Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 6 з дисципліни
«Проектування алгоритмів»

„Пошук в умовах протидії, ігри з повною інформацією, ігри з елементом випадковості, ігри з неповною інформацією”

Виконав(ла) *ІП-15 Ткач Владислав Анатолійович*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе І. Е.*
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи - вивчити основні підходи до формалізації алгоритмів знаходження рішень задач в умовах протидії. Ознайомитися з підходами до програмування алгоритмів штучного інтелекту в іграх з повною інформацією, іграх з елементами випадковості та в іграх з неповною інформацією.

2 ЗАВДАННЯ

Для ігор з повної інформацією, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм альфа-бета-відсікань. Реалізувати три рівні складності (легкий, середній, складний).

Для ігор з елементами випадковості, згідно варіанту (таблиця 2.1) реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Для реалізації стратегії гри комп'ютерного опонента використовувати алгоритм мінімакс.

Для карткових ігор, згідно варіанту (таблиця 2.1), реалізувати візуальний ігровий додаток, з користувацьким інтерфейсом, не консольним, для гри користувача з комп'ютерним опонентом. Потрібно реалізувати стратегію комп'ютерного опонента, і звести гру до гри з повною інформацією (див. Лекцію), далі реалізувати стратегію гри комп'ютерного опонента за допомогою алгоритму мінімаксу або альфа-бета-відсікань.

Реалізувати анімацію процесу жеребкування (+1 бал) або реалізувати анімацію ігрових процесів (роздачі карт, анімацію ходів тощо) (+1 бал).

Реалізувати варто тільки одне з бонусних завдань.

Зробити узагальнений висновок лабораторної роботи.

Таблиця 2.1 – Варіанти

№	**Варіант**	**Тип гри**
1	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
2	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
3	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості

4	Нейтріко http://www.iggamecenter.com/info/ru/neutreeko.html	З повною інформацією
5	Тринадцять http://www.rules.net.ru/kost.php?id=16	З елементами випадковості
6	Індійські кості http://www.rules.net.ru/kost.php?id=9	З елементами випадковості
7	Dots and Boxes https://ru.wikipedia.org/wiki/Палочки_(игра)	З повною інформацією
8	Двадцять одне http://gamerules.ru/igry-v-kosti-part8#dvadtsat-odno	З елементами випадковості
9	Тіко http://www.iggamecenter.com/info/ru/teeko.html	З повною інформацією
10	Клоббер http://www.iggamecenter.com/info/ru/clobber.html	З повною інформацією
11	101 https://www.durbetsel.ru/2_101.htm	Карткові ігри
12	Hackenbush http://www.papg.com/show?1TMP	З повною інформацією
13	Табу https://www.durbetsel.ru/2_taboo.htm	Карткові ігри
14	Заєць і Вовки (за Зайця) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією
15	Свої козирі https://www.durbetsel.ru/2_svoi-koziri.htm	Карткові ігри
16	Війна з ботами https://www.durbetsel.ru/2_voina_s_botami.htm	Карткові ігри
17	Domineering 8x8 http://www.papg.com/show?1TX6	З повною інформацією
18	Останній гравець https://www.durbetsel.ru/2_posledny_igrok.htm	Карткові ігри
19	Заєць и Вовки (за Вовків) http://www.iggamecenter.com/info/ru/foxh.html	З повною інформацією

20	Богач https://www.durbetsel.ru/2_bogach.htm	Карткові ігри
21	Редуду https://www.durbetsel.ru/2_redudu.htm	Карткові ігри
22	Эльферн https://www.durbetsel.ru/2_elfern.htm	Карткові ігри
23	Ремінь https://www.durbetsel.ru/2_remen.htm	Карткові ігри
24	Реверсі https://ru.wikipedia.org/wiki/Реверси	З повною інформацією
25	Вари http://www.iggamecenter.com/info/ru/oware.html	З повною інформацією
26	Яцзи https://game-wiki.guru/published/igryi/yaczzyi.html	З елементами випадковості
27	Лудо http://www.iggamecenter.com/info/ru/ludo.html	З елементами випадковості
28	Генерал http://www.rules.net.ru/kost.php?id=7	З елементами випадковості
29	Сим https://ru.wikipedia.org/wiki/Сим_(игра)	З повною інформацією
30	Col http://www.papg.com/show?2XLY	З повною інформацією
31	Snort http://www.papg.com/show?2XM1	З повною інформацією
32	Chomp http://www.papg.com/show?3AEA	З повною інформацією
33	Gale http://www.papg.com/show?1TPI	З повною інформацією
34	3D Noughts and Crosses 4 x 4 x 4 http://www.papg.com/show?1TND	З повною інформацією
35	Snakes http://www.papg.com/show?3AE4	З повною інформацією

3 ВИКОНАННЯ

3.1 Програмна реалізація алгоритму

3.1.1 Вихідний код

```
public Point? nextMove(int[,] map)
{
    int lostMove = Culculator.LostMove(map);
    if (lostMove % 2 == 1)
    {
        lostMove--;
    }

    int[,] newMap = Copy.Map(map);

    State state = Analysis(newMap, Color, Math.Min(lostMove, Complexity));

    if (state == null)
    {
        Point? lastChanse = Next(map, Color);
        if (lastChanse == null)
            return null;
        return lastChanse;
    }

    return state.Point;
}

private State Analysis(int[,] map, int currentPlayer, int deep)
{
    deep--;
    List<Point> points = PossibleMoveFinder.Find(map, currentPlayer);
    int? max = null;
    int maxIndex = 0;

    for (int i = 0; i < points.Count; i++)
    {
        int[,] newMap = Copy.Map(map);
        newMap[points[i].X, points[i].Y] = currentPlayer;
        MoveMaker.newMove(newMap, currentPlayer, points[i].X, points[i].Y);
        State newMin = FindMin(newMap, currentPlayer == 1 ? 2 : 1, max, deep);
        if (newMin != null)
        {
            if (max == null)
            {
                max = newMin.Score;
            }
            else if (max < newMin.Score)
            {
                max = newMin.Score;
                maxIndex = i;
            }
        }
    }

    if (max == null)
        return null;
    return new State(points[maxIndex], max);
}
```

```
private State FindMin(int[,] map, int currentPlayer, int? max, int deep)
{
    deep--;
    if (deep == 0)
    {
        int localMin = int.MaxValue;
        int minIndex = 0;
        int minScore = 0;
        List<Point> points = PossibleMoveFinder.Find(map, currentPlayer);
        for (int i = 0; i < points.Count; i++)
        {
            int[,] newMap = Copy.Map(map);
            newMap[points[i].X, points[i].Y] = currentPlayer;
            MoveMaker.newMove(newMap, currentPlayer, points[i].X, points[i].Y);
            int score = Culculator.Score(newMap);

            if (score < localMin)
            {
                localMin = score;
                minIndex = i;
                minScore = score;
            }

            if (localMin <= max)
                return new State(points[i], score);
        }

        if (localMin == int.MaxValue)
            return null;

        return new State(points[minIndex], minScore);
    }
    else
    {
        int localMin = int.MaxValue;
        State minState = null;
        List<Point> points = PossibleMoveFinder.Find(map, currentPlayer);
        foreach (var t in points)
        {
            int[,] newMap = Copy.Map(map);
            newMap[t.X, t.Y] = currentPlayer;
            MoveMaker.newMove(newMap, currentPlayer, t.X, t.Y);
            State stata = Analysis(newMap, currentPlayer == 1 ? 2 : 1, deep);
            if (stata != null)
            {
                if (stata.Score < localMin)
                {
                    localMin = (int)stata.Score;
                    minState = stata;
                }
            }
        }

        return minState;
    }
}

private Point? Next(int[,] map, int currentPlayer)
{
    int max = Int32.MinValue;
    int maxId = 0;
    List<Point> points = PossibleMoveFinder.Find(map, currentPlayer);
    for (int i = 0; i < points.Count; i++)
```

```
    {
        int[,] newMap = Copy.Map(map);
        newMap[points[i].X, points[i].Y] = currentPlayer;
        MoveMaker.newMove(newMap, currentPlayer, points[i].X, points[i].Y);
        int score = Culculator.Score(newMap);
        if (score > max)
        {
            max = score;
            maxId = i;
        }
    }

    if (points.Count == 0)
        return null;

    return points[maxId];
}
```

3.1.2 Приклади роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми.

Рисунок 3.1 – Вибір рівня складності(можна змінювати перед першим ходом)

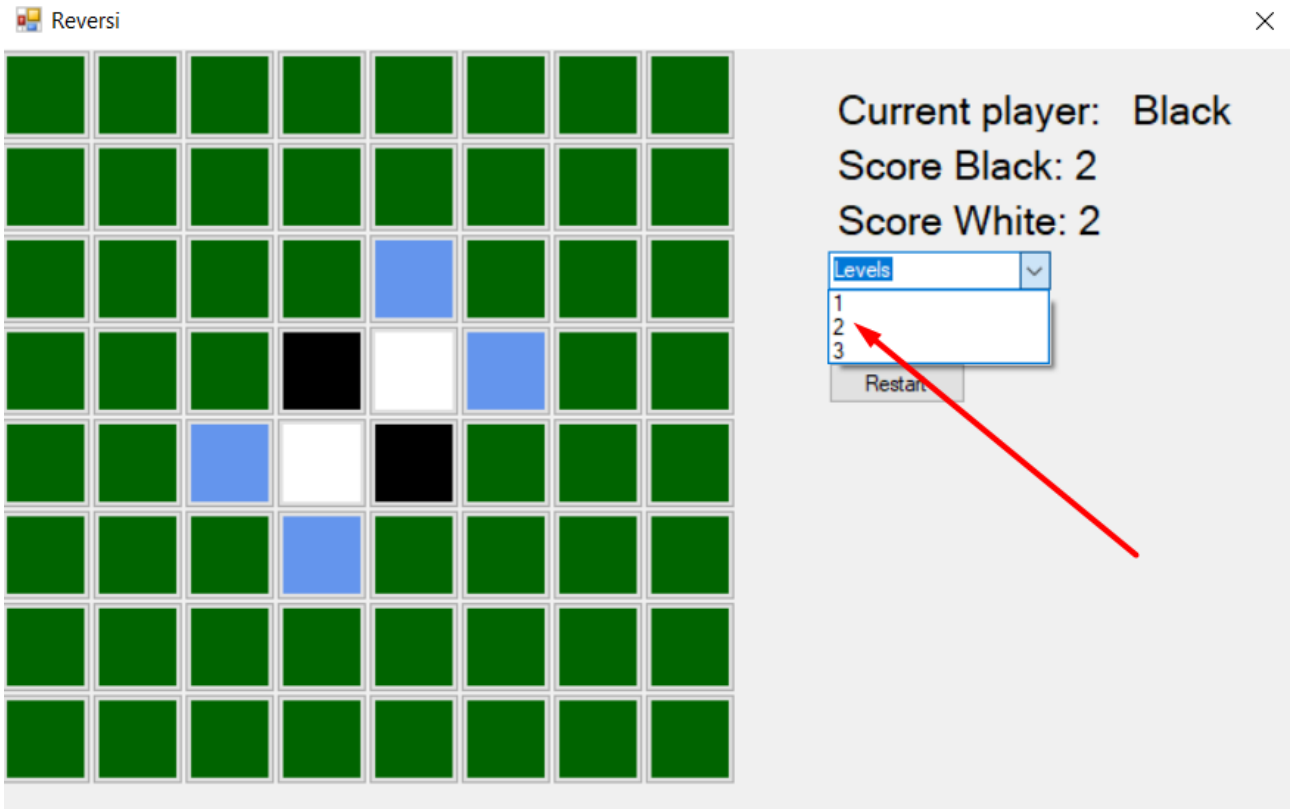
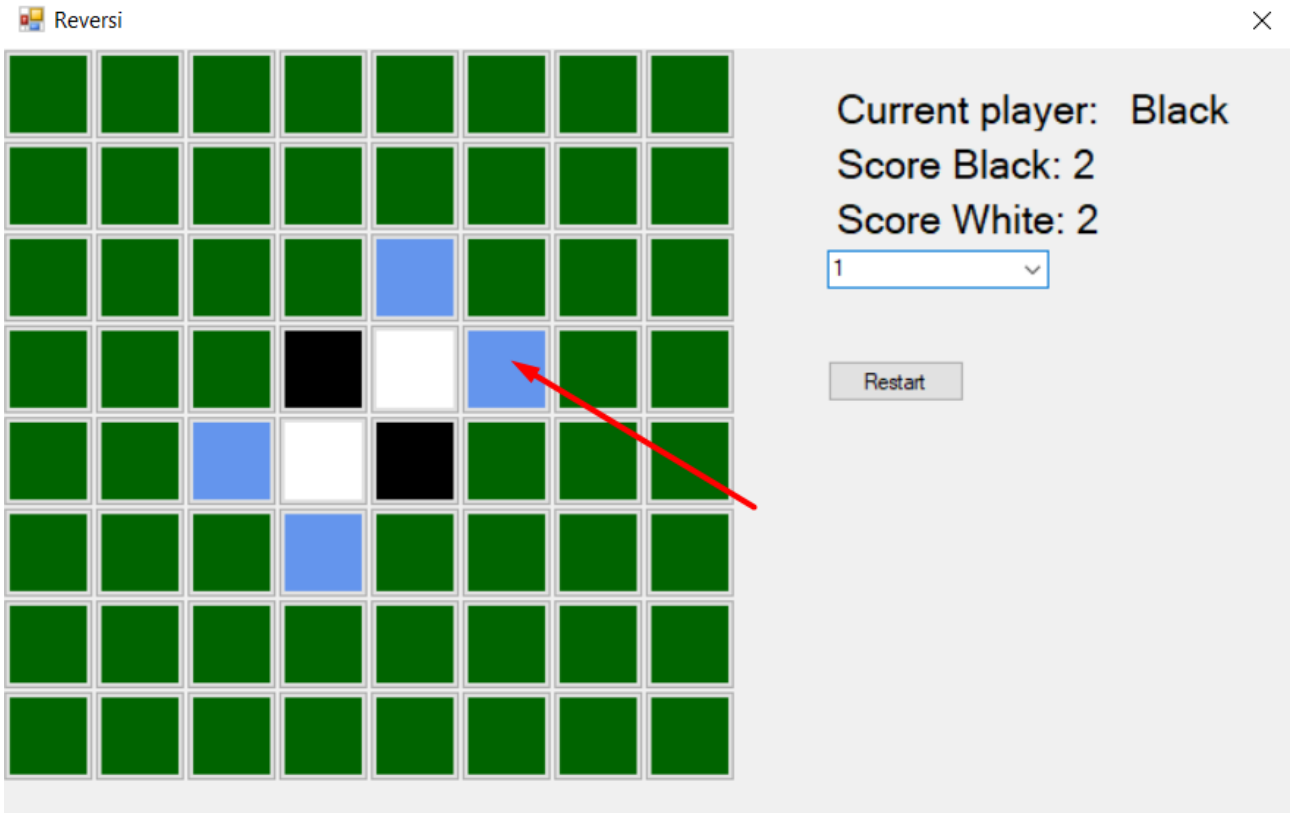


Рисунок 3.2 – хід користувача(синім підсвідчуються можливі ходи)



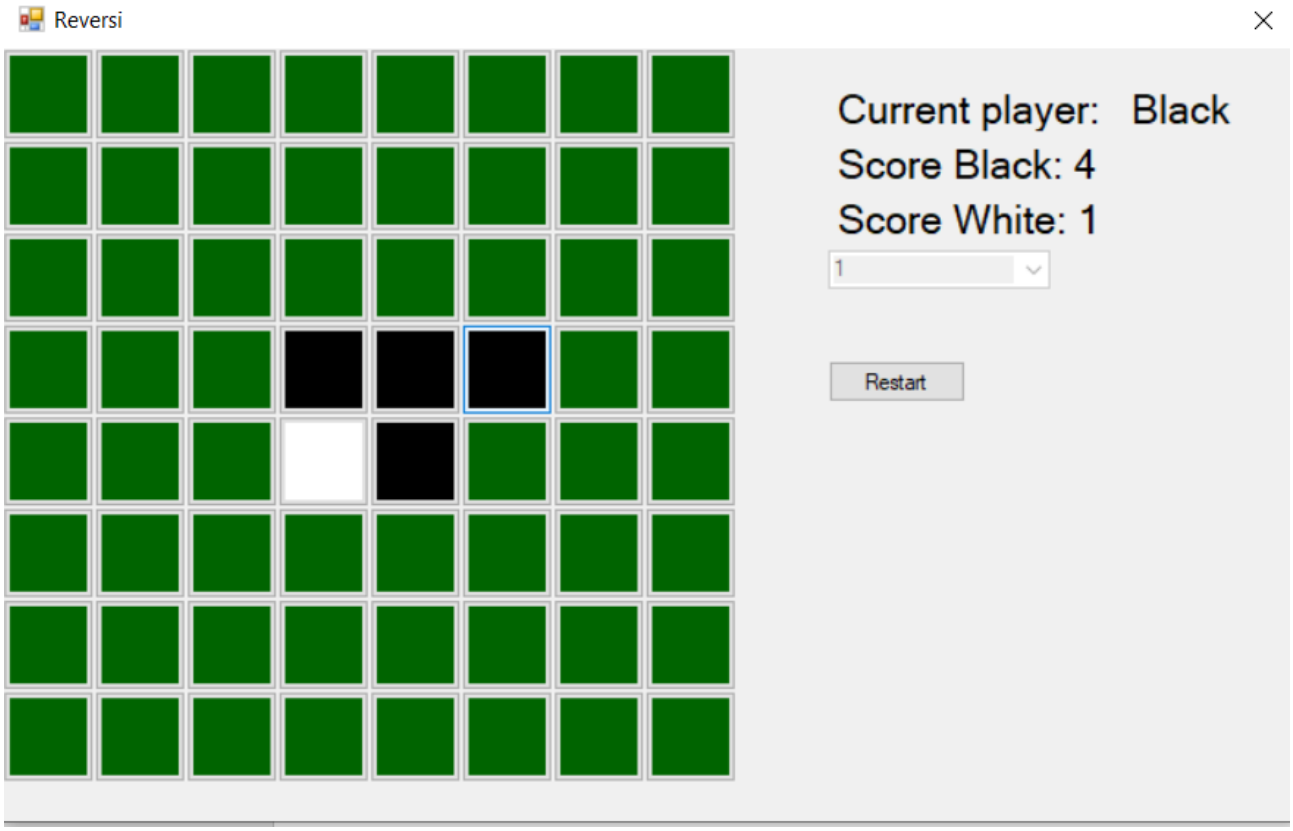
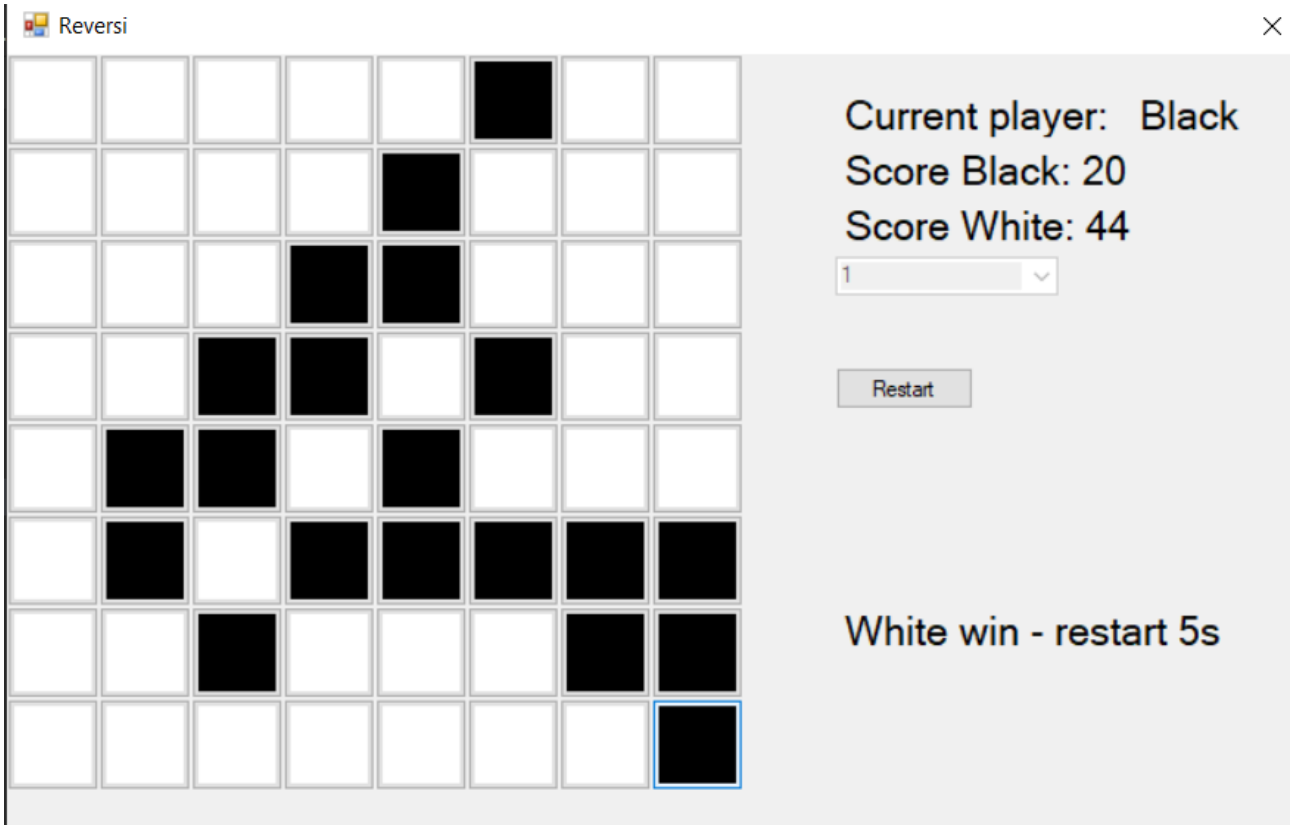


Рисунок 3.3 – хід комп’ютера(червоним підсвідчуються хід, який обрав комп'ютер)

Рисунок 3.4 – Завершення гри



ВИСНОВОК

В рамках даної лабораторної роботи було розглянути алгоритм альфа-бета відсікань на основі гри “Реверісі”, у ході чого було розроблено додаток із графічним інтерфейсом для цієї гри із суперником у ролі комп’ютера, який має 3 доступні рівні складності, складність характеризується глибиною аналізу станів(наприклад для рівня складності 1 – глибина 2), у перспективі можна вказати і більше рівнів складності – обмеження тільки на пам’ять та час. Даний алгоритм аналогічний Min Max тільки алгоритм альфа-бета відсікань відрізає уже не потрібні вітки що дає виграш і часі та пам’яті.

Проаналізувавши алгоритм та рівні складності зауважу, що комп’ютер дійсно важче перемогти із збільшенням рівня складності, але враховуючи той факт що алгоритм не аналізує ситуації коли конкретний гравець не має ходів і фактично інший може зробити більше одного ходу підряд, що призведе до суттєвого відриву, можна зробити висновок, що оптимальною стратегією користувача є саме цей варіант розвитку подій, і щоб його досягти, варто зміщувати власні фішки до однієї із стороні або кута, мінімізуючи концентрацію ворожого кольору на стороні користувача, тобто варто розділитись між кольорами уникаючи варіантів пересікання.

КРИТЕРІЇ ОЦІНЮВАННЯ

При здачі лабораторної роботи до 25.12.2022 включно максимальний бал дорівнює – 5. Після 25.12.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

– програмна реалізація – 95%;

– висновок – 5%.

+1 додатковий бал можна отримати за реалізацію анімації ігрових процесів (жеребкування, роздачі карт, анімацію ходів тощо).