Дипломна работа

на тема

.....Tetris.....

Дипломант: **София Светланова Велева**
ученик от 12А клас в
ЧПГДН "СофтУни Светлина"

Научен ръководител: **Николай Руменов Сапунов**
Рецензент: **XXX XXX XXX**

Дата: **12 май 2023**

Сесия: май-юни 2023 г.

София, 2023 г.

Съдържание

Увод

Настоящата разработка има за цел да запознае читателя с различните видове код, които са нужни за създаването на една изчистена и оптимизирана игра на “Tetris”. Дипломният проект ще е обект на изследването на видео игра, която все още е актуална, дори и след четиридесет години на маркета. Този проект не само работи върху създаването на версия на “Tetris” с различни нива на трудност, но и показва респект към една от най-известните и стари игри.

Проблем и история

Проблемът разглеждан в текущата дипломна работа е свързан със свободното време на повечето хора по света. Днешно време голяма част от всекидневието на един човек е изпълнено с малки моменти, в които не може да се направи нещо продуктивно, или интересно. Tetris е игра, която може да се играе без значение в коя част от деня - тя е интересна, лесна и не отнема много време. Играта е достъпна за всеки - може да се играе дори и на стари Nokia телефони. Днешно време хората играят на мобилни игри по телефоните си. Главно Candy Crush Saga, или може би някоя игра със събиране на герои - те обаче лишават играча от нещо, ако загуби. Tetris обаче не наказва играчите си - може да се играе безкрайно и винаги човек може да започне отначало, без да се чувства лишен от удоволствие. Все пак хубавото на Tetris е бавното повдигане на трудността.

Цели на дипломния проект

Целта на настоящата дипломна работа е да се разработи оптимизирана версия на “Tetris”. Тя трябва да изглежда добре визуално и да е интуитивна - важно е потребителят да може да използва приложението без да трябва да търси ръководство стъпка по стъпка как се използва даденият UI. Главната идея е да се създаде интерактивно и интересно приложение, което може да се използва от всички. Важно е приложението да има степени на трудност - това би го направило по-интересно.

Целта на настоящата дипломна работа е да бъде …

Във връзка с поставената цел за разработване на оптимизиран “Tetris” са поставени следните задачи:

- Да се създаде интересен дизайн

- Проучване на инструментите и технологиите за работа с импортирани снимки и файлове във Visual Studio.
- Проектиране на система за намиране на запълнени редове.
- Реализация на системата за намиране и премахване на запълнени редове плюс пресмятане на резултат при брой запълнени редове.
- Изчисляване на брой редове и премахване на ненужните редове.
- Създаване на бутони и поставяне на функциите им.
- Създаване на решетка, в която ще е се извършват движенията на блоковете.
- Гладки движения на блоковете плюс възможност за завъртане на блоковете.
- Реализация на клиентската част. Създават се лесни за използване бутони.
- Добавяне на работещ “play again” бутон.
- Запазване на определени блокове.
- Добавяне на повишаваща се трудност.
- Добре изглеждаща анимация на движение.

Глава 1. Проучване и история

Един от най-големите маркети в света е този на видеоигрите. Има стотици жанрове на видеоигрите - от roguelike до игри с пъзели, като може би най-известните от тях са игрите със стрелба за хората под двадесет години и игрите с пъзели за хора в средната възраст.

Една от най-известните игри е именно Tetris. Tetris е може би играта за която повечето хора се сещат, като чуят думата “видеоигра”. С нейните цветни блокчета и интересна механика, играта е лесно запомняща се. Името и произлиза от комбинацията на гръцката представка “тетра”, която означава четири квадратчета и думата “тенис”. Tetris се смята за класика - може да се играе на всяка електронна система. Някои хора дори и са програмирали Tetris на хладилниците си.

Идеята на Tetris е играчът да нареди различни движещи се фигури така че да запълнят един, или повече редове. Играта е безкрайна, обаче с времето става все по-трудна и по-трудна. Има много вариации, като най-известните включват добавяне на допълнителни редове и нови фигури.

Изненадващо за много хора, днес съществуват Tetris състезания - много от които са дори и на световно ниво. Тези състезания не са много известни за повечето хора, но нишовата общност, която се интересува от тях, е пълна с отдадени фенове и спонсори. Състезанията по Tetris стават все

по-популярни през последните 20 години, като играчи от цял свят се състезават за високи резултати и награди. Тези състезания варират от неофициални местни събития до широкомащабни международни турнири, като някои предлагат значителни парични награди и други стимули.

Едно от най-известните състезания по Tetris е Световното първенство по класически Tetris, което се състои за първи път през 2010 г. и оттогава се провежда всяка година в Съединените щати. Това събитие включва някои от най-добрите играчи на Tetris в света, които се състезават за голямата награда от няколко хиляди долара. В допълнение към основното състезание, събитието включва други дейности и състезания, като скоростни изпитания и игри в отбор.

Състезателният Tetris обикновено включва играчи, които се опитват да постигнат възможно най-висок резултат в рамките на определен период от време или брой нива. Системата за точкуване на играта се основава на завършване на редове и извършване на специални маневри, като изчистване на няколко реда наведнъж (известно като „Tetris“) или пускане на блок в тясно пространство (известно като „T-Spin“). Играта в състезателния Tetris може да бъде невероятно бърза, като играчите извършват сложни маневри и вземат решения за част от секундата, за да увеличат максимално своите резултати.

Играчите, които са на върха на състезателния Tetris, често имат дълбоко разбиране на механиката на играта и са опитни във визуално-пространственото мислене и бързото вземане на решения. Те могат също така да имат специфични стратегии и техники за справяне с определени комбинации от блокове и трудни ситуации.

В допълнение към Световното първенство по класически Tetris има няколко други състезания по Tetris, провеждани по целия свят, включително първенството по Tetris в Япония, европейското първенство по Tetris и различни онлайн турнири. Тези събития могат да привлекат играчи от всички възрасти и нива на умения, от случайни фенове до професионални играчи, които си изкарват прехраната, като се състезават в игрални събития.

Хубавото на състезателния Tetris е, че човек може да изкарва прехраната си, правейки нещо, което не само развива ума, но и е забавно и интересно.

Състезателният Tetris предлага няколко предимства, включително възможности за общуване и среща с други геймъри, както и шанс да се спечелят награди и да се получи признание за уменията на човек. Състезателният аспект на играта също може да бъде мотиватор за играчите да подобрят своите способности и да научат нови стратегии. За състезателите Tetris е нещо повече от хоби - играта е работа и е източник на доходи.

Като цяло състезанията по Tetris се превърнаха във вълнуващ и завладяващ начин за феновете на играта да тестват уменията си и да се състезават с други. С непрекъснатия растеж на състезателните игри и електронните спортове е вероятно състезанията по Tetris да продължат да бъдат популярна и забавна дейност за години напред.

Механиката на играта Tetris е проста. Играта започва с едно тетрамино, което пада от горната част на екрана, и играчът трябва да го завърти и премести наляво или надясно, за да го вмести в наличното пространство. Играчът може също така да ускори падането на тетраминото или да го накара да пада по-бързо, като натисне клавиша със стрелка надолу. След като тетраминото достигне долната част на екрана или бъде поставено от играча, в горната част на екрана ще се появи ново тетрамино и цикълът се повтаря.

С напредването на играта тетраминатa падат по-бързо и играчът трябва да мисли бързо и стратегически, за да изчисти линиите и да предотврати запълването на игралното поле. Играчът може да върти тетраминатa по посока на часовниковата стрелка или обратно на нея, за да ги постави в различни пространства, а също така може да премества тетраминатa наляво или надясно, като използва клавишите със стрелки. Обикновено, игралната дъска е широка десет клетки и висока двадесет клетки, а тетраминатa са съставени от четири квадрата, подредени в различни форми. Има седем различни вида тетрамино, всяко със своя уникална форма, и всички те падат на случаен принцип от горната част на екрана.

Когато играчът запълни хоризонтална линия, тя изчезва, а всички тетромини над нея паднат надолу, за да запълнят празното пространство. Играчът може да изчисти няколко линии наведнъж, като създаде повече от една хоризонтална линия едновременно. Това се нарича комбо и носи на играча повече точки. Колкото повече линии изчисти играчът в една комбинация, толкова повече точки ще спечели.

Целта на играта е да се изчистят линиите възможно най-дълго, като играта приключва, когато тетрамината се натрупат до горната част на екрана и вече няма място за поставяне на нови тетромини. Резултатът на играча се определя от броя на изчистените линии и броя на създадените комбинации. Играта става все по-трудна, тъй като тетрамината падат по-бързо и играчът трябва да мисли по-бързо и стратегически, за да се справи.

Tetris е популярна видео игра, създадена от руския компютърен програмист Алексей Пажитнов през 1984 г.

Играта бива пусната за първи път в Съветския Съюз и бързо се превръща в сензация, разпространявайки се в други страни и платформи като Nintendo Game Boy, PC и аркадни машини. Пристрастяващият геймплей и простата механика я правят хит сред играчи от всички възрасти.

Tetris е преминала през много итерации и адаптации през годините, като различни компании придобиват правата да произвеждат свои собствени версии на играта. Въпреки това основният геймплей на играта остава до голяма степен същият: играчите трябва да манипулират падащи блокове, за да създадат пълни редове, които изчезват и печелят точки.

Днес Tetris се смята за една от най-емблематичните и разпознаваеми видео игри на всички времена, с културно въздействие, което се простира отвъд света на игрите. Продължава да се играе и да се обича от милиони хора по света, а нови версии на играта все още се пускат на модерни платформи.

Tetris е класическа видео игра, която е популярна от десетилетия и предлага повече от просто забавление. Проучвания показват, че играта на Tetris може да има редица ползи за ума и тялото, които я правят полезна по много начини.

Едно от най-значимите предимства на играта на Tetris е, че може да подобри когнитивните способности като пространствено мислене, решаване на проблеми и критично мислене. Играта изисква играчите бързо да оценяват и манипулират падащи блокове, за да създадат пълни редове, което

може да помогне за подобряване на способността им да мислят стратегически и да вземат бързи решения. Визуално-пространствените предизвикателства на играта изискват от играчите да използват и умения за визуализация, за да идентифицират и подредят различните форми.И заради това обикновените играчи на Tetris могат да обработват по-добре сложна визуална информация в ситуации от реалния свят.

Доказано е също, че играта на Tetris помага за намаляване на тревожността и нивата на стрес. Според редици изследвания, играта може да помогне за отвличането на ума от негативните мисли и да насърчи чувството за релаксация и спокойствие. Играта е особено полезна за хора, които са склонни към безпокойство или депресия, тъй като осигурява разсейване с нисък стрес и изисква фокусирано внимание, което може да отклони ума от притеснения или проблеми.

Освен това Tetris може да бъде забавен начин за прекарване на времето и почивка от други по-стресиращи или взискателни задачи. Тя също е игра, на която могат да се наслаждават хора от всички възрасти и нива на умения, което я прави универсална и достъпна форма на забавление. Човек може да играе Tetris сам или с други, като състезателният характер на играта може да придостави допълнително ниво на вълнение и ангажираност.

Освен тези когнитивни и емоционални ползи, Tetris може да бъде и инструмент за изследователи и клиницисти. Играта е използвана в когнитивни и неврологични изследвания, за да се разбере как мозъкът обработва визуална информация и как видеоигрите могат да повлияят на развитието на мозъка. Изследователите са открили, че играенето на Tetris може да помогне за подобряване на когнитивната гъвкавост, работната памет и дори плътността на сивото вещество в определени области на мозъка. Tetris също е била използвана като лечение на състояния като посттравматично стресово разстройство (PTSD) и травматично мозъчно увреждане (TBI), тъй като е доказано, че помага за намаляване на симптоми като ретроспекции и когнитивно увреждане.

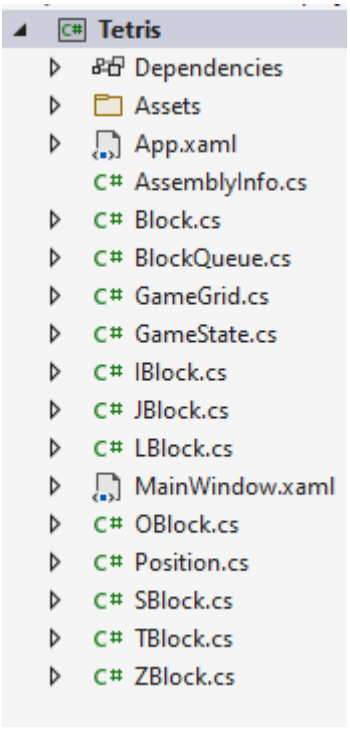
В заключение, докато Tetris може да изглежда просто като обикновена видео игра, тя предлага набор от предимства както за ума, така и за тялото. От подобряване на когнитивните способности до намаляване на безпокойството и стреса и предоставяне на забавна и увлекателна форма на забавление, Tetris може да бъде полезна дейност за много хора. Освен това потенциалното използване на играта в

изследователски и клинични условия предполага, че тя има още повече приложения отвъд развлекателната си стойност.

Глава 2. Проектиране и имплементация

Архитектура на системата

Целият код в дипломната работа се състои от около 1000 реда и 12 класа, 5 от които служат за различните главни функции на играта. И те са именно: движение на кубчетата, състояние на играта, позиция на редове, премахване на редове, граници. Подкласовете, които биват използвани от 5те важни класове, служат като контейнери, които държат информация.



Фиг 1

Класовете, от които е изграден кодът са:

- **Block.cs**

  Това е код на C#, който дефинира абстрактен клас с име "Block" за играта Tetris.

  Модификаторът за достъп "public" указва, че класът е достъпен за всеки код в програмата.

  Класът има няколко член-променливи, включително "rotationState" и "offset", които съхраняват текущото състояние на въртене и отместване на позицията на блока.

```
private int rotationState;
private Position offset;
```

Фиг 2

Класът има и няколко абстрактни свойства - "Tiles", "StartOffset" и "Id", които са дефинирани в производните класове, които наследяват този клас.

```
10 references
protected abstract Position[][] Tiles { get; }
11 references
protected abstract Position StartOffset { get; }
14 references
public abstract int Id { get; }
```

Фиг 3

Конструкторът на класа инициализира член-променливата "offset" с началните позиции на редовете и колоните, определени в свойството "StartOffset".

```
0 references
public Block()
{
    offset = new Position(StartOffset.Row, StartOffset.Column);
}
```

Фиг 4

Класът има няколко метода, включително "TilePositions", който връща колекция от обекти "Position", представляващи текущите позиции на блока върху игралното поле. Методите "RotateCW" и "RotateCCW" завъртат блока съответно по посока на часовниковата стрелка и обратно на нея.

```
2 references
public void RotateCW()
{
    rotationState = (rotationState + 1) % Tiles.Length;
}

2 references
public void RotateCCW()
{
    if (rotationState == 0)
    {
        rotationState = Tiles.Length - 1;
    }
    else
    {
        rotationState--;
    }
}
```

Фиг 5

Методът "Move" премества блока в зададен брой редове и колони. Методът "Reset" (Нулиране) връща блока в първоначалното му състояние.

```
9 references
public void Move(int rows, int columns)
{
    offset.Row += rows;
    offset.Column += columns;
}

1 reference
public void Reset()
{
    rotationState = 0;
    offset.Row = StartOffset.Row;
    offset.Column = StartOffset.Column;
}
```

Фиг 6

Като цяло този клас осигурява основа за дефиниране на различни видове блокове в играта като функциите им за въртене, движение и нулиране.

Той също така дефинира стандартен интерфейс IEnumerable<Position> за достъп до текущата позиция на блока върху игралното поле, което улеснява манипулирането и взаимодействието с блока по време на игра.

```
5 references
public IEnumerable<Position> TilePositions()
{
    foreach (Position p in Tiles[rotationState])
    {
        yield return new Position(p.Row + offset.Row, p.Column + offset.Column);
    }
}
```

Фиг 7

- **BlockQueue.cs**

Класът има частна променлива, наречена "blocks", която е метод от "Block" обекти. Масивът съдържа инстанции на различните видове блокове, които се използват в играта.

```
private readonly Block[] blocks = new Block[]
{
    new IBlock(),
    new JBlock(),
    new LBlock(),
    new OBlock(),
    new SBlock(),
    new TBlock(),
    new ZBlock()
};
```

Фиг 8

Класът има и частна променлива, наречена "random", която е инстанция на класа "Random". Класът "Random" се използва за генериране на случайни числа, които се използват за избор на следващия блок от масива "blocks".

```
private readonly Random random = new Random();
```

Фиг 9

Публичното свойство "NextBlock" получава следващия блок в опашката. Модификаторът "private set" указва, че това свойство може да се задава само в рамките на класа "BlockQueue".

```
5 references
public Block NextBlock { get; private set; }
```

Фиг 10

Конструкторът на класа инициализира свойството "NextBlock" с произволно избран блок от масива "blocks".

```
1 reference
public BlockQueue()
{
    NextBlock = RandomBlock();
}
```

Фиг 11

Методът "RandomBlock" е частен метод, който връща произволно избран блок от масива "blocks".

```
2 references
private Block RandomBlock()
{
    return blocks[random.Next(blocks.Length)];
}
```

Фиг 12

Методът "GetAndUpdate" връща текущия блок в опашката и избира нов блок, който да бъде следващ. Той извиква метода "RandomBlock" и проверява дали новият блок не е същият като текущия. Ако новият блок е същият като текущия, той продължава да избира нов блок, докато не бъде избран различен.

```
3 references
public Block GetAndUpdate()
{
    Block block = NextBlock;

    do
    {
        NextBlock = RandomBlock();
    }
    while (block.Id == NextBlock.Id);

    return block;
}
```

Фиг 13

Като цяло този клас предоставя опашка от различни видове блокове за играта Tetris, позволявайки случаен избор на блокове, които да се използват по време на игра. Класът гарантира, че един и същ блок няма да бъде избран два пъти подред, което прави играта по-разнообразна и интересна.

- **GameGrid.cs**

  Класът представлява решетка, която се използва в играта Tetris. Той съдържа следното:

  частен двумерен целочислен масив, наречен grid, в който се съхраняват стойностите на мрежата.

  Две публични целочислени свойства само за четене, наречени Rows и Columns, които връщат броя на редовете и колоните в решетката, съответно.

```
private readonly int[,] grid;
7 references
public int Rows { get; }
9 references
public int Columns { get; }
```

Фиг 14

Индексер, който позволява достъп до елементите на решетката чрез индекси на редове и колони. Използва се новият оператор => на C# 8.0, за да се осигури кратка реализация на getter и setter за индексатора.

```
2 references
public int this[int r, int c]
{
    get => grid[r, c];
    set => grid[r, c] = value;
}
```

Фиг 15

Конструктор, който приема броя на редовете и колоните като параметри и инициализира масива на решетката до зададения размер.

Три публични метода, които позволяват да се провери дали дадена клетка е вътре в решетката (IsInside), дали е празна (IsEmpty) или дали даден ред е пълен (IsRowFull).

```
1 reference
public bool IsInside(int r, int c)
{
    return r >= 0 && r < Rows && c >= 0 && c < Columns;
}

2 references
public bool IsEmpty(int r, int c)
{
    return IsInside(r, c) && grid[r, c] == 0;
}

1 reference
public bool IsRowFull(int r)
{
    for (int c = 0; c < Columns; c++)
    {
        if (grid[r, c] == 0)
        {
            return false;
        }
    }

    return true;
}
```

Фиг 16

Два частни метода, които извършват операции върху редове: ClearRow изчиства ред, като настройва всички негови стойности на нула, а MoveRowDown премества ред надолу с определен брой редове.

```
1 reference
private void ClearRow(int r)
{
    for (int c = 0; c < Columns; c++)
    {
        grid[r, c] = 0;
    }
}
```

Фиг 17

```
1 reference
private void MoveRowDown(int r, int numRows)
{
    for (int c = 0; c < Columns; c++)
    {
        grid[r + numRows, c] = grid[r, c];
        grid[r, c] = 0;
    }
}
```

Фиг 18

Публичен метод, наречен ClearFullRows, който изчиства всички редове, които са напълно запълнени, и връща броя на изчистените редове.

```
1 reference
public int ClearFullRows()
{
    int cleared = 0;

    for (int r = Rows-1; r >= 0; r--)
    {
        if (IsRowFull(r))
        {
            ClearRow(r);
            cleared++;
        }
        else if (cleared > 0)
        {
            MoveRowDown(r, cleared);
        }
    }

    return cleared;
}
```

Фиг 19

Като цяло този клас осигурява функционалност за работа с игралната мрежа в играта Tetris, включително изчистване на пълни редове и преместване на редове надолу.

- **GameState.cs**

Този код дефинира класа GameState за игра Tetris на C#. Класът GameState представлява текущото състояние на играта и съдържа информация за текущия блок, резултата, мрежата на играта, опашката от блокове, задържания блок и дали играта е приключила или не.

```
10 references
public GameGrid GameGrid { get; }
5 references
public BlockQueue BlockQueue { get; }
3 references
public bool GameOver { get; private set; }
4 references
public int Score { get; private set; }
5 references
public Block HeldBlock { get; private set; }
4 references
public bool CanHold { get; private set; }
```

Фиг 20

Класът GameState има няколко публични метода, които позволяват на потребителя да взаимодейства с играта, включително методи за завъртане на текущия блок по посока на часовниковата стрелка или обратно на нея, преместване на текущия блок наляво или надясно, преместване на текущия блок надолу, пускане на текущия блок и задържане на текущия блок.

```
1 reference
public void HoldBlock()
{
    if (!CanHold)
    {
        return;
    }

    if (HeldBlock == null)
    {
        HeldBlock = CurrentBlock;
        CurrentBlock = BlockQueue.GetAndUpdate();
    }
    else
    {
        Block tmp = CurrentBlock;
        CurrentBlock = HeldBlock;
        HeldBlock = tmp;
    }

    CanHold = false;
}
```

Фиг 21

RotateBlockCW и RotateBlockCCW проверяват дали блокчето може да се завърти и дали пасва в решетката на играта.

```
1 reference
public void RotateBlockCW()
{
    CurrentBlock.RotateCW();

    if (!BlockFits())
    {
        CurrentBlock.RotateCCW();
    }
}

1 reference
public void RotateBlockCCW()
{
    CurrentBlock.RotateCCW();

    if (!BlockFits())
    {
        CurrentBlock.RotateCW();
    }
}
```

Фиг 22

MoveBlockLeft и MoveBlockRight също проверяват дали блокчето може да се завърти и дали пасва в решетката на играта.

```
1 reference
public void MoveBlockLeft()
{
    CurrentBlock.Move(0, -1);

    if (!BlockFits())
    {
        CurrentBlock.Move(0, 1);
    }
}

1 reference
public void MoveBlockRight()
{
    CurrentBlock.Move(0, 1);

    if (!BlockFits())
    {
        CurrentBlock.Move(0, -1);
    }
}
```

Фиг 23

Класът GameState има и няколко частни метода, които се използват вътрешно за актуализиране на състоянието на играта. Тези методи включват BlockFits(), който проверява дали текущият блок се вписва в решетката на играта.

```
6 references
private bool BlockFits()
{
    foreach (Position p in CurrentBlock.TilePositions())
    {
        if (!GameGrid.IsEmpty(p.Row, p.Column))
        {
            return false;
        }
    }

    return true;
}
```

Фиг 24

IsGameOver(), който проверява дали играта е приключила.

```
1 reference
private bool IsGameOver()
{
    return !(GameGrid.IsRowEmpty(0) && GameGrid.IsRowEmpty(1));
}
```

Фиг 25

PlaceBlock(), който поставя текущия блок в решетката на играта и актуализира резултата.

```
2 references
private void PlaceBlock()
{
    foreach (Position p in CurrentBlock.TilePositions())
    {
        GameGrid[p.Row, p.Column] = CurrentBlock.Id;
    }

    Score += GameGrid.ClearFullRows();

    if (IsGameOver())
    {
        GameOver = true;
    }
    else
    {
        CurrentBlock = BlockQueue.GetAndUpdate();
        CanHold = true;
    }
}
```

Фиг 26

Класът GameState има конструктор, който инициализира игралната мрежа и опашката от блокове и задава първоначалното състояние на играта.

```
2 references
public GameState()
{
    GameGrid = new GameGrid(22, 10);
    BlockQueue = new BlockQueue();
    CurrentBlock = BlockQueue.GetAndUpdate();
    CanHold = true;
}
```

Фиг 27

Като цяло класът GameState е важен компонент от реализацията на играта Tetris, тъй като той управлява състоянието на играта и позволява на потребителя да взаимодейства с играта.

- **IBlock.cs**

  Това е клас, наречен IBlock, който наследява класа Block. Той представлява един от седемте вида блокове, използвани в играта Tetris. IBlock е права линия от четири кубчета.

  Класът IBlock замества две свойства на класа Block:

  Свойството Id връща целочислената стойност 1, която е идентификатор за IBlock.

  Свойството Tiles връща двуизмерен масив от обекти Position, които представят позициите на четирите квадрата, съставляващи IBlock, във всяка от четирите му ориентации. Всеки вътрешен масив представлява една ориентация на блока. Позициите са определени спрямо централния квадрат на блока, който е третият квадрат отляво в горния ред.

```
private readonly Position[][] tiles = new Position[][]
{
    new Position[] { new(1,0), new(1,1), new(1,2), new(1,3) },
    new Position[] { new(0,2), new(1,2), new(2,2), new(3,2) },
    new Position[] { new(2,0), new(2,1), new(2,2), new(2,3) },
    new Position[] { new(0,1), new(1,1), new(2,1), new(3,1) }
};
```

Фиг 28

Класът IBlock също така дефинира свое собствено поле tiles, което представлява масив от масиви от обекти Position. Всеки вътрешен масив представлява една от четирите възможни

ориентации на блока. Всеки обект Position представлява позицията на един квадрат спрямо централния квадрат на блока.

И накрая, класът IBlock пренаписва свойството StartOffset на класа Block, което връща обект Position, представляващ началната позиция на блока, когато той се създава в играта. В този случай StartOffset е зададено на new Position(-1, 3), което означава, че блокът ще бъде първоначално позициониран един ред над горната част на мрежата на играта и три колони надясно.

```
8 references
public override int Id => 1;
5 references
protected override Position StartOffset => new Position(-1, 3);
4 references
protected override Position[][] Tiles => tiles;
```

Фиг 29

- **JBlock.cs**

Този код дефинира клас, наречен JBlock, който разширява класа Block. JBlock представлява блок на Tetris във формата на буквата J.

Класът JBlock замества три свойства на класа Block: Id, StartOffset и Tiles.

Id: Това свойство връща цяло число, което представлява ID на JBlock. В този случай ID е 2.

```
8 references
public override int Id => 2;
```

Фиг 30

StartOffset: Това свойство връща обект Position, който представлява началната позиция на JBlock. В този случай началната позиция е (0, 3).

```
5 references
protected override Position StartOffset => new(0, 3);
```

Фиг 31

Position: Това свойство връща двуизмерен масив от обекти Position, които представляват позициите на четирите плочки, съставящи JBlock. Съществуват четири възможни завъртания на JBlock, така че масивът Tiles съдържа четири масива от по четири обекта Position. Всеки обект Position представлява ред и колона на плочка спрямо началната позиция на JBlock.

```
4 references
protected override Position[][] Tiles => new Position[][] {
    new Position[] {new(0, 0), new(1, 0), new(1, 1), new(1, 2)},
    new Position[] {new(0, 1), new(0, 2), new(1, 1), new(2, 1)},
    new Position[] {new(1, 0), new(1, 1), new(1, 2), new(2, 2)},
    new Position[] {new(0, 1), new(1, 1), new(2, 1), new(2, 0)}
};
```

Фиг 32

- **LBlock.cs**

Този код дефинира клас LBlock в Tetris, който представлява блок от тетраминота във формата на буквата L. Класът е производен на класа Block.

Класът LBlock получава следните свойства от класа Block:

Id: Връща ID на този блок. В този случай той връща стойността 3.

```
8 references
public override int Id => 3;
```

Фиг 33

StartOffset: Връща началната позиция на блока върху игралната дъска. В този случай тя връща стойността Position (0, 3).

```
5 references
protected override Position StartOffset => new(0, 3);
```

Фиг 34

Position: Връща масив от масиви, които представят формата на блока във всяко от четирите му възможни завъртания. Всеки подмасив съдържа четири обекта Position, които представят позициите на четирите квадрата, съставляващи блока в тази ротация. Позициите са относителни

спрямо StartOffset. В този случай свойството Tiles връща масив от четири подмасива, всеки от които съдържа четири обекта Position, които определят формата на блока L в различна ротация.

```
4 references
protected override Position[][] Tiles => new Position[][] {
    new Position[] {new(0,2), new(1,0), new(1,1), new(1,2)},
    new Position[] {new(0,1), new(1,1), new(2,1), new(2,2)},
    new Position[] {new(1,0), new(1,1), new(1,2), new(2,0)},
    new Position[] {new(0,0), new(0,1), new(1,1), new(2,1)}
};
```

Фиг 35

Като цяло този код определя формата на L-образен блок тетрамино в играта Tetris.

- **OBlock.cs**

Този код дефинира клас, наречен OBlock. Той наследява класа Block, който осигурява някои общи функционалности за всички типове блокове на Tetris.

```
public class OBlock : Block
```

Фиг 36

Класът OBlock има частно поле, наречено tiles (плочки), което представлява двуизмерен масив от обекти Position. Този масив представя формата на блока в различните му ориентации. В този случай има само една ориентация, която е квадрат 2x2.

```
private readonly Position[][] tiles = new Position[][]
{
    new Position[] { new(0,0), new(0,1), new(1,0), new(1,1) }
};
```

Фиг 37

Свойството Id връща целочислената стойност 4, която е идентификатор за този тип блок.

```
8 references
public override int Id => 4;
```

Фиг 38

Свойството StartOffset връща обект Position, представящ началната позиция на блока при първоначалното му създаване. В този случай StartOffset е зададено на (0, 4), което означава, че блокът ще започне в четвъртата колона на първия ред.

```
5 references
protected override Position StartOffset => new Position(0, 4);
```

Фиг 39

Свойството Tiles връща масива tiles, който представя формата на блока. Този масив ще бъде използван от класа Block, за да се нарисува блокът върху игралното поле.

```
4 references
protected override Position[][] Tiles => tiles;
```

Фиг 40

Като цяло, класът OBlock дефинира блок на Tetris, който е с форма на квадрат 2x2 и има идентификатор 4.

- **Position.cs**

Този код дефинира клас Position. Обектът Position представлява местоположение в двуизмерна мрежа или матрица и има две свойства Row и Column, които посочват съответно координатите на реда и колоната.

Конструкторът на класа Position приема два аргумента, row и column, които се използват за задаване на свойствата Row и Column на обекта.

```
99+ references
public class Position
{
    14 references
    public int Row { get; set; }
    14 references
    public int Column { get; set; }

    99+ references
    public Position(int row, int column)
    {
        Row = row;
        Column = column;
    }
}
```

Фиг 41

- **SBlock.cs**

  Този код дефинира клас, наречен SBlock. Класът SBlock наследява от класа Block, който е базовият клас за всички различни видове блокове в играта.

  Класът SBlock замества три свойства на класа Block: Id, StartOffset и Tiles.

  Свойството Id връща идентификатор на този тип блок, който в този случай е 5.

```
8 references
public override int Id => 5;
```

Фиг 42

Свойството StartOffset връща обект Position, който задава началните отмествания на редовете и колоните за блока. В този случай то е (0, 3), което означава, че блокът ще започне в първия ред и четвъртата колона на игралната мрежа.

```
5 references
protected override Position StartOffset => new(0, 3);
```

Фиг 43

Свойството Tiles връща двумерен масив от обекти Position, които представляват четирите плочки на блока във всяко от четирите му възможни завъртания. Всеки ред в масива представлява ротация, а всеки елемент в реда представлява позицията на плочка в тази ротация.

Другите редове на масива Tiles представят блока в другите му три завъртания, които се получават чрез трикратно завъртане на първото завъртане на 90 градуса по посока на часовниковата стрелка.

```
4 references
protected override Position[][] Tiles => new Position[][] {
    new Position[] { new(0,1), new(0,2), new(1,0), new(1,1) },
    new Position[] { new(0,1), new(1,1), new(1,2), new(2,2) },
    new Position[] { new(1,1), new(1,2), new(2,0), new(2,1) },
    new Position[] { new(0,0), new(1,0), new(1,1), new(2,1) }
};
```

Фиг 44

- **TBlock.cs**

Този код дефинира клас TBlock, който представлява блок на Tetris с формата на буквата T.

TBlock наследява класа Block.

Класът TBlock взима три свойства от класа Block:

Id: връща идентификаторът на този блок (който в този случай е 6).

```
8 references
public override int Id => 6;
```

Фиг 45

StartOffset: указва началната позиция на блока върху игралната дъска, когато той се създава. Позицията се определя като двойка (x, y), където x е индексът на реда, а y е индексът на колоната. В този случай блокът започва от ред 0, колона 3.

```
5 references
protected override Position StartOffset => new(0, 3);
```

Фиг 46

Position: определя позициите на четирите плочки, които съставляват блока, във всяка от четирите му възможни ориентации. Позициите се определят като двойки (x, y) спрямо точката на въртене на блока. Съществуват четири масива с позиции, всяка от които представлява една от четирите възможни ориентации на T-образния блок.

```
4 references
protected override Position[][] Tiles => new Position[][] {
    new Position[] {new(0,1), new(1,0), new(1,1), new(1,2)},
    new Position[] {new(0,1), new(1,1), new(1,2), new(2,1)},
    new Position[] {new(1,0), new(1,1), new(1,2), new(2,1)},
    new Position[] {new(0,1), new(1,0), new(1,1), new(2,1)}
};
```

Фиг 47

- **ZBlock.cs**

Този код дефинира клас ZBlock, който представлява Z-образен блок в играта Tetris. Той наследява класа Block и получава три от неговите свойства.

Id: връща цялото число 7, което идентифицира блока Z.

```
8 references
public override int Id => 7;
```

Фиг 48

StartOffset: връща обект Position, представляващ началната позиция на блока върху игралната дъска. В този случай блокът започва от ред 0 и колона 3.

```
5 references
protected override Position StartOffset => new(0, 3);
```

Фиг 49

Position: връща двуизмерен масив от обекти Position, представящи различните завъртания на блока. Всяко завъртане е представено чрез масив от четири обекта Position, по един за всяко квадратче в блока. Блокът Z има четири възможни завъртания, които са представени от четирите масива в свойството Tiles.

```
4 references
protected override Position[][] Tiles => new Position[][] {
    new Position[] {new(0,0), new(0,1), new(1,1), new(1,2)},
    new Position[] {new(0,2), new(1,1), new(1,2), new(2,1)},
    new Position[] {new(1,0), new(1,1), new(2,1), new(2,2)},
    new Position[] {new(0,1), new(1,0), new(1,1), new(2,0)}
};
```

Фиг 50

Класът ZBlock няма никакви допълнителни методи или полета извън тези, наследени от Block.

Има и 1 .xaml файл:

- **MainWindow**

  Това е кодът XAML за главния прозорец на играта Tetris.

  Елементът Window дефинира главния прозорец и задава различни атрибути, като например неговия размер, заглавие, минимален размер и стил на шрифта. Той също така

задава обработчик на събитие за събитието KeyDown, което се задейства винаги, когато се натисне клавиш, докато прозорецът има фокус.

```
Title="Tetris" Height="600" Width="800"
MinWidth="600" MinHeight="600"
Foreground="White"
FontFamily="Segoe UI Light" FontSize="28"
KeyDown="Window_KeyDown">
```

Фиг 51

Във вътрешността на елемента Grid са дефинирани няколко елемента, включително елемент Viewbox, който съдържа елемент Canvas, който представя областта на играта, в която падат тетрамината.

```
<Canvas x:Name="GameCanvas"
        Background="#101010"
        Width="250"
        Height="510"
        ClipToBounds="True"
        Loaded="GameCanvas_Loaded"/>
```

Фиг 52

Два елемента TextBlock се използват за показване на резултата на играча и на екрана за приключване на играта.

```
<TextBlock x:Name="FinalScoreText"
           Text="Score: "
           FontSize="36"
           TextAlignment="Center"/>
```

Фиг 53

Два елемента StackPanel се използват за показване на тетраминатa "Hold" (Задръж)

```
<StackPanel Grid.Row="1"
            Grid.Column="0"
            VerticalAlignment="Center"
            HorizontalAlignment="Right">
    <TextBlock Text="Hold"
               TextAlignment="Center"/>

    <Image x:Name="HoldImage"
           Margin="20"
           Width="125"/>
</StackPanel>
```

Фиг 54

и "Next" (Следващ),

```
<StackPanel Grid.Row="1"
            Grid.Column="2"
            VerticalAlignment="Center"
            HorizontalAlignment="Left">
    <TextBlock Text="Next"
               TextAlignment="Center"/>

    <Image x:Name="NextImage"
           Margin="20"
           Width="125"/>
</StackPanel>
```

Фиг 55

а елементът ImageBrush се използва за задаване на фона на игралната мрежа.

```
<Grid.Background>
    <ImageBrush ImageSource="Assets/Background.png"/>
</Grid.Background>
```

Фиг 56

Накрая е дефиниран елемент Grid с име GameOverMenu, за да се покаже екранът за край на играта, когато играчът загуби. Той съдържа StackPanel с голям текстов блок, показващ "Game Over" (Край на играта), текстов блок, показващ крайния резултат на играча, и бутон за започване на нова игра. Видимостта на тази мрежа по подразбиране е зададена на "Hidden" (Скрит) и се променя на "Visible" (Видим), когато играчът загуби играта.

```
<Grid x:Name="GameOverMenu"
      Background="#CC000000"
      Grid.RowSpan="2"
      Grid.ColumnSpan="3"
      Visibility="Hidden">
    <StackPanel HorizontalAlignment="Center"
                VerticalAlignment="Center">
        <TextBlock Text="Game Over"
                   FontSize="48"
                   TextAlignment="Center"/>

        <TextBlock x:Name="FinalScoreText"
                   Text="Score: "
                   FontSize="36"
                   TextAlignment="Center"/>

        <Button Content="Play Again"
                Background="LightGreen"
                Margin="0,20,0,0"
                Padding="5"
                Click="PlayAgain_Click"/>
    </StackPanel>
</Grid>
```

Фиг 57

Това е основният преглед на начина, по който протича една игра на Tetris. С напредването на играта изчистването на линиите и избягването на натрупването на блокове става все по-голямо предизвикателство. С практиката играчите могат да подобрят уменията си и да постигнат по-високи резултати.

Тестове

Играта е тествана във VisualStudio 2022, като когато се пусне от конзолата, играта върви перфектно. Единствени проблеми излизат със самите снимки, като трябваше да им се променя големината няколко пъти през процеса на създаване на кода.

Глава 3. Ръководство за потребителя (смени заглавието)

Когато потребителят пусне играта, тя започва директно.

1. Играта започва с празно игрално поле, където ще падат тетраминатa. Играчът получава първото тетрамино, което обикновено е прав блок, съставен от четири квадратчета.
2. Играчът може да премести тетраминото наляво или надясно, като използва клавишите със стрелки. Той може също така да завърти тетраминото по посока на часовниковата стрелка или обратно на нея, като натисне стрелката нагоре.
3. тетраминото пада равномерно към дъното на игралното поле. Играчът може да ускори спускането му, като натисне стрелката надолу.
4. Целта е да позиционирате тетраминото така, че да се вмести плътно в другите блокове на игралното поле. Когато хоризонталната линия е напълно запълнена с блокове, тя изчезва, а всички блокове над нея падат надолу, за да запълнят празното пространство. Играчът печели точки за всяка изчистена линия.
5. Играта продължава с нови тетромини, които падат с постоянно нарастваща скорост. Играчът трябва да продължи да изчиства линиите и да не позволява на блоковете да се натрупват до върха на игралното поле. Ако блоковете достигнат върха, играта приключва.
6. Играта продължава, докато играчът не загуби, като позволи на блоковете да достигнат върха на игралното поле, или докато сам не реши да прекрати играта.
7. Задържането на блок в играта се извършва с използването на бутона “c”. С повторно използване на бутона “c”, играчът може да използва запазеният блок, вместо блокът, който е на полето.
8. Играчът може да види своят резултат докато играе играта, или като загуби.
9. Когато играчът загуби, той може да започне наново играта чрез бутонът “play again”, като може и да си види резултата.

Заключение

В заключение, Tetris е класическа видеоигра, която е завладяла сърцата на милиони играчи по света. Нейната проста, но предизвикателна механика на манипулиране на падащи тетрамино, за да се създадат хоризонтални линии без пропуски, в комбинация с пристрастяващия ѝ геймплей, я превръща в една от най-обичаните и трайни видеоигри на всички времена. Нейната трайна популярност десетилетия след първоначалното ѝ пускане е доказателство за привлекателността на играта и нейния емблематичен статут в света на игрите.

Информационни източници

Въведение в програмирането със C# - Светлин Наков

Tetris - Britannica.com

Microsoft - Create your first WPF Application in Visual Studio 2019

Рецензия на дипломен проект

Тема на дипломния проект			
Ученик			
Клас			
Професия			
Специалност			
Ръководител- консултант			
Рецензент			
Критерии за допускане до защита на дипломен проект		**Да**	**Не**
Съответствие на съдържанието и точките от заданието			
Съответствие между тема и съдържание			
Спазване на препоръчителния обем на дипломния проект			
Спазване на изискванията за оформление на дипломния проект			
Готовност за защита на дипломния проект			
Силни страни на дипломния проект			
Допуснати основни слабости			
Въпроси и препоръки към дипломния проект			

ЗАКЛЮЧЕНИЕ:

Качествата на дипломния проект дават основание ученикът/ ученичката ..... да бъде допуснат/а до защита пред членовете на комисията за подготовка, провеждане и оценяване на изпит чрез защита на дипломен проект- част по теория на професията.

.....05.2023г. Рецензент:.....

град София