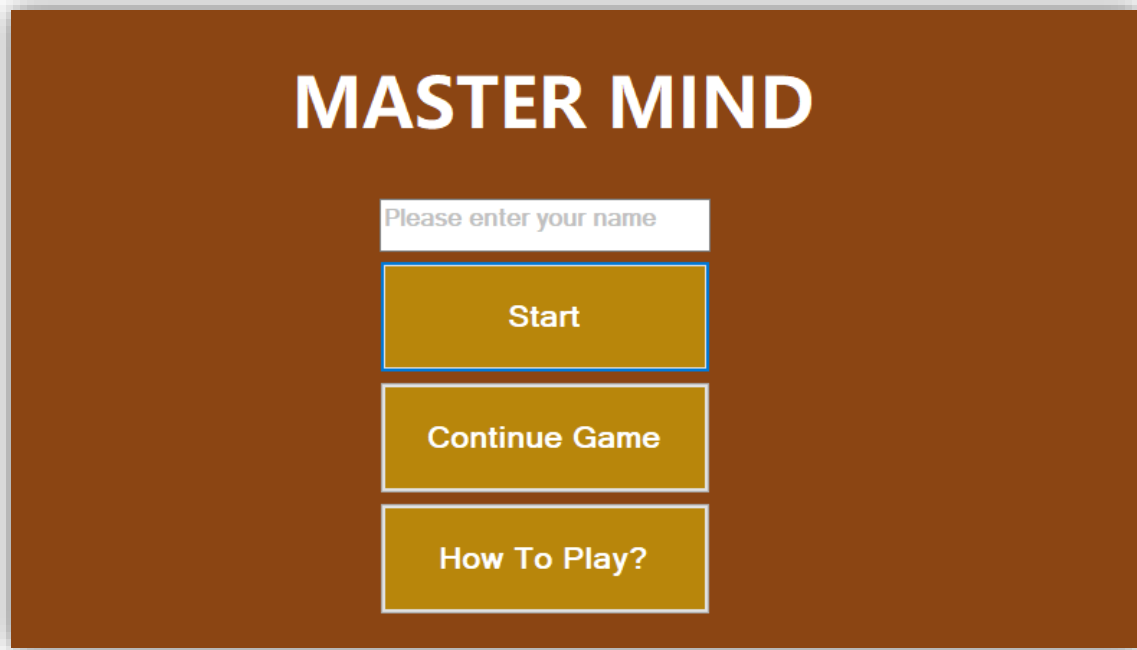


MASTERMIND



Programmering B

Projektet er udarbejdet af: Jonas Johan Kjæreby Kühn

Vejlederens navn: Jan Boddum Larsen

H. C. Ørsted Ballerup - 3.EI

Afleveringsdato: 09-04-2021

Antal anslag: 19404 tegn

Indholdsfortegnelse

Abstract	3
Problemformulering	3
Funktionsbeskrivelse	4
Skærmlayout.....	4
Struktur.....	6
Dokumentation af selve programmet	7
Variabler	7
Array2D + Array	7
Klasser.....	7
Metoder.....	10
Metode til at tjekke spillerens gæt.....	10
Metode til at tegne spilleren bud på farvekombination	13
Metode til at tegne svar-firkanter	13
Metode til at genoptage sit gemte spil	15
Test af programmet.....	17
Konklusion	21
Litteraturliste:.....	21
Bilag	22
Bilag 1 – Startside	22
Bilag 2 – HowToPlayForm	22
Bilag 3 – GameForm før start	23
Bilag 4 – GameForm under spillet	24
Bilag 5 – Kodens til DrawMasterMind()	25
Bilag 6 – Kodens til DrawCheck()	26
Bilag 7 – Kodens til CountCheck-klasse.....	28
Bilag 8 – Resterende kode:	29
Bilag 8.1 – Startside.cs	29
Bilag 8.2 - HowToPlayForm.cs	30
Bilag 8.3 – GameForm.cs	31
Bilag 8.4 – NumberInArrayClass.cs	42
Bilag 8.5 – UndoArrayClass.cs.....	43
Bilag 8.6 – CountCheck.cs - klasse	44

Abstract

Formålet er at lave et simpelt spil med brugerinteraktion via mus, hvor jeg gerne vil lave en applikation af spillet "MasterMind". Mit "MasterMind" er et simpelt spil, hvor man bl.a. spiller mod computeren, og spilleren kan selv starte og afslutte spillet ved hjælp af informationer, der bliver vist på skærmen.

Udviklingen af selve programmet er baseret på en teknik kaldes "Stepwise Improvement" og programmet bliver skabt på "windows form" og bliver skrevet i 'C#' på "Visual studio".

Jeg har igennem projektet udarbejdet metoder og klasser for at kunne udarbejde et simpelt spil, så selv mine forældre kan spille spillet ud fra retningslinjer og forstå lidt af koden bag programmet. Jeg kan derfor konkludere at projektet har været vellykket, da spillet er både teknisk interessant og underholdende.

Problemformulering

Dette projekt går ud på at lave et simpelt spil med brugerinteraktion via mus. Jeg har valgt at lave en applikation af spillet "MasterMind", hvor man normalt spiller mod hinanden. Men i dette projekt vil jeg lave en version af MasterMind, hvor man spiller mod computeren. Projektet skal udarbejdes i Visual studio og skrives i C#. Spillet "MasterMind" går ud på at computeren genererer en vilkårlig farvekombination, som spilleren ikke kan se. Spilleren skal så komme med et bud på en farvekombination, hvor computeren vil melde tilbage om, hvor korrekt buddet er med to forskellige farver, hvid og sort.

Formålet med projektet er at spilleren skal kunne starte, gemme, genoptage og afslutte spillet. Spilleren skal kunne forstå, hvordan man spiller spillet, og derudover kan spilleren konkurrere mod sine venner om at være hurtigst, for der vil være en highscore sorteret efter tid. Projektet bliver løst vha. teknikken "Stepwise Improvement", hvor man hele tester sit program under udviklingen.

Funktionsbeskrivelse

Skærmlayout

Mit program er opbygget af tre forskellige forms; Startside (Form1), HowToPlayForm og GameForm. Når man starter spillet, er startside den første side/form, man møder. Startside kan ses i [bilag 1](#), hvor jeg har lavet 3 knapper og en tekstboks. Tekstboksen er tiltænkt spillerens navn.

På de tre knapper er der en forbindelse til en ny form. Klikker på den nederste knap (How to play), åbnes en ny form 'HowToPlayForm', hvor jeg har udarbejdet en tekst om hvordan man spiller MasterMind. (se [bilag 2](#)) Er man færdig med at læse, klikkes man bare på exit-knappen, for at vende tilbage til startside. Bag exit-knappen benyttes programmet "*this.Close()*", hvor man lukker formen ned.

Ønsker spilleren at starte spillet, trykker man på start-knappen, så åbnes GameForm, se [bilag 3](#). Hvis spilleren ikke har skrevet sit navn, poppes en Message Box op, for at påminde spilleren om at indtaste navn. Navnet bliver sendt videre til GameForm.

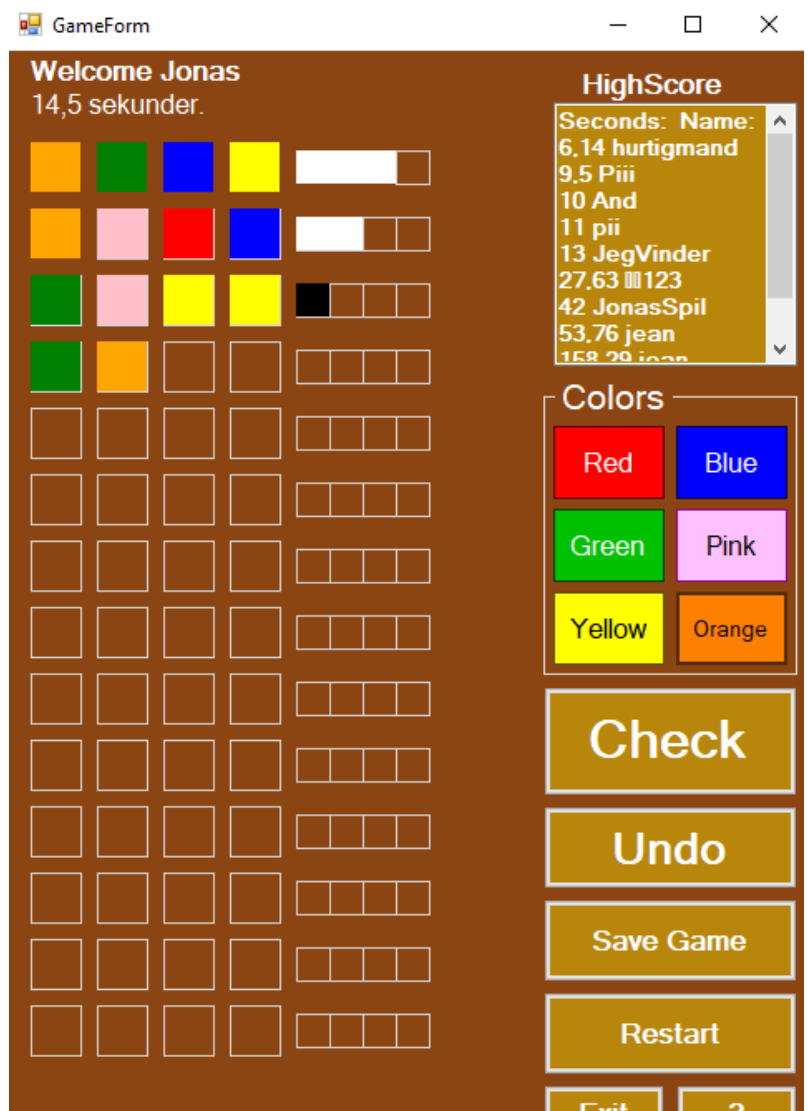
GameForm

Kommer man ind på GameForm, se figur 1, kan man se en velkomst-tekst øverst oppe, hvor der står "Welcome {spillerens navn}", som er hentet fra startside. Under velkomst-teksten ses der et stopur, som bliver aktiveret når man starter spillet. Til højre kan man se en highscore, som bliver printet direkte fra en highscore-fil.

Ønsker spilleren at starte spillet, klikker spilleren på Start-knappen. Så forsvinder start-knappen og resten af spillet bliver synliggjort, bl.a. knapperne med farverne, som man skal trykke på for at vælge sine bud på en farvekombination.

Hvis spilleren ønsker at genoptage sit gemte spil, skal spilleren trykke på "Continue Game-knap" på startside. Så åbnes stifinder op, hvor man skal vælge den fil, der indeholder det gemte spil.

Når spilleren har lavet 4 bud på en farvekombination til MasterMind, trykkes på "Check"-knap, hvor efter der bliver tjekket for den række, spilleren er i gang med. Svarene printes til højre for de farvede felter. Sorte firkanter printes først, og derefter hvid.



Figur 1 - Et skærbillede af GameForm under spillet

Undo - knap

Hvis spilleren fortryder et træk, så kan man trykke på undo-knap, så der fjernes et træk på den igangværende række. Hvis spilleren er på en række, hvor spilleren ikke har nogen gæt, så sker der ikke noget, når spilleren trykker på undo. Undo fjerner kun det seneste gæt på rækken.

SaveGame - knap

Hvis spilleren ønsker at gemme sit spil til senere, kan der trykkes på *"Save Game-knap"*, hvor efter der åbnes en stifinder op, hvor man så skal trykke på *"Gem"*. Derefter bliver spillet gemt på en fil, og GameForm lukkes ned.

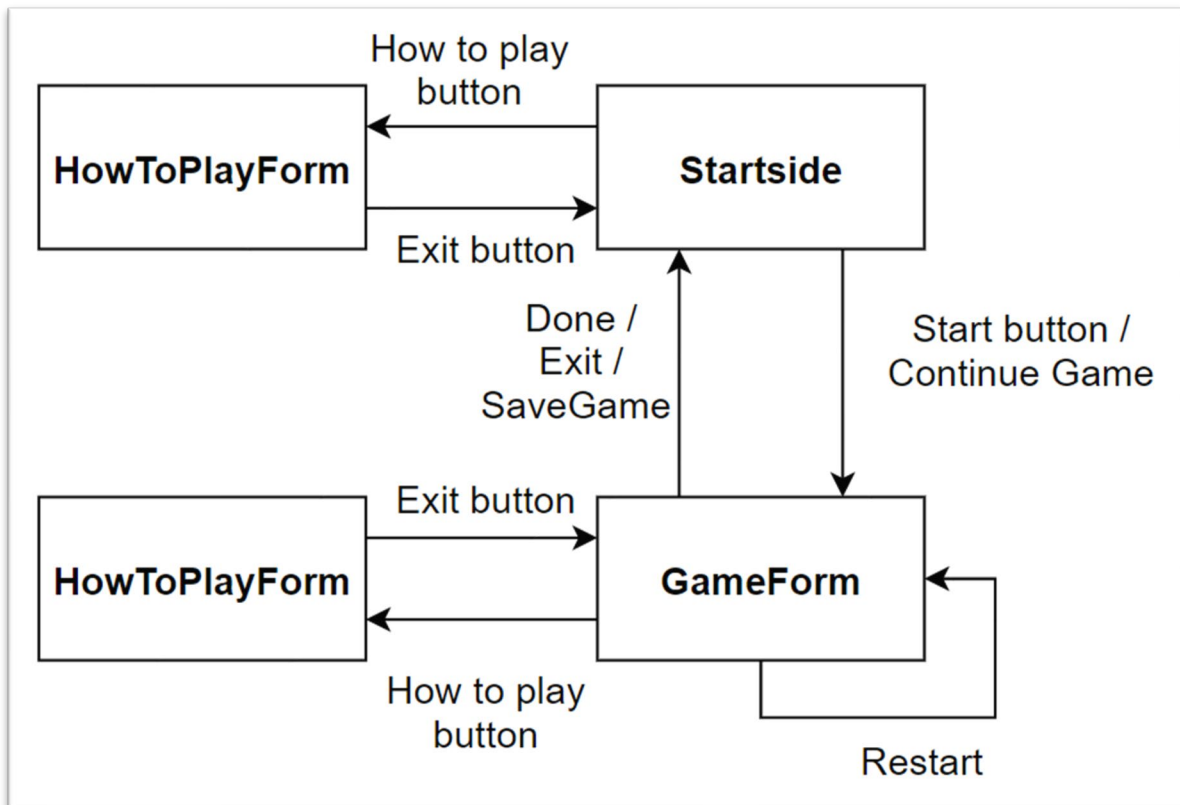
Restart, "?" og exit knap

Hvis spilleren ønsker at starte helt forfra med et nyt spil, trykkes der på *"restart-knap"*. Bliver spilleren usikker på, hvordan man spiller Mastermind, kan der trykkes på How to play-knap (?). Herved åbnes en HowToPlayform op. Programmet er designet så tiden ikke stopper ved tryk på howtoplay(?) -knap, så spilleren ikke kan "snyde" ved at få en tænkepause ved at trykke på "?" -knap. Ønsker spilleren at stoppe spillet, trykkes på Exit-knappen, hvor efter GameForm lukkes ned.

Struktur

Nedenunder er der en skitse over strukturen i programmets forms, se figur 2.

Som man kan se, at der er en forbindelse mellem alle tre forms programmet har. Der er en forbindelse mellem HowToPlayForm og startside. Hvis spilleren tilgår HowToPlay igennem GameForm, vender man tilbage til GameForm, når man trykker på Exit-knappen på HowToPlayForm.



Figur 2 - En illustration, der beskriver strukturen af forms

Dokumentation af selve programmet

Variabler

Array2D + Array

De vigtigste variabler i mit program er et todimensionelt array, hvor jeg blandt andet har brugt et int todimensionelt array; *"mmArray2D[14, 4];"* til at opbevare spillerens bud på den rigtige farvekombination. Jeg har valgt at oprette 14 rækker, med 4 kolonner. Dette vil sige, at spilleren har 4 forskellige gæt på en række og spilleren har 14 rækker til at gætte farvekombinationen som computeren har genereret. Derudover har jeg et normalt array med 4 elementer; *"gMasterMind"*, som bruges til at opbevare værdier fra farvekombinationen, som er udarbejdet af computeren.

Klasser

I mit program har jeg lavet tre klasser. Jeg vil komme ind på de to klasser, der har betydning for spillerens bud på en farvekombination.

NumberInArrayClass

Den første klasse indeholder en metode; *"UpdateArr"*, som skal have 3 parametrene; et todimensionelt array, et tal og tallet for den igangværende række, se figur 3.

Metoden gennemgår en for-løkke, som leder efter et 0-tal i den igangværende række. Forløkken bliver gentaget efter *"Get-length(1)"*, der angiver antallet af kolonner i den valgte array. Hvis et 0-tal er fundet, bliver 0-tallet erstattet med det tal, som spilleren ønsker at ændre, med parameteren *"num"*. Til sidst bliver arrayet returneret.

```

2 references
class NumberInArrayClass
{
    1 reference
    public int[,] UpdateArr(int[,] arr, int num, int rownum)
    {
        //Makes one for loops looking for a 0 number and
        //replaces this number with the selected number (num)
        for (int j = 0; j < arr.GetLength(1); j++)
        {
            if (arr[rownum, j] == 0)
            {
                //We replace 0 with the number (num)
                arr[rownum, j] = num;
                //break for a change has been made
                break;
            }
        }
        //Return the array
        return arr;
    }
}

```

Figur 3 - Et skærbillede af NumberInArrayClass

Med denne klasse kan programmet hele tiden opdatere dette todimensionelt array, som indeholder værdier for spillerens bud. På figur 4 ses der, at der bliver skabt en ny instans af klassen igennem metoden *"GamePlay()"*. Derefter bliver der tegnet MasterMind, så spilleren hele tiden kan se deres bud visuelt på GameForm.

```

private void GamePlay(int num101)
{
    //Creates a new instance of this class
    NumberInArrayClass UpdataArr = new NumberInArrayClass();
    mmArray2D = UpdataArr.UpdateArr(mmArray2D, num101, numRows);

    //Draw MasterMind
    DrawMasterMind();
}

```

Figur 4 - Et skærbillede af metoden GamePlay()

UndoArrayClass

Den anden klasse er også nyttig for spillet for det er en såkaldte "fortryde-klasse". Klassen indeholder også en metode; "UndoArr", der skal have to parametre; et todimensionelt array og tallet for den igangværende række, se figur 5.

Metoden starter med at tjekke, om der ikke er tale om det første indeks i rækken ved hjælp af en "if-statement". Dette sikrer, at spilleren kun kan fortryde trækket på den igangværende række.

Derefter bliver der brugt en for-løkke, som leder efter et 0-tal, og starter allerede ved det 1. indeks, fordi et array med 4 elementer lyder indekser således; 0,1,2,3. For-løkken starter ved indeks 1, for at minimere for-løkkens antal af gennemløb, da jeg tidligere har lavet en "if-statement", der tjekker at den første indeks ikke er 0.

For at lede efter 0-tal benytter programmet et "if-statement", som tjekker om tallet er 0. Hvis tallet er 0, erstattes et tidligere indeks med 0 (Vi går en indeks tilbage, -1), hvilket kan ses, at der bliver skrevet "i - 1". Derefter bliver bool omdannet til "false", og så "breaker" vi for-løkken, for nu har programmet lavet en ændring i arrayet.

Hvis for-løkken er kørt igennem uden en ændring, er bool dims stadig true.

Dette kan vi benytte til at fjerne det sidste indeks. Efter for-løkken er der en if, der tjekker om dims er true. Hvis det er true, omdanner programmet automatisk det sidste indeks om til 0-tal.

Dette gør vi for at sikre os, at spilleren også kan fortryde deres træk på den 3. indeks i et array med 4 elementer. For for-løkken stopper ved 3. indeks, men hvis spilleren har et tal på den 3. indeks, så kan programmet ikke finde et 0-tal. Til sidst bliver arrayet returneret:

Med metoden kan jeg skabe en ny instans, hvor metoden erstatter det seneste tal om til et 0-tal. Se figur 6 nedenfor. Jeg benytter klassen til undo-knappen, hvor der startes med at 'rense' canvas, og så bliver arrayet opdateret ved hjælp af metoden. Derefter bliver både MasterMind og 'Checkliste' tegnet.

```
class UndoArrayClass
{
    1 reference
    public int[,] UndoArr(int[,] arr, int rownum)
    {
        //Creates a bool variable that checks if you have
        //been through a course without a change
        bool dims = true;

        //Checking if it is not the first index in the row.
        //This ensures that you can only remove numbers
        //within the range you are working on
        if (arr[rownum, 0] != 0)
        {
            //makes forloop that looks for a 0-number
            //GetLength(1) is the number of columns
            //the forloop starts already at the 1st index,
            //since i is 1
            for (int i = 1; i < arr.GetLength(1); i++)
            {
                //if there is a 0, then we go back an index
                //and replace the number with 0
                if (arr[rownum, i] == 0)
                {
                    //We replace the previous index with 0
                    arr[rownum, i - 1] = 0;
                    //bool is false now, for forloop has been used
                    dims = false;
                    //breaker the course,
                    //for we have finished our task
                    break;
                }
            }
            //if dims is true, then we remove we automatically
            //convert the last index(3) to 0
            if (dims)
            {
                arr[rownum, arr.GetLength(1) - 1] = 0;
            }
        }
        //Return array
        return arr;
    }
}
```

Figur 5 - Et skærbillede af UndearrayClass


```
private void btnUndo_Click(object sender, EventArgs e)
{
    //Clear my panel (canvas1)
    canvas1.Refresh();
    //Creates a new instance of this class and convert
    //the latest number to 0
    UndoArrayClass undoarr = new UndoArrayClass();
    mmArray2D = undoarr.UndoArr(mmArray2D, numRowsArr);

    //Draw MasterMind
    DrawMasterMind();
    //Draw Check
    DrawCheck();
}
```

Figur 6 - Et skærbillede af koden bag undo-knappen

Metoder

Metode til at tjekke spillerens gæt

Når spilleren ønsker at tjekke sit bud, tjekker programmet først, om spilleren har valgt 1 fuld farvekombination på 4 elementer på rækken. Dette tjekker programmet igennem et "if-statement", hvor programmet tjekker, at der ikke indeholdes et 0-tal i den igangværende række.

Hvis spilleren har lavet en farvekombination, bliver metoden *CheckArr()* indkaldt. Metoden kræver en parameter, som er tallet for den igangværende række.

Metoden er delt op i to for-løkker, hvor den første tjekker for de rigtige farver med de rigtige placeringer (sort) og den anden tjekker for de rigtige farver, men med de forkerte placeringer (hvide).

Før programmet går i gang med de to for-løkker, bliver der skabt en bool array variabel med 4 elementer, der bliver brugt til at tjekke at tallet ikke bliver genbrugt.

Den ene array er *mmArray2D* indeholder værdier fra dit gæt, mens *gMasterMind* indeholder de 4 elementer, som spilleren skal gætte.

Sort og hvid

For at finde de sorte, gennemløber programmet en for-løkke, der tjekker om tallet på den samme indeks på de to forskellige arrays (*mmArray2D* og *gMasterMind*) er ens. Hvis de er ens, bliver der erklæret false på bool array *bMMCheck* på indekset, hvilket indikerer, at tallet fra *gMasterMind* ikke kan bruges mere. Og derefter bliver der skrevet et 2-tal på *checkmmArray2D*, der indeholder spillerens 'checkliste'. 2-tallet returnerer en sort firkant og 1-tal returnerer en hvid firkant igennem metoden *DrawMasterMind()*, se figur 7.

```

2 references
private void CheckArr(int rownums)
{
    //Restart bMMCheck
    bMMCheck = new bool[4] { true, true, true, true };

    //Checking for the right colors with the right placement(black)
    for (int i = 0; i < mmArray2D.GetLength(1); i++)
    {
        if (mmArray2D[rownums, i] == gMasterMind[i])
        {
            //Makes sure the number is used
            bMMCheck[i] = false;

            //Sets 2 = black
            checkmmArray2D[rownums, i] = 2;
        }
    }
}

```

Figur 7 - Et skærmbillede af metoden *CheckArr()* (sort-del)

Med bool *bMMCheck* kan programmet sikre, at spilleren ikke fx får to hvide for en rigtig farve på farvekombinationen, hvis farvekombinationen indeholder to eller flere ens farver og spilleren også har selv et bud på to eller flere farver.

For at finde de hvide, gennemløber programmet en for-løkke, se figur 8. Undervejs i for-løkken tjekker programmet først om det første indeks fra spillerens bud er blevet brugt før ved hjælp af "if-statement". Dette gør programmet ved at tjekke om tallet på indekset er 0 på *checkmmArray2D*. Med dette if-

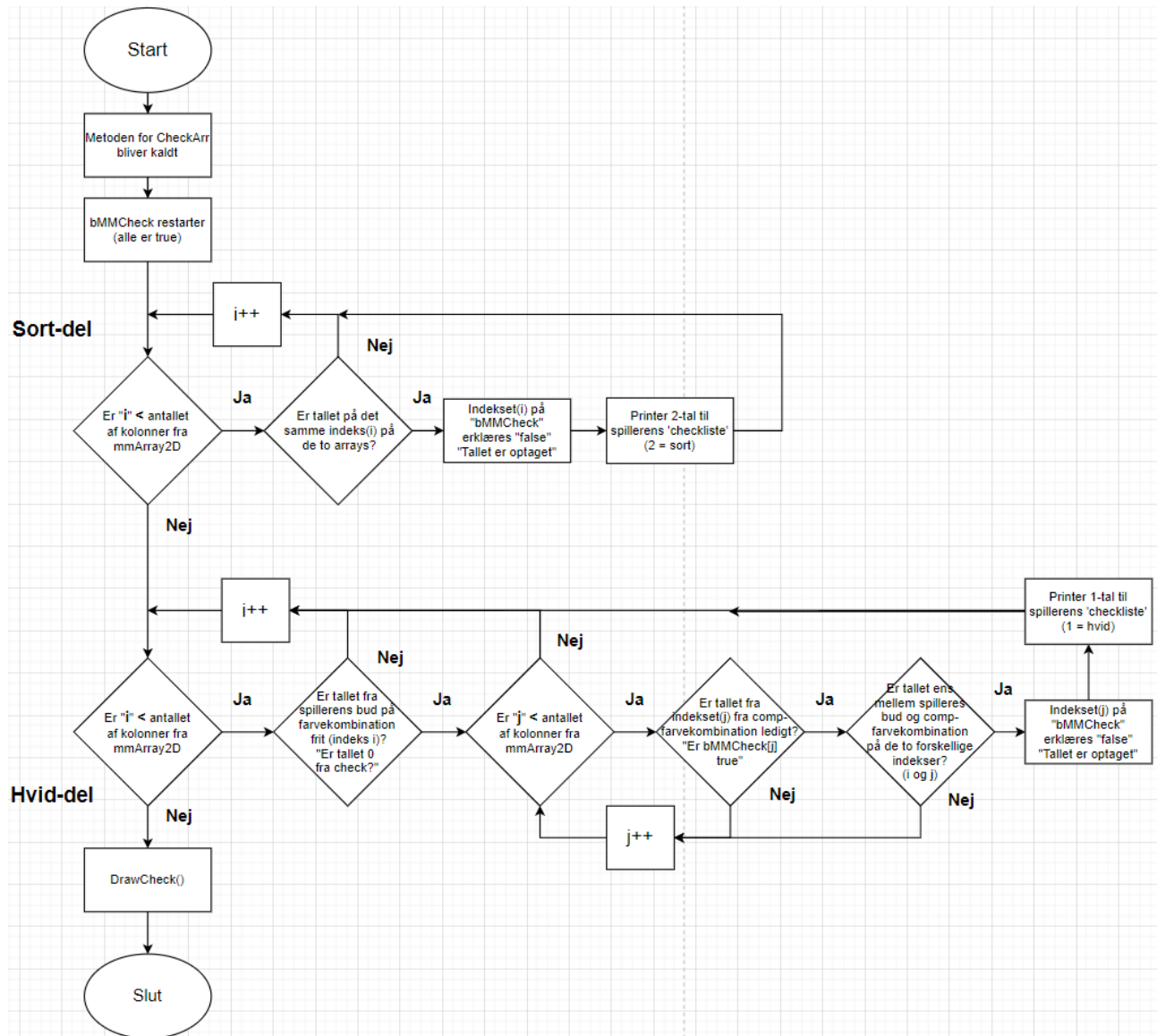
statement sikres det, at spilleren ikke kan få både en sort og en hvid på det samme bud.

Derefter gennemløber programmet en ny for-løkke, der både tjekker om tallet fra computerens farvekombination ikke er optaget ved hjælp af `bMMCheck` og om tallet er ens på de to forskellige indekser. Når programmet har fundet en lighed, bliver der erklæret `false` på `indekset(j)` på `"bMMCheck"` og så bliver der skrevet et 1-tal på `checkmmArray2D`, og så bryder programmet den anden for-løkke, og programmet tager det næste indeks i spillerens bud på farvekombinationen. Til sidst tegnes checkliste vha. metoden `"DrawCheck"`, så spilleren kan se tilbagemelding på budet.

```
//Checking for the right colors but wrong placement (white)
for (int i = 0; i < mmArray2D.GetLength(1); i++)
{
    //Checking if the number from 'checklist' is available
    if (checkmmArray2D[rownums, i] == 0)
    {
        for (int j = 0; j < mmArray2D.GetLength(1); j++)
        {
            //checks if the number is recorded
            if (bMMCheck[j])
            {
                //check if they is identical
                if (mmArray2D[rownums, i] == gMasterMind[j])
                {
                    //Makes sure the number is used
                    bMMCheck[j] = false;
                    checkmmArray2D[rownums, i] = 1;
                    break;
                }
            }
        }
    }
}
```

Figur 8 - Et skærbillede af metoden `CheckArr()` (hvid-del)

Jeg har udarbejdet et flowdiagram over metoden `"CheckArr()"`, som kan ses nedenunder, se figur 9.



Figur 9 - En illustration, der beskriver flowdiagram af metoden `CheckArr()`

Metode til at tegne spilleren bud på farvekombination

Programmet indeholder en metode, der kaldes *"DrawMasterMind()"*. Metoden gennemlæser hele det todimensionelle array; *"mmArray2D"*, der indeholder værdier på spillerens bud på farvekombination og tegner dem på *"canvas"*, hvor efter programmet gennemløber to for-løkker, en for arrayets rækker og en for arrayets kolonner. Under den sidste for-løkke gennemgår programmet et switch-statement, der har 6 forskellige case og en default. Hver case indeholder et tal efter *"farven-koden"*, se figur 10. Så hvis spilleren trykker på rød, så bliver der lavet et et-tal i arrayet, hvor der så returneres en rød firkant efter case 1.

Farve	Tal
Rød	1
Blå	2
Grøn	3
Pink	4
Gul	5
Orange	6

Figur 10 - En tabel over farven-koden

Under default, hvilket er de resterende tal som et 0-tal, bliver der tegnet en firkant med *"grå pen"*. Koden for *"DrawMasterMind()"* kan ses under [bilag 5](#).

Metode til at tegne svar-firkanter

For at tegne svar-firkanter, som de sorte og hvide, når spilleren har gættet rigtigt, benyttes metoden *"DrawCheck()"*, se [bilag 6](#) for koden. Når der kaldes på metoden *"DrawCheck()"*, gennemløber programmet en for-løkke, der bliver gentaget efter antallet af rækker på arrayet. Programmet har et 2d array 'checkliste', der indeholder værdier for hvilke tal, der er rigtige. Men spilleren skal ikke kunne se, hvilke bud der er sorte eller hvide, hvorfor der bliver skabt en instans fra en *"CountCheck"*-klasse, der tæller, hvor mange sorte (2'er) og hvide (1'er) der er på hver række. Klassen skal have to parametre; et todimensionelt array og tallet for rækken. Klassen sorterer derefter det 'valgte' array efter tal (2,1,0 som er hhv. sorte, hvide og ingen rigtige). Under klassen er der lavet 3 metoder, der returnerer de 3 variabler, der indeholder de tre oplysninger. Når programmet har gennemgået klassen, skal programmet gennemgå tre små for-løkker, se figur 11. De tre små for-løkker bliver gentaget efter antal af de sorte, hvide eller ingen rigtige igennem metoden fra klassen.

Koden for klassen *"CountCheck-class"* ses under [bilag 7](#).

```

for (int g = 0; g < checks.Black(); g++)
{
    brushColor = new SolidBrush(Color.Black);
    gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
    //Shifts x-position after each drawing of square
    xCposi = xCposi + CwidthDraw;
}

for (int h = 0; h < checks.White(); h++)
{
    brushColor = new SolidBrush(Color.White);
    gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
    //Shifts x-position after each drawing of square
    xCposi = xCposi + CwidthDraw;
}

for (int j = 0; j < checks.Nothing(); j++)
{
    gObject.DrawRectangle(GrayPen, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
    //Shifts x-position after each drawing of square
    xCposi = xCposi + CwidthDraw;
}

```

Figur 11 - Et skærbillede af de tre små for-løkker under metoden "DrawCheck()"

Til sidst tjekker programmet om spilleren har 4 rigtige (4 sorte firkanter) vha. et "if-statement", der er true, når antallet af sorte er lig med antallet af kolonner af arrayet. Når spilleren har 4 rigtige, så bliver stopuret stoppet og spillerens navn og tid printes til highscore, der bagefter bliver opdateret på textbox på GameForm, så spilleren kan se sit resultat på highscore-listen.

Derefter åbner programmet en messagebox til spilleren, der har to knapper. En ja-knap og en nej-knap.

For at tjekke hvad spilleren vælger, har programmet et "if-statement", der tjekker om spilleren trykker på ja eller nej. Hvis spilleren trykker på ja, om at ønske at starte spillet igen, bliver "restartBool" erklæret true, hvilket så vil genstarte spillet. Hvis spilleren trykker på nej, så lukkes GameFom af "this.Close()", se figur 12.

```

//Check if you have 4 blacks = then you have won
if (checks.Black() == gMasterMind.Length)
{
    //Stop the stopwatch
    mmStopWatch.Stop();

    //updates high score
    PrintToHighscore();

    //Make a messagebox
    string message = "Congratulations you have won" + "\r\n" +
        "Do you want to play again?";
    string title = "Close Window";

    //Creates a message box with a yes and no button
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, title, buttons);

    if (result == DialogResult.Yes)
    {
        //"true" is declared that the player wants to restart the game
        restartBool = true;
        break;
    }
    else if (result == DialogResult.No)
    {
        //Closing the form
        this.Close();
    }
}

```

Figur 12 - Et skærbillede af metoden DrawCheck()

Metode til at genoptage sit gemte spil

Metoden *"ContinueGame()"* sørger for at indlæse filen og hente de vigtige informationer, der skal bruges til at spille, se figur 13, for at se et skærbillede af en "SaveGame"-fil.

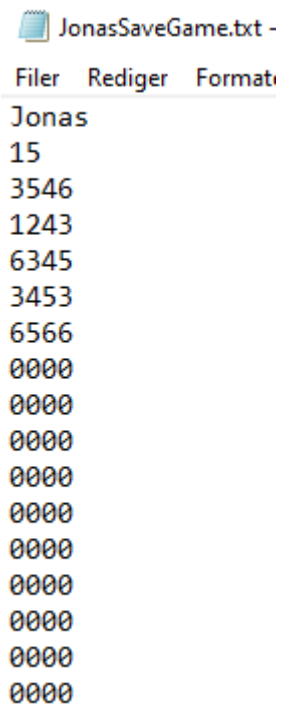
Metoden starter med at skabe en variabel for *"OpenFileDialog"*, der åbner for computerens stifinder, hvor spilleren så skal finde sit gemte spil på computeren. Når spilleren har trykket på *"åbn"*, bliver der skabt en string variabel *"filepath"* til filens navn. Denne variabel bliver brugt til en string array *"lines"*, der læser alle linjer fra filens navn; *"File.ReadAllLines(filepath)"*.

```
//lines contains all lines that are read from the file
string[] lines = File.ReadAllLines(filepath);
```

"lines" indeholder nu alle værdier for hver linje. Den første linje er spillerens navn og derfor startes programmet med at indlæse spillerens navn og sætte navnet ind i *"playerName"*. Derefter hentes tiden til *"extraTime"*, så spilleren kan fortsætte fra sidst.

Den tredje linje fra filen er tallene til computerens farvekombination og resten af linjerne er spillerens bud på farvekombinationen. Der bliver brugt en for-løkke til at læse alle tallene på hver linje. For-løkken starter ved den 3. linje, da *"i"* i for-løkken er sæt til at være 2. (0,1,2). Det første der læses, er den 3. linje, som indeholder tal til computerens farvekombination. Her benytter jeg en *"if-sætning"*, der er true, når *i = 2*, hvilket er med det samme når for-løkken startes. For at indsætte alle værdier fra 3. linje ind til computerens farvekombination *"gMasterMind"*, benyttes der en for-løkke igen, der konverterer tallet fra string til int.

Derefter er der en ny for-løkke, der læser alle tal fra de resterende linjer, der genindlæses på det 2d array, som indeholder spillerens bud *"mmArray2D"*, se figur 14.



JonasSaveGame.txt -

Filer	Rediger	Format
Jonas		
15		
3546		
1243		
6345		
3453		
6566		
0000		
0000		
0000		
0000		
0000		
0000		
0000		
0000		
0000		

Figur 13 - Et skærbillede af en 'SaveGame'-fil

```

//columns contains all values from a row (columns)
//Subtracting one as I want to look at the last line.
string columns1 = lines[lines.Length-1];

//Loads number of length to newly created array.
//I subtract 3, due to the first 3 lines (Name, time and "gMasterMind"
mmArray2D = new int[lines.Length - 3, columns1.Length];

//I start from 2, as I disregard the first two lines
for (int i = 2; i < lines.Length; i++)
{
    //print all numbers in line to hall
    string hall = lines[i];

    //Loads numbers to gMasterMind that the player must guess
    //i must be 2, as it is on the 3rd line from the file
    if (i == 2)
    {
        for (int j = 0; j < hall.Length; j++)
        {
            //Convert char to string and then to int
            gMasterMind[j] = Convert.ToInt32(hall[j].ToString());
        }
    }
    else
    {
        for (int j = 0; j < hall.Length; j++)
        {
            //i subtract 3 from i since i have to start at 0
            mmArray2D[i - 3, j] = Convert.ToInt32(hall[j].ToString());
        }
    }
}

```

Figur 14 - Et skærbillede af metoden "ContinueGame()"

Derefter skaber programmet et nyt 2d array til 'checkliste', som har den samme længde som arrayet, der indeholder spillerens bud (mmArray2D).

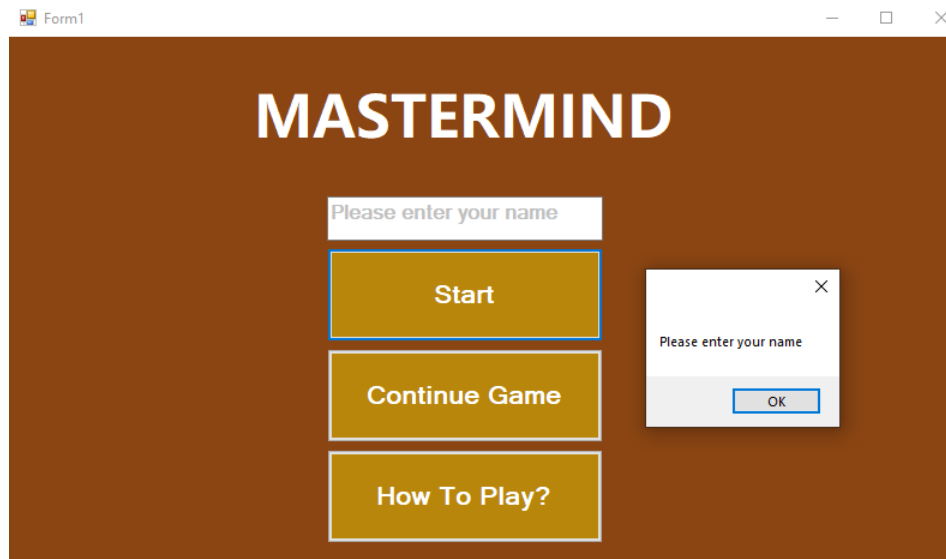
Til sidst skal programmet undersøge, hvilken række, spilleren var kommet til, så spilleren ikke skal starte forfra. Programmet gennemløber en for-løkke, der leder efter et 0-tal i på de resterende linjer, som er tiltænkt spillerens bud. Når programmet har fundet et 0-tal, bliver variablen "numRowarr" bestemt efter hvilken linje, programmet fandt 0-tal i, og trækker 3 linjer fra, da tallene for spillerens bud først startede ved den 3. linje fra filen.

Hvis programmet indser at spilleren ikke har trykket på "åbn", men trykket på krydset eller 'annuller', så bliver GameForm lukket ned, så spilleren forbliver på startside.

Test af programmet

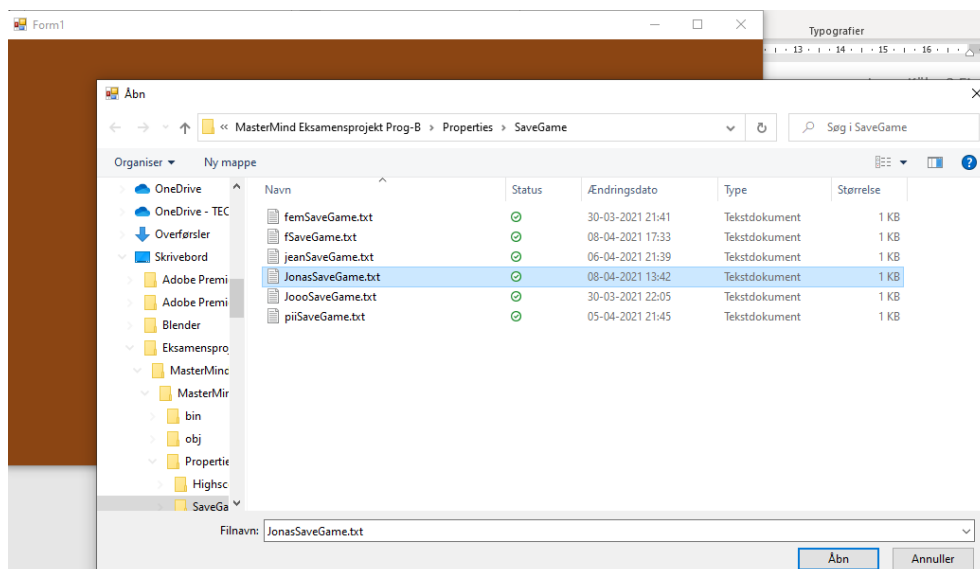
For at teste om mit program er velfungerende, har jeg testet mit program på min forældre undervejs, for at teste om det hele virker som det skal.

For det første kan der ses, at når man har glemt at indtaste sit navn og trykke på start, popper der en messagebox op, der påminder spilleren om at indtaste sit navn, se figur 15.



Figur 15 - Et skærbillede af startsidens med messagebox

Jeg vil vise en test af hvordan man kan genoptage sit gemte spil. For at genoptage sit gemte spil, skal man trykke på "Continue Game-knap", hvor efter der bliver åbnet en stifinder, hvor man kan finde sit gemte spil, se figur 16.

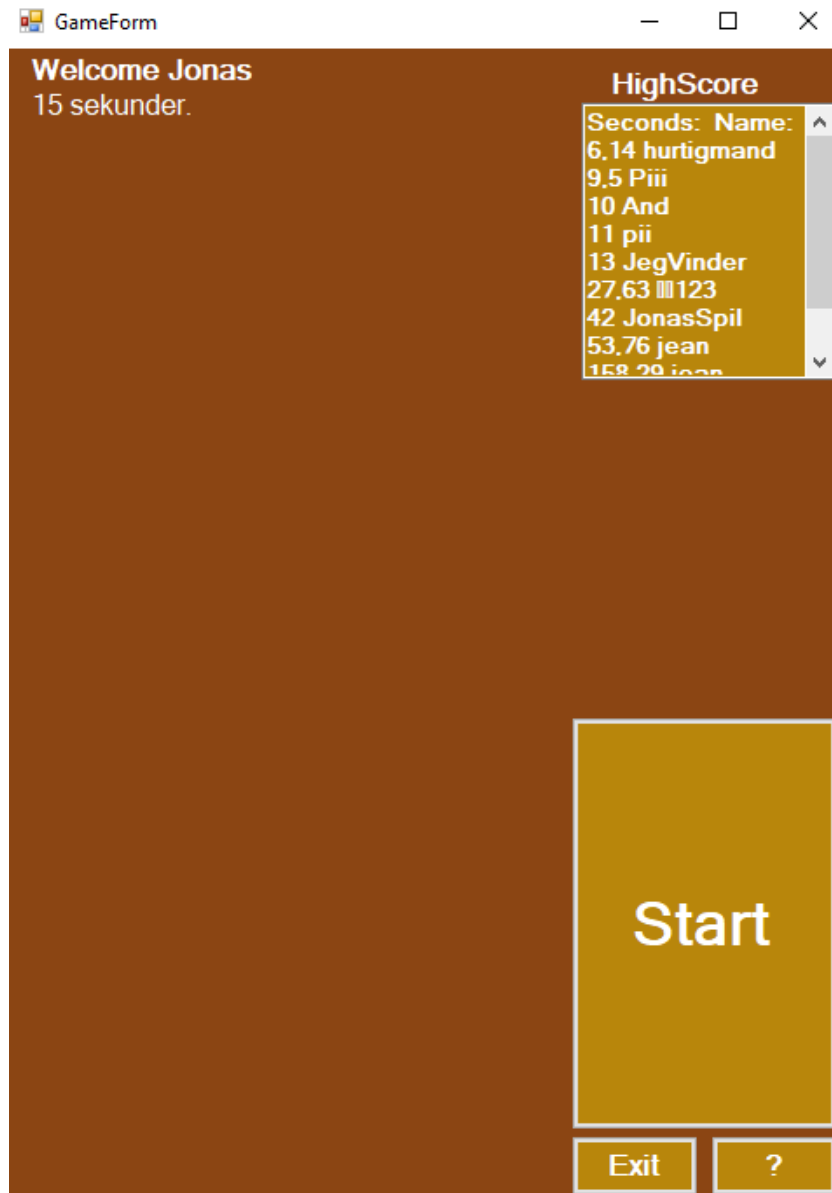


Figur 16 - Et skærbillede af computerens stifinder

Hvis man gerne vil genoptage sit tidligere spil, skal man trykke på den fil, der har dit navn udenfor "SaveGame", hvis man skulle tage mit navn, så skulle man åbne filen "JonasSaveGame", da filen er gemt ud fra spillerens navn + "SaveGame" i en txt-fil.

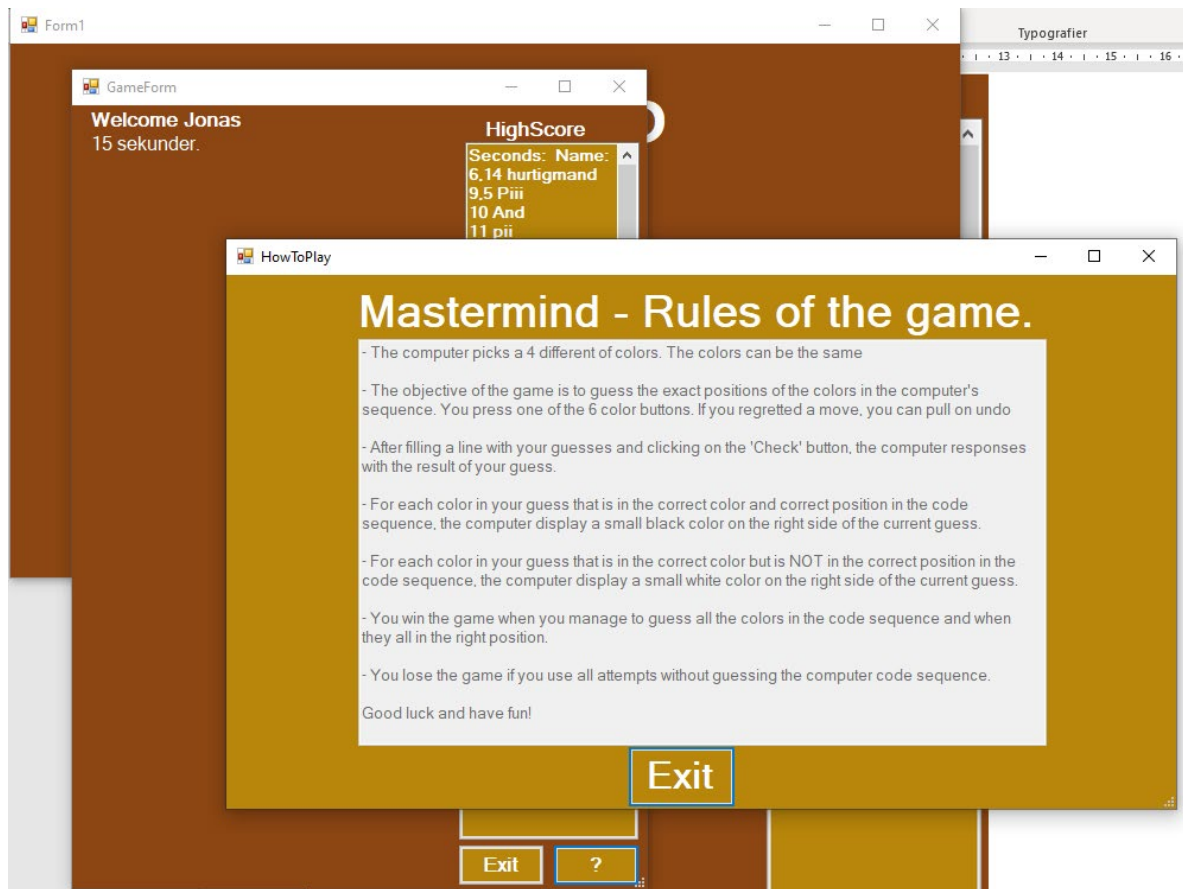
Når man så trykker på filen, åbnes GameForm med oplysninger fra spillet. Man kan se sit navn på velkomsttekst og hvad tiden er fra sidst øverst oppe. Men man ikke se sin bud på farvekombination før man starter spillet, se figur 17.

Dette skyldes for at undgå at "snyde" som sidst med en tænkepause, og derudover er filen skrevet med tal, så man ikke rigtigt kan gennemskue, hvad er hvad på filen.



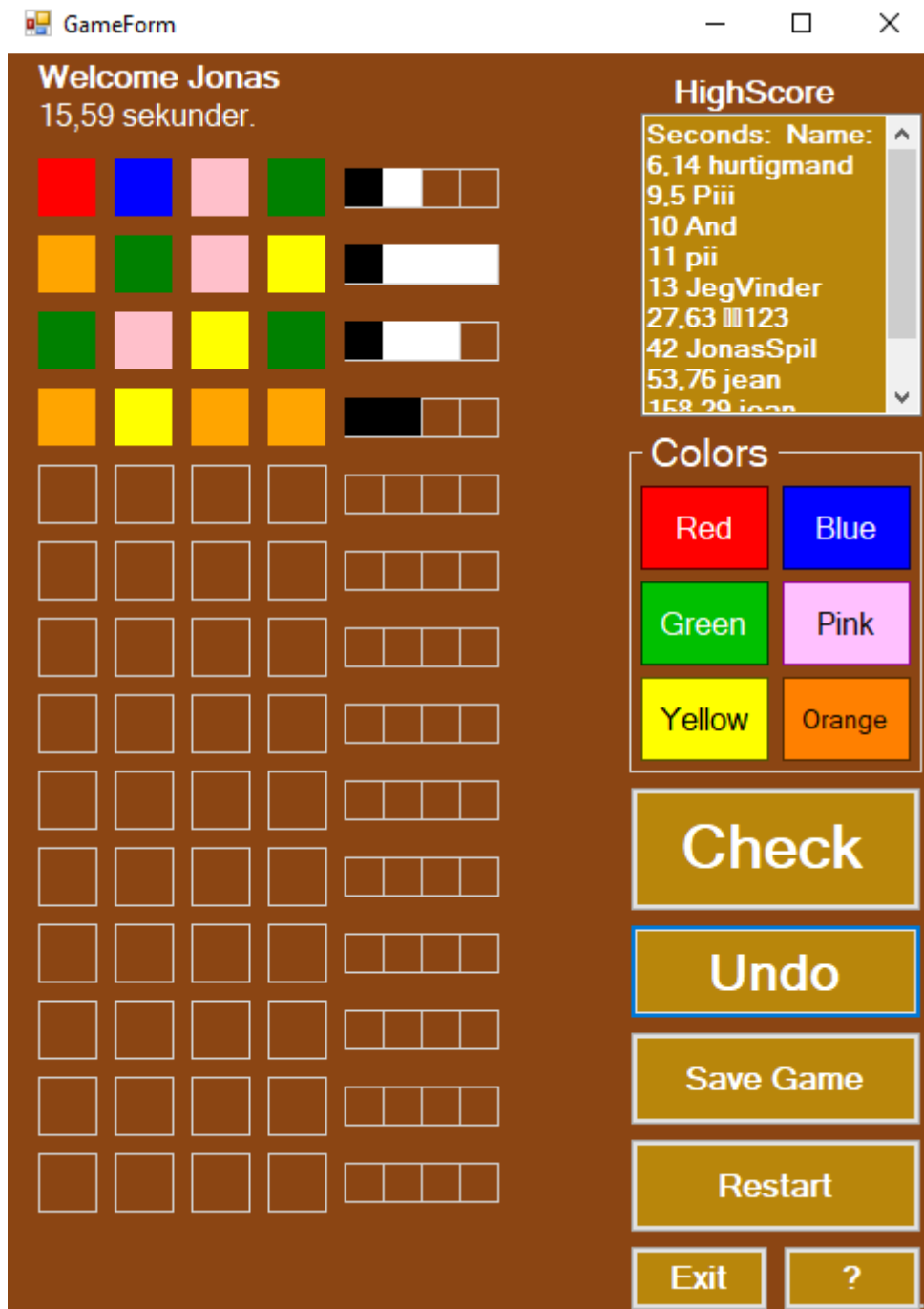
Figur 17 - Et skærbillede af GameForm før start

Hvis man lige skal genopfriske reglerne for MasterMind kan man trykke på "?"-knap, hvor efter der åbnes en HowToPlayForm, se figur 18.



Figur 18 - Et skærbillede af HowToPlayForm

Når man så gerne vil starte spillet, trykker man bare på start, og så bliver resten af spillet synliggjort og tiden startes, se figur 19.



Figur 19 - Et skærbillede af GameForm efter start

Konklusion

Projektet har været spændende og vellykket. Jeg har fået udarbejdet et simpelt spil med brugerinteraktion via mus, hvor jeg har lavet en version af MasterMind, hvor man spiller med computeren.

Programmet opfylder mange af kravene til projektet, som bl.a. at man kan genoptage eller gemme sit spil efter brug af metoder og klasser og efter brug af teknikken "Stepwise Improvement" (htx-elev.ucholstebro 2013), der betyder at programmet løbende bliver testet og kodet i små steps, hvilket har gjort udviklingen overskuelig.

Jeg kan konkludere at projektet er vellykket efter testning på mine forældre, hvor deres feedback er at mit program er velfungerende og er nemt, og ikke mindst sjovt at spille på. De kunne godt manøvrere rundt i mit program efter, hvordan man gemmer sit spil eller genoptager sit gemte spil ved hjælp af informationer, der bliver vist på skærmen. De kunne også godt lide den grafiske del af programmet, da det var nemt for dem at se tilbagemelding på deres bud.

Litteraturliste:

htx-elev.ucholstebro 2013: " Stepwise Improvement"

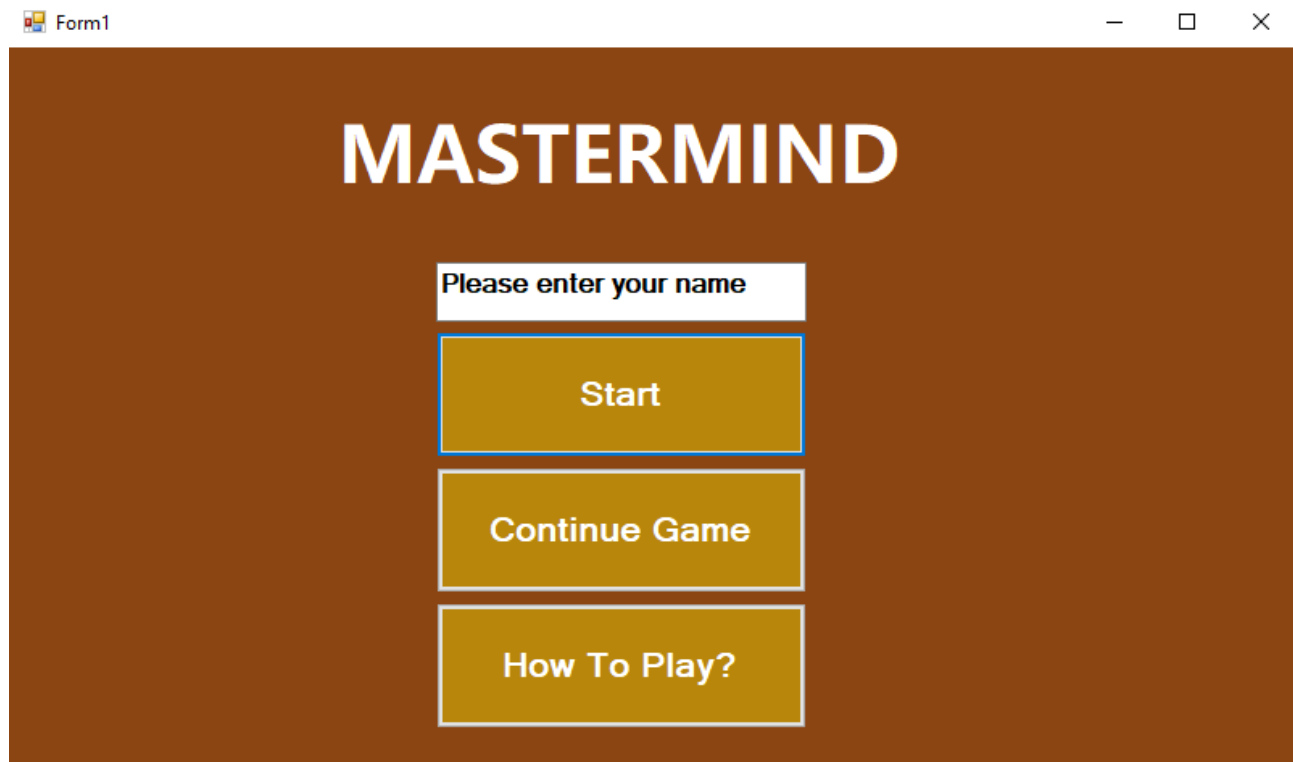
htx-elev.ucholstebro.dk, d. 14 november.

[http://htx-elev.ucholstebro.dk/wiki/index.php?title=Stepwise Improvement](http://htx-elev.ucholstebro.dk/wiki/index.php?title=Stepwise_Improvement)

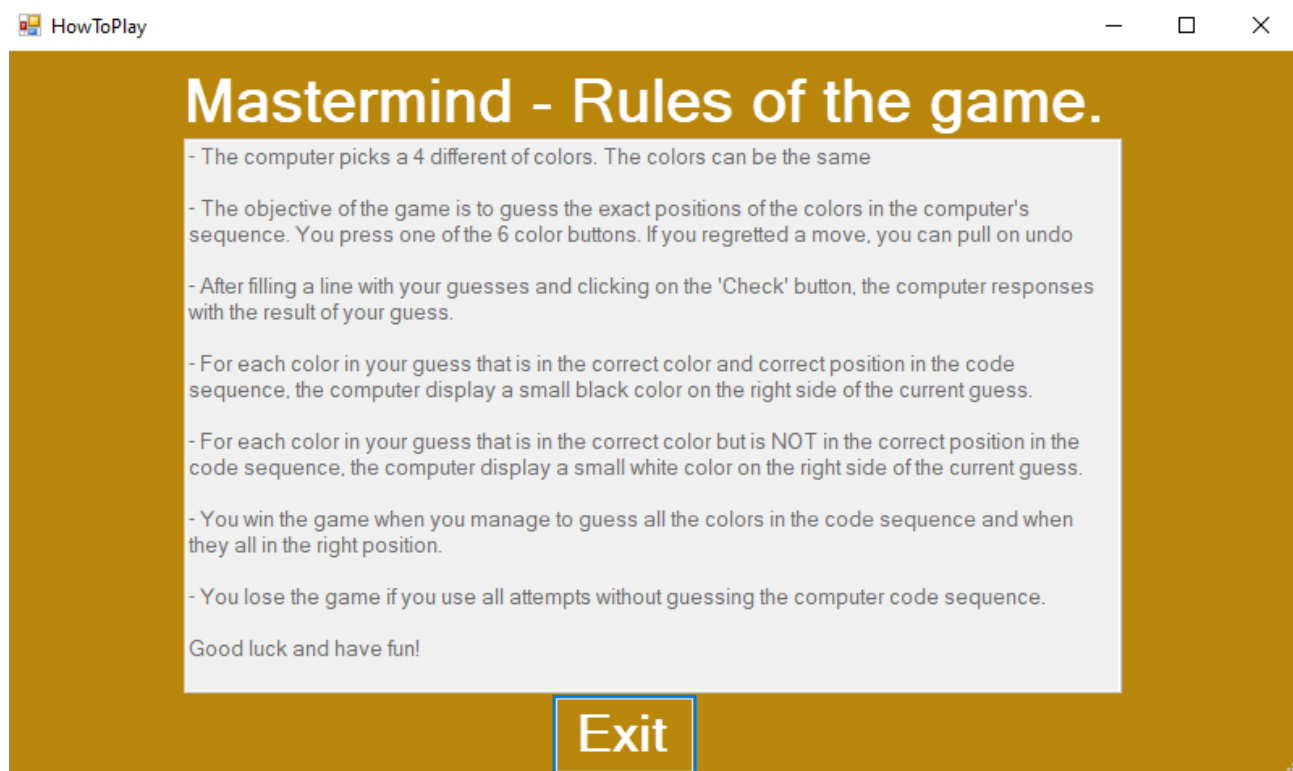
Lokaliseret d. 29.03.2020

Bilag

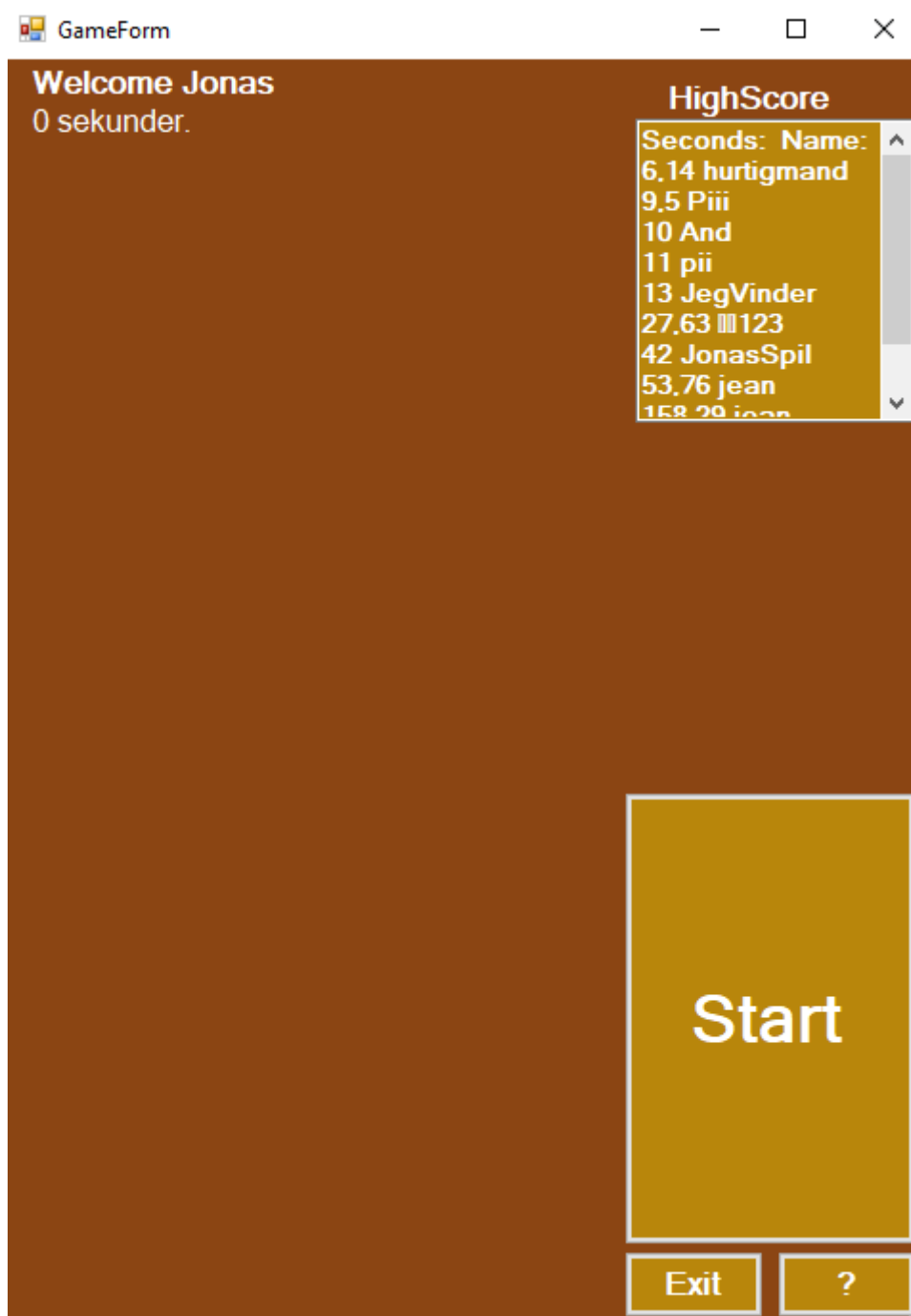
Bilag 1 – Startside



Bilag 2 – HowToPlayForm



Bilag 3 – GameForm før start



Bilag 4 – GameForm under spillet



Bilag 5 – Koden til DrawMasterMind()

```
private void DrawMasterMind()
{
    //Restarts squares positions
    Coloredsquares();

    //creates an object to draw on canvas
    Graphics gObject = canvas1.CreateGraphics();

    //creates an object to draw with a pen
    Pen GrayPen = new Pen(Color.LightGray, 1);

    //two forerunners for each rows and columns
    for (int i = 0; i < mmArray2D.GetLength(0); i++)
    {
        //yposition is controlled by rows and 10 is air between the boxes
        yposi = 10 + 40 * i;

        //restarts x-position
        xposi = 10;

        for (int j = 0; j < mmArray2D.GetLength(1); j++)
        {
            //Adding x-position to the next
            xposi = 10 + 40 * j;

            //Uses switch statement to color the player's bid
            //for the correct color combination
            switch (mmArray2D[i, j])
            {
                case 1:
                    brushColor = new SolidBrush(Color.Red);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 2:
                    brushColor = new SolidBrush(Color.Blue);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 3:
                    brushColor = new SolidBrush(Color.Green);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 4:
                    brushColor = new SolidBrush(Color.Pink);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 5:
                    brushColor = new SolidBrush(Color.Yellow);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 6:
                    brushColor = new SolidBrush(Color.Orange);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                default:
                    gObject.DrawRectangle(GrayPen, xposi, yposi, widthDraw, heightDraw);
                    break;
            }
        }
    }
}
```

Bilag 6 – Kodet til DrawCheck()

```
private void DrawCheck()
{
    //creates an object to draw on canvas
    Graphics gObject = canvas1.CreateGraphics();

    //creates an object to draw with a pen
    Pen GrayPen = new Pen(Color.LightGray, 1);

    for (int i = 0; i < checkmmArray2D.GetLength(0); i++)
    {
        //Creates a new instance of CountCheck-class
        //The class counts after black and white after each lines
        CountCheck checks = new CountCheck(checkmmArray2D, i);

        //Restart checks squares
        CheckSquares();

        //yposition for Check squares is controlled by rows (i)
        //and 10 is air between the boxes
        yCposi = 10 + 40 * i;

        for (int g = 0; g < checks.Black(); g++)
        {
            brushColor = new SolidBrush(Color.Black);
            gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }

        for (int h = 0; h < checks.White(); h++)
        {
            brushColor = new SolidBrush(Color.White);
            gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }

        for (int j = 0; j < checks.Nothing(); j++)
        {
            gObject.DrawRectangle(GrayPen, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }
    }
}
```

```
//
//Check if you have 4 blacks = then you have won
if (checks.Black() == gMasterMind.Length)
{
    //Stop the stopwatch
    mmStopWatch.Stop();

    //updates high score
    PrintToHighscore();

    //Make a messagebox
    string message = "Congratulations you have won" + "\r\n" +
        "Do you want to play again?";
    string title = "Close Window";

    //Creates a message box with a yes and no button
    MessageBoxButtons buttons = MessageBoxButtons.YesNo;
    DialogResult result = MessageBox.Show(message, title, buttons);

    if (result == DialogResult.Yes)
    {
        //"true" is declared that the player wants to restart the game
        restartBool = true;
        break;
    }
    else if (result == DialogResult.No)
    {
        //Closing the form
        this.Close();
    }
}
}
```

Bilag 7 – Kodet til CountCheck-klasse

```
3 references
class CountCheck
{
    //The basis class
    public int correctlocation;
    public int correctColor;
    public int noCorrect;

    //Class CountCheck being made
    1 reference
    public CountCheck(int[,] arr, int rownums)
    {
        for (int j = 0; j < arr.GetLength(1); j++)
        {
            if (arr[rownums, j] == 2)
            {
                //Sort
                correctlocation++;
            }
            if (arr[rownums, j] == 1)
            {
                //Hvid
                correctColor++;
            }
            if (arr[rownums, j] == 0)
            {
                noCorrect++;
                //firkant
            }
        }
    }

    //Create a method for black
    2 references
    public int Black()
    {
        return correctlocation;
    }

    //Create a method for white
    1 reference
    public int White()
    {
        return correctColor;
    }

    //Create a method for nothing
    1 reference
    public int Nothing()
    {
        return noCorrect;
    }
}
```

Bilag 8 – Resterende kode:

Bilag 8.1 – Startside.cs

```

3 references
public partial class Startside : Form
{
    1 reference
    public Startside()
    {
        InitializeComponent();
        txtName.Text = "Please enter your name";
    }
    //Create variable for playersname
    string player;
    1 reference
    private void btnStart_Click(object sender, EventArgs e)
    {
        //checks that the player has written his name
        if (txtName.Text != "" && txtName.Text != "Please enter your name")
        {
            //Create name to the player
            player = txtName.Text;

            //Switch forms
            GameForm GF = new GameForm(player, false);
            GF.ShowDialog();
        }
        else
        {
            MessageBox.Show("Please enter your name");
        }
    }

    1 reference
    private void btnHowToPlay_Click(object sender, EventArgs e)
    {
        //Switch form1 to HowToPlay-form
        HowToPlayForm HTPF = new HowToPlayForm();
        HTPF.ShowDialog();
    }

    1 reference
    private void btnContinueGame_Click(object sender, EventArgs e)
    {
        //Switch forms
        GameForm GF = new GameForm("", true);
        GF.ShowDialog();
    }

    1 reference
    private void txtName_Enter(object sender, EventArgs e)
    {
        //checks if he has written a name, and if not, then the text is "blank"
        if (txtName.Text == "Please enter your name")
        {
            txtName.Text = "";
            txtName.ForeColor = Color.Black;
        }
    }

    1 reference
    private void txtName_Leave(object sender, EventArgs e)
    {
        //if the player has not written the name, then gray text will appear stating that the player is asked to write his name
        if (txtName.Text == "")
        {
            txtName.Text = "Please enter your name";
            txtName.ForeColor = Color.Silver;
        }
    }
}

```

Bilag 8.2 - HowToPlayForm.cs

```
public partial class HowToPlayForm : Form
{
    public HowToPlayForm()
    {
        InitializeComponent();
    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        //Closing the form
        this.Close();
    }
}
```

Bilag 8.3 – GameForm.cs

```

6 references
public partial class GameForm : Form
{
    2 references
    public GameForm(string nameoftheplayer, bool continueGame)
    {
        InitializeComponent();
        playerName = nameoftheplayer;
        continueMyGame = continueGame;
    }
    //Create a bool variable for the player's desire to resume a game
    bool continueMyGame;
    //Create a variable for the playername
    string playerName;

    //Creates variables to timer
    double mmTime;
    double extraTime;

    //Computer' color combination
    //the numbers that the player has to guess
    int[] gMasterMind = new int[4];

    //Creates a 2d array that stores all values from the player's bid
    //for color combination
    int[,] mmArray2D = new int[14, 4];

    //Checklist
    //2d array for all our checklists (black and white, 2.1)
    int[,] checkmmArray2D = new int[14, 4];
    bool[] bmmCheck = new bool[4] { true, true, true, true};

    //Create a variabel for BrushColor;
    Brush brushColor;

    //create object for stopwatch
    Stopwatch mmStopwatch = new Stopwatch();

    //number of the row
    int numRowArr = 0;

    //created to check if the player wants to restart
    bool restartBool;

    //square to the colored squares
    int xposi, yposi, widthDraw, heightDraw;

    //square to Check
    int xCposi = 10, yCposi = 10, CwidthDraw = 20, CheightDraw = 20;

```

Bilag 8.3 – GameForm.cs – Del 2

```

1 reference
private void GameForm_Load(object sender, EventArgs e)
{
    //Checks if the player wants to take a normal game or a continegame
    if (continueMyGame == true)
    {
        ContinueGame();
    }
    if (continueMyGame == false)
    {
        gMasterMindGuess();
    }

    //Print your name
    lblName1.Text = "Welcome " + playerName;

    //Read Highscore:
    ReadHighScore();
}

//Makes an array that contains various
//numbers that the player has to guess
2 references
private void gMasterMindGuess()
{
    //Minimum number
    int Min = 1;
    //Max number, and random.next(max) cant be 7
    //(max 6)(0<= x < 7)
    int Max = 7;

    //Instantiate random number generator
    Random randNum = new Random();

    //Creates a random color combination for the player
    for (int i = 0; i < gMasterMind.Length; i++)
    {
        gMasterMind[i] = randNum.Next(Min, Max);
    }
}

1 reference
private void Coloredsquares()
{
    //Square starter for colored squares
    xpos1 = 10;
    ypos1 = 10;
    widthDraw = 30;
    heightDraw = 30;
}

1 reference
private void CheckSquares()
{
    //Square-starter for check squares
    xCpos1 = 10 + 40 * mmArray2D.GetLength(1);
    yCpos1 = 10;
    CwidthDraw = 20;
    CheightDraw = 20;
}

```


Bilag 8.3 – GameForm.cs – Del 3

1 reference

```
private void btnStart_Click(object sender, EventArgs e)
{
    //Drawing the mastermind and checklist
    DrawMasterMind();
    DrawCheck();

    //Goes through checks when, for example, continuing the game
    //only works when resuming the game, because then numRowsArr is greater than 0
    for (int i = 0; i < numRowsArr; i++)
    {
        CheckArr(i);
    }

    //invisible start-button
    btnStart.Visible = false;
    //visible colors-gropbox
    groupBox1.Visible = true;

    //Declared that the player does not want to restart the game
    restartBool = false;

    //Start the stopwatch
    mmStopWatch.Start();
}
```

6 references

```
private void GamePlay(int num101)
{
    //Creates a new instance of this class
    NumberInArrayClass UpdataArr = new NumberInArrayClass();
    mmArray2D = UpdataArr.UpdateArr(mmArray2D, num101, numRowsArr);

    //Draw MasterMind
    DrawMasterMind();
}
```

Bilag 8.3 – GameForm.cs – Del 4

4 references

```

private void DrawMasterMind()
{
    //Restarts squares positions
    Coloredsquares();

    //creates an object to draw on canvas
    Graphics gObject = canvas1.CreateGraphics();

    //creates an object to draw with a pen
    Pen GrayPen = new Pen(Color.LightGray, 1);

    //two forerunners for each rows and columns
    for (int i = 0; i < mmArray2D.GetLength(0); i++)
    {
        //yposition is controlled by rows and 10 is air between the boxes
        yposi = 10 + 40 * i;

        //restarts x-position
        xposi = 10;

        for (int j = 0; j < mmArray2D.GetLength(1); j++)
        {
            //Adding x-position to the next
            xposi = 10 + 40 * j;

            //Uses switch statement to color the player's bid
            //for the correct color combination
            switch (mmArray2D[i, j])
            {
                case 1:
                    brushColor = new SolidBrush(Color.Red);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 2:
                    brushColor = new SolidBrush(Color.Blue);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 3:
                    brushColor = new SolidBrush(Color.Green);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 4:
                    brushColor = new SolidBrush(Color.Pink);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 5:
                    brushColor = new SolidBrush(Color.Yellow);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                case 6:
                    brushColor = new SolidBrush(Color.Orange);
                    gObject.FillRectangle(brushColor, xposi, yposi, widthDraw, heightDraw);
                    break;
                default:
                    gObject.DrawRectangle(GrayPen, xposi, yposi, widthDraw, heightDraw);
                    break;
            }
        }
    }
}

```

Bilag 8.3 – GameForm.cs – Del 5

```

4 references
private void DrawCheck()
{
    //creates an object to draw on canvas
    Graphics gObject = canvas1.CreateGraphics();

    //creates an object to draw with a pen
    Pen GrayPen = new Pen(Color.LightGray, 1);

    for (int i = 0; i < checkmmArray2D.GetLength(0); i++)
    {
        //Creates a new instance of CountCheck-class
        //The class counts after black and white after each lines
        CountCheck checks = new CountCheck(checkmmArray2D, i);

        //Restart checks squares
        CheckSquares();

        //yposition for Check squares is controlled by rows (i)
        //and 10 is air between the boxes
        yCposi = 10 + 40 * i;

        for (int g = 0; g < checks.Black(); g++)
        {
            brushColor = new SolidBrush(Color.Black);
            gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }

        for (int h = 0; h < checks.White(); h++)
        {
            brushColor = new SolidBrush(Color.White);
            gObject.FillRectangle(brushColor, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }

        for (int j = 0; j < checks.Nothing(); j++)
        {
            gObject.DrawRectangle(GrayPen, xCposi, yCposi + 5, CwidthDraw, CheightDraw);
            //Shifts x-position after each drawing of square
            xCposi = xCposi + CwidthDraw;
        }

        //Check if you have 4 blacks = then you have won
        if (checks.Black() == gMasterMind.Length)
        {
            //Stop the stopwatch
            mmStopWatch.Stop();

            //updates high score
            PrintToHighscore();

            //Make a messagebox
            string message = "Congratulations you have won" + "\r\n" +
                             "Do you want to play again?";
            string title = "Close Window";

            //Creates a message box with a yes and no button
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult result = MessageBox.Show(message, title, buttons);

            if (result == DialogResult.Yes)
            {
                //"true" is declared that the player wants to restart the game
                restartBool = true;
                break;
            }
            else if (result == DialogResult.No)
            {
                //Closing the form
                this.Close();
            }
        }
    }
}

```

Bilag 8.3 – GameForm.cs – Del 6

```
2 references
private void CheckArr(int rownums)
{
    //Restart bMMCheck
    bMMCheck = new bool[4] { true, true, true, true };

    //Checking for the right colors with the right placement(black)
    for (int i = 0; i < mmArray2D.GetLength(1); i++)
    {
        if (mmArray2D[rownums, i] == gMasterMind[i])
        {
            //Makes sure the number is used
            bMMCheck[i] = false;

            //Sets 2 = black
            checkmmArray2D[rownums, i] = 2;
        }
    }

    //Checking for the right colors but wrong placement
    for (int i = 0; i < mmArray2D.GetLength(1); i++)
    {
        //Checking if the number from 'checklist' is available
        if (checkmmArray2D[rownums, i] == 0)
        {
            for (int j = 0; j < mmArray2D.GetLength(1); j++)
            {
                //checks if the number is recorded
                if (bMMCheck[j])
                {
                    //check if they is identical
                    if (mmArray2D[rownums, i] == gMasterMind[j])
                    {
                        //Makes sure the number is used
                        bMMCheck[j] = false;
                        checkmmArray2D[rownums, i] = 1;
                        break;
                    }
                }
            }
        }
    }

    //Drawing check
    DrawCheck();
}
```

Bilag 8.3 – GameForm.cs – Del 7

```

private void btnCheck_Click(object sender, EventArgs e)
{
    //Check that we have all 4 guesses
    //Writes '-1' at the end, because GetLength is 4 in total and the last index is 3
    if (mmArray2D[numRowArr, mmArray2D.GetLength(1)-1] != 0)
    {
        //Review the method of checking the player's bids
        CheckArr(numRowArr);
        //adding a number to row
        numRowArr++;

        //checks if the player want to restart
        if (restartBool)
        {
            //restart the game
            RestartGame();
        }

        //Checks if you have no more guesses
        if (mmArray2D.GetLength(0) == numRowArr)
        {
            //A message box pops up giving the player two choices to answer, yes and no if they want to play again
            string message = "Unfortunately you have no more guesses" + "\r\n" +
                "Do you want to play again?";
            string title = "Close Window";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult result = MessageBox.Show(message, title, buttons);
            if (result == DialogResult.Yes)
            {
                RestartGame();
            }
            else if (result == DialogResult.No)
            {
                this.Close();
            }
        }
    }
}

1 reference
private void btnUndo_Click(object sender, EventArgs e)
{
    //Clear my panel (canvas1)
    canvas1.Refresh();
    //Creates a new instance of this class and convert
    //the latest number to 0
    UndoArrayClass undoarr = new UndoArrayClass();
    mmArray2D = undoarr.UndoArr(mmArray2D, numRowArr);

    //Draw MasterMind
    DrawMasterMind();
    //Draw Check
    DrawCheck();
}

```

Bilag 8.3 – GameForm.cs – Del 8

```

1 reference
private void PrintToHighscore()
{
    //Creates a sortedlist that always sorts by double, so the fastest player is Nr. 1
    SortedList<double, string> highscore = new SortedList<double, string>();

    //Setting the locations of the file
    string filepath = @"C:\Users\kyhnj\OneDrive\Skribord\Eksamensprojekt Programming B\MasterMind Eksamensprojekt Prog-B\MasterMind Eksamensprojekt Prog-B\Properties\Highscore\HighScore.txt";

    //Read all lines from the file
    string[] lines = File.ReadAllLines(filepath);

    //i starts at 1 as I do not count the first line with(due to text "Second: Name:")
    for (int i = 1; i < lines.Length; i++)
    {
        //split number in between ' '
        string[] hall = lines[i].Split(' ');

        //Adds all names in each variable
        highscore.Add(Convert.ToDouble(hall[0]), hall[1]);
    }
    //Adder the latest name
    highscore.Add(mmTime, playerName);

    //Clear the file
    File.WriteAllText(filepath, String.Empty);

    //Adding extra text
    StreamWriter sw = File.AppendText(filepath);

    //printer "title" on highscore
    sw.WriteLine("Seconds: Name: ");

    //Goes through a loop that prints all names from the highscore list
    foreach (var item in highscore)
    {
        sw.WriteLine(item.Key + " " + item.Value);
    }
    sw.Flush();
    sw.Close();

    //Printer highscore using the method
    ReadHighScore();
}

2 references
private void ReadHighScore()
{
    //Clear textbox for restart
    txtHighScore.Clear();
    //Setting the locations of the file
    string filepath = @"C:\Users\kyhnj\OneDrive\Skribord\Eksamensprojekt Programming B\MasterMind Eksamensprojekt Prog-B\MasterMind Eksamensprojekt Prog-B\Properties\Highscore\HighScore.txt";

    //use the function "File.ReadAllText" for read all the text from the file.
    txtHighScore.Text = File.ReadAllText(filepath);
}

1 reference
private void btnSaveGame_Click(object sender, EventArgs e)
{
    //Stop the timer
    mmStopWatch.Stop();

    //SaveFileDialog allows you to save the file wherever you want
    //the file is saved in a txt file
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.FileName = playerName + "SaveGame"; // Default file name
    dlg.DefaultExt = ".txt"; // Default file extension
    dlg.Filter = "Text documents (*.txt)|*.txt"; // Filter files by extension

    //If u have press "Ok"
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        //creates a variable that prints to a file
        StreamWriter fs = new StreamWriter(dlg.FileName);

        //Print name
        fs.WriteLine(playerName);

        //Saves time
        mmTime = mmStopWatch.Elapsed.Seconds;

        //Print time
        fs.WriteLine(mmTime);

        //Runner MasterMindCode
        foreach (int num in gMasterMind)
        {
            fs.Write(num);
        }
        fs.WriteLine("");

        for (int i = 0; i < mmArray2D.GetLength(0); i++)
        {
            for (int j = 0; j < mmArray2D.GetLength(1); j++)
            {
                fs.Write(mmArray2D[i, j].ToString());
            }
            fs.WriteLine("");
        }
        fs.Close();
        //Closes this form
        this.Close();
    }
    //Starter tiden igen
    mmStopWatch.Start();
}

```

Bilag 8.3 – GameForm.cs – Del 9

```

1 reference
private void ContinueGame()
{
    //Create variabel fd for OpenFileDialog
    OpenFileDialog fd = new OpenFileDialog();

    //Show file dialog
    DialogResult dr = fd.ShowDialog();

    //Continue if user wants to open file
    if (DialogResult.Cancel != dr)
    {
        //Creates a variable for the filename
        string filepath = fd.FileName;

        //lines contains all lines that are read from the file
        string[] lines = File.ReadAllLines(filepath);

        //Setting the playername from the first line
        playerName = lines[0];

        //Setting the extraTime from the second line
        extraTime = Convert.ToDouble(lines[1]);

        //columns contains all values from a row (columns)
        //Subtracting one as I want to look at the last line.
        string columns1 = lines[lines.Length-1];

        //Loads number of length to newly created array.
        //I subtract 3, due to the first 3 lines (Name, time and "gMasterMind")
        mmArray2D = new int[lines.Length - 3, columns1.Length];

        //I start from 2, as I disregard the first two lines
        for (int i = 2; i < lines.Length; i++)
        {
            //print all numbers in line to hall
            string hall = lines[i];

            //Loads numbers to gMasterMind that the player must guess
            //i must be 2, as it is on the 3rd line from the file
            if (i == 2)
            {
                for (int j = 0; j < hall.Length; j++)
                {
                    //Convert char to string and then to int
                    gMasterMind[j] = Convert.ToInt32(hall[j].ToString());
                }
            }
            else
            {
                for (int j = 0; j < hall.Length; j++)
                {
                    //i subtract 3 from i since i have to start at 0
                    mmArray2D[i - 3, j] = Convert.ToInt32(hall[j].ToString());
                }
            }
        }

        //Creates a new 2d array to check that measures by the size of mmarray2d
        checkmmArray2D = new int[mmArray2D.GetLength(0), mmArray2D.GetLength(1)];

        // Examines which row we are now working on
        // starts at the 4th line, and therefore i = 3
        for (int i = 3; i < lines.Length; i++)
        {
            string hall = lines[i];
            int hallint = Convert.ToInt32(hall[3].ToString());
            if (hallint == 0)
            {
                //I subtract 3 due to the first line being deduced.
                numRowArr = i - 3;
                break;
            }
        }
    }

    //Check if you cancel
    if (DialogResult.Cancel == dr)
    {
        //Closing the form
        this.Close();
    }
}

```

Bilag 8.3 – GameForm.cs – Del 10

3 references

```
private void RestartGame()
{
    //Startbutton makes visible
    btnStart.Visible = true;
    //color groupbox makes invisible
    groupBox1.Visible = false;

    //Reset the time
    extraTime = 0;
    mmStopWatch.Reset();

    //Clear my panel (canvas1)
    canvas1.Refresh();

    //Restart time, and color combination and
    //the two 2d arrays that store values from the player's bid
    numRowsArr = 0;
    gMasterMindGuess();
    checkmmArray2D = new int[14, 4];
    mmArray2D = new int[14, 4];

    //drawing the mastermind and checklist
    DrawMasterMind();
    DrawCheck();
}
```

1 reference

```
private void btnRestart_Click(object sender, EventArgs e)
{
    //restarts the game using the method
    RestartGame();
}
```

1 reference

```
private void timer1_Tick(object sender, EventArgs e)
{
    //rounds to 2 decimal places
    mmTime = Math.Round(extraTime + mmStopWatch.Elapsed.TotalSeconds, 2);

    //The timer print
    lblTime.Text = mmTime.ToString() + " sekunder.";
}
```

1 reference

```
private void BtnHowToPlay_Click(object sender, EventArgs e)
{
    //Switch GameForm to HowToPlay-form
    HowToPlayForm HTPF = new HowToPlayForm();
    HTPF.ShowDialog();
}
```

1 reference

```
private void btnExit_Click(object sender, EventArgs e)
{
    //closing the form
    this.Close();
}
```

1 reference

```
private void BtnRed_Click(object sender, EventArgs e)
{
    //Uses the method to update the array after a bid for a color
    GamePlay(1);
}
```


Bilag 8.3 – GameForm.cs – Del 11

```
1 reference
private void btnBlue_Click(object sender, EventArgs e)
{
    . . .
    Gameplay(2);
}

1 reference
private void btnGreen_Click(object sender, EventArgs e)
{
    . . .
    Gameplay(3);
}

1 reference
private void btnPink_Click(object sender, EventArgs e)
{
    . . .
    Gameplay(4);
}

1 reference
private void btnYellow_Click(object sender, EventArgs e)
{
    . . .
    Gameplay(5);
}

1 reference
private void btnOrange_Click(object sender, EventArgs e)
{
    . . .
    Gameplay(6);
}
```

Bilag 8.4 – NumberInArrayClass.cs

[2 references](#)`class` NumberInArrayClass

```
{  
    1 reference  
    public int[,] UpdateArr(int[,] arr, int num, int rownum)  
    {  
        //Makes one for loops looking for a 0 number and  
        //replaces this number with the selected number (num)  
        for (int j = 0; j < arr.GetLength(1); j++)  
        {  
            if (arr[rownum, j] == 0)  
            {  
                //We replace 0 with the number (num)  
                arr[rownum, j] = num;  
                //break for a change has been made  
                break;  
            }  
        }  
        //Return the array  
        return arr;  
    }  
}
```

Bilag 8.5 – UndoArrayClass.cs

2 references

class UndoArrayClass

```
{
    1 reference
    public int[,] UndoArr(int[,] arr, int rownum)
    {
        //Creates a bool variable that checks if you have
        //been through a course without a change
        bool dims = true;

        //Checking if it is not the first index in the row.
        //This ensures that you can only remove numbers
        //within the range you are working on
        if (arr[rownum, 0] != 0)
        {
            //makes forloop that looks for a 0-number
            //GetLenght (1) is the number of columns
            //the forloop starts already at the 1st index,
            //since i is 1
            for (int i = 1; i < arr.GetLength(1); i++)
            {
                //if there is a 0, then we go back an index
                //and replace the number with 0
                if (arr[rownum, i] == 0)
                {
                    //We replace the previous index with 0
                    arr[rownum, i - 1] = 0;
                    //bool is false now, for forloop has been used
                    dims = false;
                    //breaker the course,
                    //for we have finished our task
                    break;
                }
            }
            //if dims is true, then we remove we automatically
            //convert the last index(3) to 0
            if (dims)
            {
                arr[rownum, arr.GetLength(1) - 1] = 0;
            }
        }
        //Return array
        return arr;
    }
}
```

Bilag 8.6 – CountCheck.cs - klasse

```
3 references
class CountCheck
{
    //defining variables
    public int correctlocation;
    public int correctColor;
    public int noCorrect;

    //Class CountCheck being made
    1 reference
    public CountCheck(int[,] arr, int rownums)
    {
        for (int j = 0; j < arr.GetLength(1); j++)
        {
            if (arr[rownums, j] == 2)
            {
                //black
                correctlocation++;
            }
            if (arr[rownums, j] == 1)
            {
                //White
                correctColor++;
            }
            if (arr[rownums, j] == 0)
            {
                //Pen squares
                noCorrect++;
            }
        }
    }

    //Create a method for black
    2 references
    public int Black()
    {
        return correctlocation;
    }

    //Create a method for white
    1 reference
    public int White()
    {
        return correctColor;
    }

    //Create a method for nothing
    1 reference
    public int Nothing()
    {
        return noCorrect;
    }
}
```