палітри кольорів та дружелюбного інтерфейсу – враження приємної графік(на мій розсуд). Під час тестування гри я не виявив ніяких похибок у виконанні програми. Для подальшого розвитку гри можливі такі покращення:

- Більш коректна анімація повороту фігурки;
- Додавання музичного супроводу до гри;
- Покращення графіки та надання об’ємності фігуркам;
- Створення рівнів складності;
- Створення ігрового меню;
- Оптимізація коду;

Список використаної літератури

1. Методичні вказівки до виконання та оформлення курсової роботи з дисциплін “Основи програмування”, “Програмування та алгоритмічні мови”, “Алгоритмізація та програмування” для студентів, які навчаються за напрямами підготовки 050101 – „Комп’ютерні науки”, 050103 – „Програмна інженерія”, 040303 – „Системний аналіз” усіх форм навчання.
2. https://docs.microsoft.com/en-us/dotnet/api/system.random?view=netcore-3.1
3. https://habr.com/ru/post/165459/
4. https://metanit.com/sharp/windowsforms/1.1.php
5. https://habr.com/ru/post/433908/
6. https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%82%D1%80%D0%B8%D1%81