

Tetris

**door:
Tim Langens**

1 Inhoudsopgave

| | | |
|----------|-------------------------------|-----------|
| 1 | Inhoudsopgave | 2 |
| 2 | Omschrijving..... | 3 |
| 2.1 | Programmafunctie's..... | 3 |
| 2.2 | programmastructuur..... | 4 |
| 3 | Het hoofdvenster | 5 |
| 3.1 | Het speelveld..... | 6 |
| 3.2 | De blokken..... | 7 |
| 3.2.1 | Makeshape | 7 |
| 3.2.2 | Turn..... | 7 |
| 3.2.3 | Paint | 7 |
| 3.3 | De scorebox | 7 |
| 3.4 | De previewbox | 7 |
| 4 | Opties | 8 |
| 4.1 | Geluid afspelen | 8 |
| 4.2 | Preview | 8 |
| 4.3 | Beginlevel, beginhoogte | 8 |
| 4.4 | Besturing..... | 8 |
| 5 | Highscores..... | 9 |
| 6 | Over..... | 10 |
| 7 | Appendix..... | 12 |
| A | Logboek | 12 |
| B | Bijlagen..... | 14 |
| B.1 | Bronnen..... | 14 |
| B.2 | Codes | 14 |
| B.2.1 | Block.cs..... | 14 |
| B.2.2 | Field.cs..... | 18 |
| B.2.3 | Gescoord.cs..... | 19 |
| B.2.4 | Highscore.cs..... | 20 |
| B.2.5 | Opties.cs..... | 25 |
| B.2.6 | Over.cs | 30 |
| B.2.7 | Tetris.cs..... | 31 |

2 Omschrijving

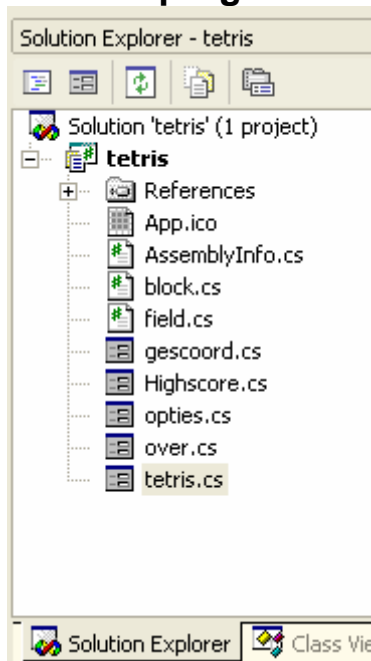
2.1 Programmafunctie's

Het tetrisspel is gemaakt in C# en bevat volgende functie's

- 3 zichtbare vensters:
 - Score
 - Preview
 - Speelveld
- standaardaansturing met pijltoetsen
- spel kunnen pauzeren
- Verschillende levels
- Muziek afspelen
- Opties:
 - Startlevel keuze
 - Voorafingestelde lijnen
 - Muziek aan/uit
 - Preview aan/uit
 - Bediening instellen
- Highscore bijhouden
- Een formulier met informatie over het spel
- De blokken hebben een verschillende kleur
- Elke level heeft een andere achtergrond
- Start van het spel op commando

Het verloop van deze uitbreidingen vind je terug in Appendix A

2.2 programmastructuur



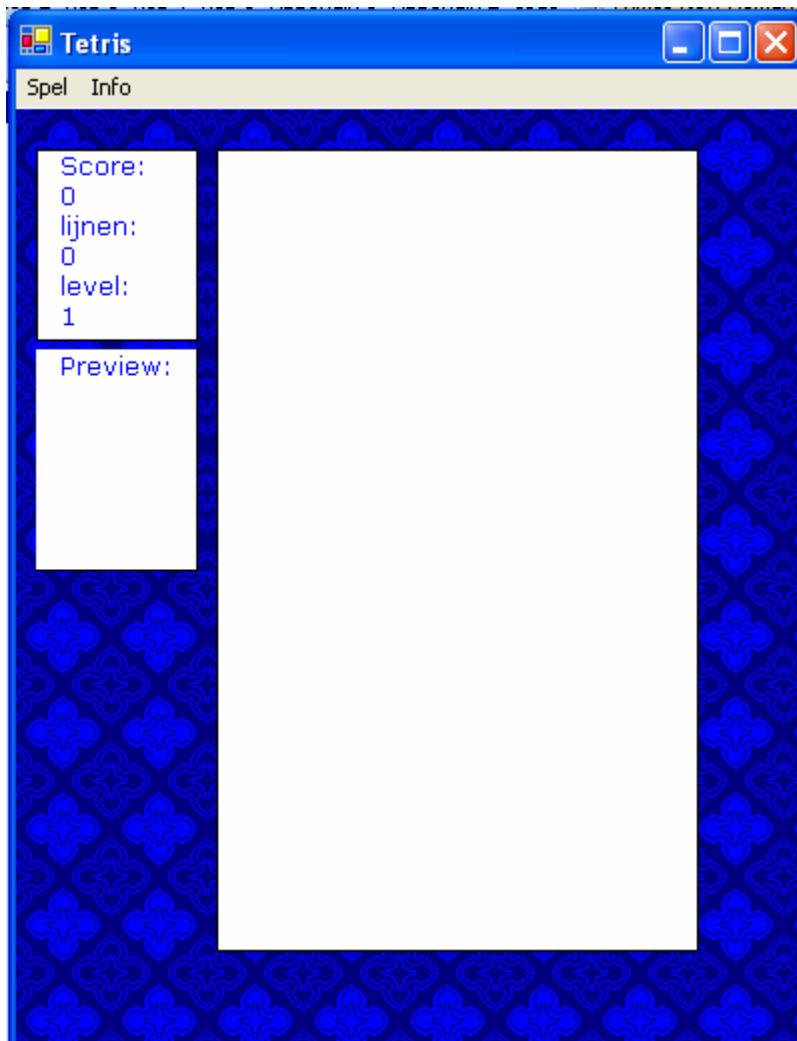
Figuur 1: De programmastructuur in Visual Studio .NET

Het spel bevat 8 verschillende cs-bestanden:

- AssemblyInfo.cs: door Visual studio zelf aangemaakt bestand waarin de eigenschappen van het programma staan
- Block.cs: de klasse block
- Field.cs: de klasse field
- Gescoord.cs: formulier dat verschijnt wanneer je een topscore behaalt hebt
- Highscore.cs: formulier dat de topscores weergeeft
- Opties.cs: formulier dat de opties weergeeft
- Over.cs: formulier dat informatie weergeeft over het programma
- Tetris.cs: hoofdvenster, dit bestand bevat de mainfunctie.

De code van deze bestanden vindt je in Appendix B.

3 Het hoofdvenster



Figuur 2: Grafische weergave van het hoofdvenster

Het hoofdvenster is opgebouwd uit drie delen:

- Speelveld
- Scorebox
- Previewbox

Het hoofdvenster bevat ook een menu, waarin naar de verschillende formulieren wordt verwezen.

Het tekenen van het hoofdvenster gebeurt met de functie OnPaint:

```
protected override void OnPaint( PaintEventArgs e )
{
    // Get Graphics Object
    Graphics g = e.Graphics;

    // Font bepalen
    Font f = new Font( "Verdana", 10 );
```

```
// Brush bepalen
SolidBrush b = new SolidBrush( Color.Blue );

// Teken de scorebox
GraphicsPath scorepath = new GraphicsPath();
Rectangle scorebox = new Rectangle(10,20,80,95);
scorepath.AddRectangle(scorebox);
PathGradientBrush pgbs = new PathGradientBrush(scorepath);
pgbs.SurroundColors = new Color[] { Color.White };
g.FillRectangle(pgbs, scorebox);
g.DrawRectangle(new Pen(Color.Black), scorebox);
g.DrawString("Score:", f, b, 20, 20);
g.DrawString(score.ToString(), f, b, 20, 35);
g.DrawString("lijnen:", f, b, 20, 50);
g.DrawString(level.ToString(), f, b, 20, 65);
g.DrawString("level:", f, b, 20, 80);
g.DrawString(huidiglevel.ToString(), f, b, 20, 95);

// Teken de previewbox
GraphicsPath previewpath = new GraphicsPath();
Rectangle previewbox = new Rectangle(10,120,80,110);
previewpath.AddRectangle(previewbox);
PathGradientBrush pgbp = new PathGradientBrush(previewpath);
pgbp.SurroundColors = new Color[] { Color.White };
g.DrawRectangle(new Pen(Color.Black), previewbox);
g.FillRectangle(pgbp, previewbox);
g.DrawString("Preview:", f, b, 20, 120);
if(preview)
    bBlock.paint( g );

// Teken het field
aField.paint( g );

// Teken het blok
if(!pauze)
    aBlock.paint( g );

// Pauze tekenen
if(pauze & aBlock.shape.Length > 1)
{
    f = new Font("Verdana", 50);
    b.Color = Color.Tomato;
    g.DrawString("Pauze", f, b, LEFT, 100);
}
}
```

In deze functie wordt ook een blok getekent indien er niet gepauzeerd wordt. Wanneer er wel gepauzeerd wordt verdwijnt de getekende blok en verschijnt er de tekst Pauze

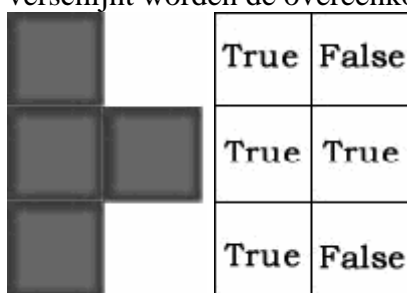
3.1 Het speelveld

Het speelveld bestaat uit een matrix van blokken. Deze blokken zijn booleans die vertellen of er een blok op die plaats staat of niet. Indien de boolean op true staat, zal er een blok getekend worden. Hiervoor is er nog het attribuut kleur, dit attribuut vertelt in welke kleur het blok moet worden getekend.

Het tekenen van het veld gebeurt in de functie Paint.

3.2 De blokken

Tetris heeft 7 verschillende blokken. Het maken van deze blokken gebeurt net zoals het speelveld door matrixen van booleans. Deze matrixen krijgen een bepaalde vorm door de boolean op true te zetten indien het blok zichtbaar is. Wanneer het blok in het veld verschijnt worden de overeenkomstige waarden op true gezet.



| | |
|------|-------|
| True | False |
| True | True |
| True | False |

Figuur 3: Een blokvorm met overeenkomstige matrix van booleans.

De blokken hebben verschillende methoden. Hieronder de belangrijkste:

3.2.1 Makeshape

Geef een andere vorm aan de blokken ipv de standaardvorm, deze methode wordt opgeroepen wanneer de defaultconstructor niet wordt aangeroepen. Hiervoor moet er wel een integer meegegeven worden die de vorm bepaald. Vanuit de hoofdfunctie wordt de vorm bepaald door een random integer.

3.2.2 Turn

Deze functie draait een blok met bepaalde vorm.

3.2.3 Paint

Het tekenen van een blok. Dit gebeurt op dezelfde wijze als het veld tekenen.

3.3 De scorebox

In de scorebox worden 3 verschillende waarden teruggegeven aan de gebruiker:

- Score,
- Aantal lijnen,
- Level

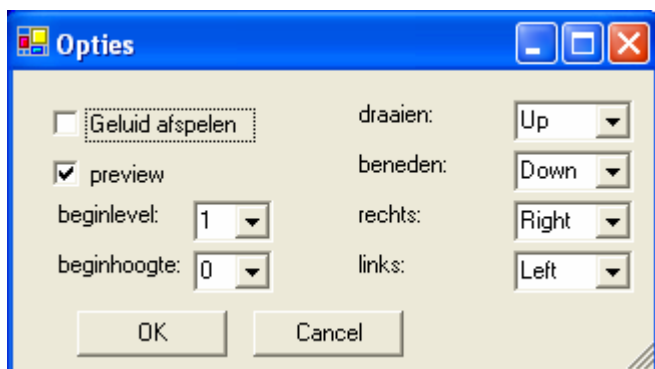
De level wordt bepaald door het aantal lijnen dat de speler heeft behaald en is gelijk aan het tental + 1.

3.4 De previewbox

Hierin staat het volgende blok weergegeven indien deze functie aanstaat. Het programma maakt bij aanvang twee blokken aan. Een blok is het startblok, het andere blok is de volgende blok. Wanneer een blok geplaatst is wordt het volgende blok, het startblok en wordt er een nieuw volgende blok aangemaakt.

4 Opties

De optie's kunnen ingesteld worden in een apart formulier.



Figuur 4: Het optieformulier.

4.1 Geluid afspelen

Om geluid af te spelen is de eenvoudigste oplossing het importeren van de winmm.dll. Deze.dll zorgt voor de multimedia. Het enige probleem is dat wanneer het geluid continu moet afgespeeld worden het programma in een lus kan raken. De oplossing voor dit probleem is door het afspelen asynchroon met ons programma uit te voeren. Deze optie zit mee in de.dll. Dit doe je door aan de functie PlaySound de waarden SND_LOOP en SND_ASYNC mee te geven.

```
PlaySound(Application.StartupPath+"\\tetris.wav",0, SND_LOOP | SND_ASYNC);
```

4.2 Preview

Deze optie bepaald of er een voorbeeldblok moet worden getekend. Hiervoor wordt een boolean meegegeven aan het hoofdformulier, en in de OnPaint functie kijkt men deze waarde na.

4.3 Beginlevel, beginhoogte

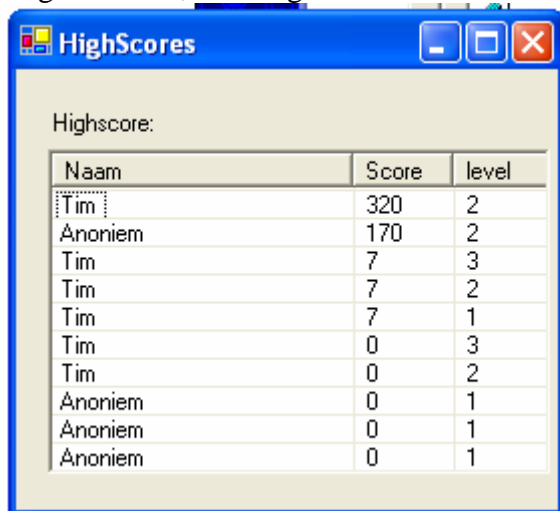
Met deze optie's kan de gebruiker een startlevel en een starthoogte bepalen. Er wordt een integer doorgegeven aan het hoofdprogramma, zodat dit een nieuw spel start met de juiste optie's.

4.4 Besturing

Via de volgende vier comboboxen kan je andere toetsen instellen als besturing. Een vastgelegde waarde verschijnt niet in de keuzemenu's. Wanneer een waarde hiervan gewijzigd wordt, zal deze ook gewijzigd worden in het keuzemenu van de anderen. Dmv. de methoden (upchanged, downchanged, rightchanged, leftchanged

5 Highscores

Wanneer het spel gedaan is wordt de behaalde score vergeleken met de 10 best behaalde scores en indien deze score hoger is wordt deze toegevoegd. Wanneer een score al behaald is zal men de levels vergelijken. De score's worden bijgehouden in het bestand Highscore.txt, dat aangemaakt wordt indien het niet aanwezig is.

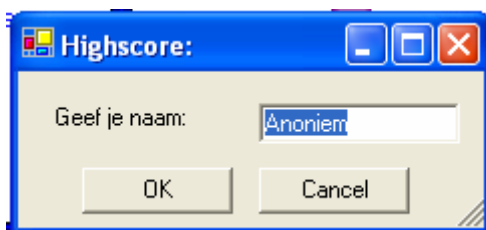


The screenshot shows a dialog box titled "HighScores" with a table containing 10 rows of high scores. The table has three columns: "Naam", "Score", and "level".

| Naam | Score | level |
|---------|-------|-------|
| Tim | 320 | 2 |
| Anoniem | 170 | 2 |
| Tim | 7 | 3 |
| Tim | 7 | 2 |
| Tim | 7 | 1 |
| Tim | 0 | 3 |
| Tim | 0 | 2 |
| Anoniem | 0 | 1 |
| Anoniem | 0 | 1 |
| Anoniem | 0 | 1 |

Figuur 5: Het highscore formulier

Indien er een nieuwe topscore behaald is verschijnt het formulier gescoord.cs waarin men een naam kan meegeven.



The screenshot shows a dialog box titled "Highscore:" with a text input field labeled "Geef je naam:" containing the text "Anoniem". Below the input field are two buttons: "OK" and "Cancel".

Figuur 6: het gescoord formulier

6 Over

Dit formulier geeft informatie over het programma. Het bevat een label met de maker, een linklabel naar de website van de maker.



Figuur 7: Het over-formulier.

7 Problemen

7.1 Het OnPaint probleem

Het grootste probleem dat voorkwam was dat wanneer het formulier hertekent werd met `invalidate()` er steeds flikkeringen optraden. De beste manier om dit te verwijderen was door aan dubbele buffering te doen:

```
// buffer tegen het flikkeren  
this.SetStyle(ControlStyles.UserPaint | ControlStyles.AllPaintingInWmPaint  
| ControlStyles.DoubleBuffer, true);
```

7.2 Plaatsing van de blokken

De verplaatsing van de blokken zorgde voor errors en gebeurde niet correct. Ook het plaatsen van de blokken gebeurde niet correct, sommige bestaande stukjes verdwenen, dit kwam doordat de blok zijn volledige shape in het field zette. Terwijl hij enkel de bools moet schrijven die true zijn.

8 Appendix

A Logboek

Woensdag 12 maart:

- ontwerp grafische omgeving (previewbox, scorebox en speelveld)
- een blok laten bewegen met de pijltoetsen
- opzoekwerk over timer
- de blok automatisch laten vallen
- een nieuwe blok laten vallen, wanneer de blok niet verder kan vallen

Duur: 2 uur

Donderdag 13 maart:

- een blok enkel laten verplaatsen naar een plaats waar geen blok staat
- een lijn detecteren

Duur: 1/2 uur

Vrijdag 15 maart:

- een lijn verwijderen + score bijhouden
- score weergeven
- menu's laten werken

Duur: 2 uur

Vrijdag 31 maart:

- de originele blokken implementeren

Duur: 2uur

Maandag 3 april:

- Er wordt maar 1 lijn verwijderd, geen goede controlestructuur, oplossing andere structuur

Duur: 1 uur

Donderdag 6 april:

- De verplaatsing van de blokken zorgt voor errors en gebeurt niet correct → volledig herschrijven van de functie's.
- Het plaatsen van de blokken gebeurt niet correct, sommige bestaande stukjes verdwijnen, dit komt doordat de blok zijn volledige shape in het field zet. Terwijl hij enkel de bools moet schrijven die true zijn.
- De blokken laten draaien
- Geluid toevoegen

Duur: 3 1/2 uur

Vrijdag 7 april:

- De blokken draaien langs rechts buiten het veld, de locatie van het blok moeten aanpassen indien dit ging gebeuren.
- Het optie's veld toevoegen
- Voorbeeldblok weergeven

Duur: 2 uur

Zaterdag 8 april

- verschillende levels aanmaken

Duur: 1 uur

Zondag 9 april

- keuze aanmaken van de beginlevel.
- Probleem gevonden bij het draaien van een blok.
- De blokken een verschillende kleur geven
- Met een vooraf ingestelde hoogte kunnen beginnen
- Highscore bijhouden
- Starten op commando

Duur: 3 uur

Donderdag 13 april

- aanpassen zodat de lange blok logischer draaien

Duur: 1/2 uur

Vrijdag 14 april

- flikkering van bij het verplaatsen van een blok wegwerken. Mbv. doublebuffering

Duur: 1 uur

Zaterdag 15 april

- De gebruiker toetsen laten toewijzen aan de bewegingen van het blokje

Duur: 1 uur

Zondag 16 april

- Een echte achtergrond aan de levels toewijzen ipv een kleur
- Bij de highscoretabel de bereikte level toevoegen
- Enkele kleine aanpassingen

Duur: 1/2 uur

B Bijlagen

B.1 Bronnen

- http://www.devcity.net/Articles/44/1/gdiplus_tetris.aspx
- <http://www.publicjoe.f9.co.uk/csharp/tut.html>
- <http://www.c-sharpcorner.com>

B.2 Codes

B.2.1 *Block.cs*

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

/// klasse om blocken te tekenen
public class Block
{
    public Point location;           // positie in window
    public int width,height;        // breedte en hoogte
    van blok
    public bool[,] shape;           // vorm van het blok
    public Color kleur;

    // default constructor
    public Block()
    {
        shape = new bool[1,4];
        location = (new Point(10,10));
        width = 1;
        height = 4;
        for(int i = 0; i < 4; i++)
            shape[0,i] = true;
        kleur = Color.Blue;
    }

    // constructor om een block in het veld te tekenen
    public Block( Point p, Color k )
    {
        location = p;
        shape = new bool[1,1];
        width = 1;
        height = 1;
        shape[0,0] = true;
        kleur = k;
    }
}
```

```
// constructor voor nieuw block
public Block( Point p,int i)
{
    MakeShape(i);
    location = p;
}

public void MakeShape(int keuze)
{
    switch(keuze)
    {
        case 1:           // long block
            shape = new bool[1,4];
            width = 1;
            height = 4;
            for(int i = 0; i < 4; i++)
                shape[0,i] = true;
            kleur = Color.Blue;
            break;
        case 2:           // rectangle block
            shape = new bool[2,2];
            width = 2;
            height = 2;
            for(int i = 0; i < 2; i++)
                for(int j = 0; j < 2; j++)
                    shape[j,i] = true;
            kleur = Color.Red;
            break;
        case 3:           // cornerright block
            shape = new bool[2,3];
            width = 2;
            height = 3;
            for(int i = 0; i < height; i++)
                shape[0,i] = true;
            shape[1,0] = false;
            shape[1,1] = false;
            shape[1,2] = true;
            kleur = Color.Yellow;
            break;
        case 4:           // cornerleft block
            shape = new bool[2,3];
            width = 2;
            height = 3;

            shape[0,0] = false;
            shape[0,1] = false;
            shape[0,2] = true;
            for(int i = 0; i < height; i++)
                shape[1,i] = true;
            kleur = Color.Green;
            break;
        case 5:           // cross block
            shape = new bool[2,3];
            width = 2;
            height = 3;
            for(int i = 0; i < 3; i++)
```

```

        shape[0,i] = true;
        shape[1,0] = false;
        shape[1,1] = true;
        shape[1,2] = false;
        kleur = Color.Purple;
        break;
    case 6:          // zigzag leftright block
        shape = new bool[2,3];
        width = 2;
        height = 3;
        shape[0,0] = true;
        shape[1,0] = false;
        shape[0,1] = true;
        shape[1,1] = true;
        shape[0,2] = false;
        shape[1,2] = true;
        kleur = Color.GreenYellow;
        break;
    default:        // zigzag rightleft block
        shape = new bool[2,3];
        width = 2;
        height = 3;
        shape[0,0] = false;
        shape[1,0] = true;
        shape[0,1] = true;
        shape[1,1] = true;
        shape[0,2] = true;
        shape[1,2] = false;
        kleur = Color.Lime;
        break;
    }
}

public void setLocation( Point p ) { location = p; }

// methode dat de attributen weergeeft van het block
public Point getLocation() { return location; }

//methode om de block te tekenen
public void paint( Graphics g )
{
    for (int i = 0; i < width; i ++ )
    {
        for (int j = 0; j < height; j++)
        {
            if (shape[i,j] == true)
            {
                Rectangle rect = new Rectangle(location.X +
(i*20),location.Y + (j * 20),20,20);
                DrawBlock( g, rect, kleur );
            }
        }
    }
}

public void DrawBlock( Graphics grfx, Rectangle rect, Color c )
{

```

```
GraphicsPath path = new GraphicsPath();

path.AddRectangle(rect);

PathGradientBrush pgbrush = new PathGradientBrush(path);
pgbrush.SurroundColors = new Color[] { c };

grfx.FillRectangle( pgbrush,rect );
}

//methode om de block te verplaatsen
public void moveTo( int x, int y )
{
    location.X = x;
    location.Y = y;
}

//methode om de block te draaien
public void Turn()
{
    int h;
    bool[,] b = new bool[width, height];

    h = height;
    height = width;
    width = h;
    b = shape;
    shape = new bool[width,height];
    if(!(height == 2 & width == 2))
    {
        if(height == 4 | width == 4)
        {
            for(int i = 0; i < width; i++)
                for(int j =0; j < height; j++)
                    shape[i,j] = b[j,i];
            if(height < width)
                location = new Point(location.X - 20,
location.Y + 20);
            else
                location = new Point(location.X + 20,
location.Y - 20);
        }
        else
        {
            if(height == 2)
            {
                shape[0,0] = b[1,0];
                shape[0,1] = b[0,0];
                shape[1,0] = b[1,1];
                shape[1,1] = b[0,1];
                shape[2,0] = b[1,2];
                shape[2,1] = b[0,2];
            }
            else
            {
                shape[0,0] = b[2,0];
                shape[0,1] = b[1,0];
            }
        }
    }
}
```

```

        shape[0,2] = b[0,0];
        shape[1,0] = b[2,1];
        shape[1,1] = b[1,1];
        shape[1,2] = b[0,1];
    }
}
else
    shape = b;
}
}

```

B.2.2 Field.cs

```

using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

/// klasse om blokken te tekenen
public class Field
{
    public const int WIDTH = 12;
    public const int HEIGHT = 20;
    public const int UP = 20;
    public const int LEFT = 100;
    public Point location; // positie in window
    private Size grootte; // grootte van het veld
    public bool[,] blocks = new bool[WIDTH,HEIGHT]; // waar staan
    // er blokken op het veld?
    public Color[,] kleur = new Color[WIDTH,HEIGHT]; // welke kleur
    // hebben de blokken

    // default constructor
    public Field()
    {
        location = (new Point(10,10));
        grootte = (new Size(WIDTH * 20, HEIGHT * 20));
        for(int i = 0; i < WIDTH; i++)
            for(int j = 0; j < HEIGHT ;j++)
            {
                blocks[i,j] = false;
                kleur[i,j] = Color.Blue;
            }
    }

    // constructor voor nieuw veld
    public Field( Point p)
    {
        location = p;
        grootte = (new Size(WIDTH * 20, HEIGHT * 20));
        for(int i = 0; i < WIDTH; i++)
            for(int j = 0; j < HEIGHT ;j++)
            {
                blocks[i,j] = false;
                kleur[i,j] = Color.Blue;
            }
    }
}

```

```

    }

    // methode om de attributen te veranderen
    public void setLocation( Point p ) { location = p; }
    public void setBlockX( int x, int y, bool b) { blocks[x,y] = b; }

    // methode dat de attributen weergeeft van het field
    public Point getLocation() { return location; }
    public Size getSize() { return grootte; }
    public bool getBlockX( int x, int y) { return blocks[x,y]; }

    //methode om het field te tekenen
    public void paint( Graphics g )
    {
        // Draw the field
        GraphicsPath fieldpath = new GraphicsPath();
        Rectangle field = new Rectangle(location,grootte);
        fieldpath.AddRectangle(field);
        PathGradientBrush pgbf = new PathGradientBrush(fieldpath);
        pgbf.SurroundColors = new Color[] { Color.White };
        g.FillRectangle(pgbf,field);
        g.DrawRectangle(new Pen(Color.Black),field);
        for(int i=0; i < WIDTH; i++)
            for(int j=0; j < HEIGHT; j++)
                if(blocks[i,j])
                {
                    Block b = new Block(new Point( (i*20) +
LEFT, (j*20) + UP),kleur[i,j]);
                    b.paint( g );
                }
    }
}

```

B.2.3 Gescoord.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

public class Gescoord : Form
{
    private Label label1;
    public TextBox textBox1;
    private Button button1;
    private Button button2;

    private System.ComponentModel.Container components = null;

    public Gescoord()
    {
        InitializeComponent();
    }

    // Resources opruimen
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {

```

```
        if(components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

private void InitializeComponent()
{
    label1 = new Label();
    textBox1 = new TextBox();
    button1 = new Button();
    button2 = new Button();

    // label1
    label1.Location = new Point(16, 16);
    label1.Name = "label1";
    label1.Text = "Geef je naam:";

    // textBox1
    textBox1.Location = new Point(120, 16);
    textBox1.Name = "textBox1";
    textBox1.TabIndex = 0;
    textBox1.Text = "Anoniem";
    textBox1.Focus();

    // button1
    button1.DialogResult = DialogResult.OK;
    button1.Location = new Point(32, 48);
    button1.TabIndex = 1;
    button1.Name = "button1";
    button1.Text = "OK";

    // button2
    button2.DialogResult = DialogResult.Cancel;
    button2.Location = new Point(120, 48);
    button2.Name = "button2";
    button2.TabIndex = 2;
    button2.Text = "Cancel";

    // name
    ClientSize = new Size(232, 78);
    Controls.Add(button2);
    Controls.Add(button1);
    Controls.Add(textBox1);
    Controls.Add(label1);
    this.Text = "Highscore:";
}
}
```

B.2.4 Highscore.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
```

```
using System.Text;
using System.Collections;

public class HighScoreEntry
{
    public string name;
    public int score;
    public int level;

    public HighScoreEntry( string name, int score, int level )
    {
        this.name = name;
        this.score = score;
        this.level = level;
    }
}

public class HighScoreTable
{
    private ArrayList table;

    private bool isLoading;

    public HighScoreTable()
    {
        table = new ArrayList();
        isLoading = false;
    }

    public void Load( string path )
    {
        if( File.Exists( path ) )
        {
            table = new ArrayList();

            using( StreamReader textStream = new StreamReader(path,
Encoding.UTF8) )
            {
                string scoreLine;

                while( (scoreLine = textStream.ReadLine()) != null
)
                {
                    string[] scoreParts = scoreLine.Split(',');

                    if( scoreParts.Length != 3 )
                    {
                        throw new ApplicationException("Score
file corrupt!");
                    }
                    else
                    {
                        table.Add(new
HighScoreEntry(scoreParts[0], Int32.Parse(scoreParts[1]), Int32.Parse(scoreP
arts[2])));
                    }
                }
            }
        }
    }
}
```

```
        }
        textStream.Close();
    }
}
else
{
    for( int index = 0; index < 10; index++ )
    {
        table.Add( new HighScoreEntry("Anoniem",0,1));
    }
    Save( path );
}
isLoading = true;
}

public void Save( string path )
{
    if( File.Exists( path ) )
    {
        File.Delete( path );
    }

    using( StreamWriter textStream = new StreamWriter(path,false,
Encoding.UTF8) )
    {
        foreach( object tableEntry in table )
        {
            HighScoreEntry highScoreEntry = tableEntry as
HighScoreEntry;

            textStream.WriteLine("{0},{1},{2}",highScoreEntry.name,highScoreEntr
y.score, highScoreEntry.level );
        }
        textStream.Close();
    }
}

public int GetIndexOfScore( int score, int level )
{
    for( int index = 0; index < table.Count; index++ )
    {
        HighScoreEntry highScoreEntry = table[index] as
HighScoreEntry;

        if( score > highScoreEntry.score && index < 10)
            return index;
        else
            if( level > highScoreEntry.level && score >=
highScoreEntry.score & index < 10)
                return index;
    }
    return -1;
}

public void Update( string name, int score, int level )
{
```

```
        if( !isLoading ) Load(
Application.StartupPath+"\\highscore.txt" );

        int index = GetIndexOfScore( score, level );

        if( index > -1 )
        {
            if( table.Count > 9 ) table.RemoveAt( 9 );
            table.Insert( index, new HighScoreEntry( name, score,
level ) );
            Save( Application.StartupPath+"\\highscore.txt" );
        }
    }

    public void Populate( ListView listView )
    {
        listView.Items.Clear();

        foreach( object entry in table )
        {
            HighScoreEntry highScoreEntry = entry as HighScoreEntry;
            listView.Items.Add( new ListViewItem( new string[] {
highScoreEntry.name, highScoreEntry.score.ToString() ,
highScoreEntry.level.ToString()} ) );
        }
    }
}

public class HighScore : Form
{
    private ListView listView;
    private ColumnHeader naamHeader;
    private ColumnHeader scoreHeader;
    private ColumnHeader levelHeader;
    private HighScoreTable highScoreTable = null;
    private Label label;

    private System.ComponentModel.Container components = null;

    public HighScore( HighScoreTable highScoreTableReference )
    {
        InitializeComponent();
        highScoreTable = highScoreTableReference;
    }

    //resources opruimen
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if (components != null)
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }
}
```

```
}

private void InitializeComponent()
{
    this.listView = new ListView();
    this.naamHeader = new ColumnHeader();
    this.scoreHeader = new ColumnHeader();
    this.levelHeader = new ColumnHeader();
    this.label = new Label();

    // label
    label.Text = "Highscore:";
    label.Size = new Size(210,15);
    label.Location = new Point(16,20);

    // listViewC

    listView.Columns.AddRange(new ColumnHeader[] {naamHeader,
scoreHeader, levelHeader});
    listView.GridLines = true;
    listView.Location = new Point(16,40);
    listView.Name = "listView1";
    listView.Scrollable = false;
    listView.Size = new Size(250, 162);
    listView.TabIndex = 0;
    listView.View = View.Details;

    //
    // naamheader
    //
    naamHeader.Text = "Naam";
    naamHeader.Width = 150;

    //
    // scoreheader
    //
    scoreHeader.Text = "Score";
    scoreHeader.Width = 50;

    //
    // levelheader
    //
    levelHeader.Text = "level";
    levelHeader.Width = 50;

    //
    // HighScore
    //
    this.ClientSize = new Size(270, 220);
    this.Controls.Add( listView );
    this.Controls.Add( label );
    this.Text = "HighScores";

    // Now Load Scores
    this.Load += new System.EventHandler( HighScoreForm_Load );

    this.ResumeLayout( false );
}
```

```

    }

    private void HighScoreForm_Load( object sender, System.EventArgs e )
    {
        highScoreTable.Load( Application.StartupPath+"\\highscore.txt"
    );
        highScoreTable.Populate( listView );
    }
}

```

B.2.5 Opties.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

public class Opties : Form
{
    public      CheckBox muziek;
    public      CheckBox preview;
    public      ComboBox level;
    public      ComboBox lijn;
    public      ComboBox up;
    public      ComboBox down;
    public      ComboBox right;
    public      ComboBox left;
    private Label levelname;
    private Label lijnname;
    private Label draainame;
    private Label downname;
    private Label rightname;
    private Label leftname;
    private Button button1;
    private Button button2;
    private string upkey,downkey,leftkey,rightkey;
    private string [] keys =
{ "Up", "Down", "Right", "Left", "Space", "Enter", "A", "B", "C", "D", "E", "F", "G", "H",
  "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z"
};

    private System.ComponentModel.Container components = null;

    public Opties( bool muplay, bool ex, int niveau, int hoog, string up,
string down, string right, string left)
    {
        InitializeComponent( muplay, ex, niveau, hoog, up, down, left, right );
    }

    // Resources opruimen
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if( components != null )
            {
                components.Dispose();
            }
        }
    }
}

```

```

    }
    base.Dispose( disposing );
}

private void InitializeComponent( bool muplay, bool ex, int
niveau, int hoog, string op, string near, string links, string rechts)
{
    muziek = new CheckBox();
    preview = new CheckBox();
    level = new ComboBox();
    lijn = new ComboBox();
    up = new ComboBox();
    down = new ComboBox();
    right = new ComboBox();
    left = new ComboBox();
    levelname = new Label();
    lijnname = new Label();
    draainame = new Label();
    downname = new Label();
    rightname = new Label();
    leftname = new Label();
    button1 = new Button();
    button2 = new Button();

    // muziek
    muziek.Location = new Point(20, 15);
    muziek.Name = "muziek";
    muziek.TabIndex = 0;
    muziek.Text = "Geluid afspelen";
    muziek.Focus();
    muziek.Checked = muplay;

    // preview
    preview.Location = new Point(20, 40);
    preview.Name = "preview";
    preview.TabIndex = 1;
    preview.Text = "preview";
    preview.Checked = ex;

    //level
    level.Location = new Point(90,65);
    level.Size = new Size(40,20);
    level.Name = "level";
    level.TabIndex = 2;
    level.Text = niveau.ToString();
    string [] levels = { "1", "2", "3", "4", "5", "6", "7", "8",
"9", "10" };
    level.Items.AddRange(levels);

    //lijn
    lijn.Location = new Point(90,90);
    lijn.Size = new Size(40,20);
    lijn.Name = "lijn";
    lijn.TabIndex = 3;
    lijn.Text = hoog.ToString();
    string [] lijnen = { "0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "10", "11", "12", "13", "14", "15" };

```

```
lijn.Items.AddRange(lijnen);

//up
up.Location = new Point(250,15);
up.Size = new Size(60,20);
up.Name = "up";
up.TabIndex = 4;
up.Text = op;
up.Items.AddRange(keys);
up.TextChanged += new System.EventHandler(this.upchanged);

//down
down.Location = new Point(250,40);
down.Size = new Size(60,20);
down.Name = "down";
down.TabIndex = 5;
down.Text = neer;
down.Items.AddRange(keys);
down.TextChanged += new System.EventHandler(this.downchanged);

//right
right.Location = new Point(250,65);
right.Size = new Size(60,20);
right.Name = "right";
right.TabIndex = 6;
right.Text = rechts;
right.Items.AddRange(keys);
right.TextChanged += new
System.EventHandler(this.rightchanged);

//left
left.Location = new Point(250,90);
left.Size = new Size(60,20);
left.Name = "left";
left.TabIndex = 7;
left.Text = links;
left.Items.AddRange(keys);
left.TextChanged += new System.EventHandler(this.leftchanged);

//levellabel
levelname.Location = new Point(20,65);
levelname.Text = "beginlevel:";

//lijnlabel
lijnname.Location = new Point(20,90);
lijnname.Text = "beginhoogte:";

//draailabel
draainame.Location = new Point(170,15);
draainame.Text = "draaien:";

//downlabel
downname.Location = new Point(170,40);
downname.Text = "beneden:";

//rightlabel
rightname.Location = new Point(170,65);
```

```
rightname.Text = "rechts:";

//leftlabel
leftname.Location = new Point(170,90);
leftname.Text = "links:";

// button1
button1.DialogResult = DialogResult.OK;
button1.Location = new Point(32, 120);
button1.TabIndex = 8;
button1.Name = "button1";
button1.Text = "OK";

// button2
button2.DialogResult = DialogResult.Cancel;
button2.Location = new Point(120, 120);
button2.Name = "button2";
button2.TabIndex = 9;
button2.Text = "Cancel";

//bestaande waarden verwijderen
down.Items.Remove(up.Text);
left.Items.Remove(up.Text);
right.Items.Remove(up.Text);
up.Items.Remove(down.Text);
left.Items.Remove(down.Text);
right.Items.Remove(down.Text);
up.Items.Remove(right.Text);
down.Items.Remove(right.Text);
left.Items.Remove(right.Text);
up.Items.Remove(left.Text);
down.Items.Remove(left.Text);
right.Items.Remove(left.Text);

//vorige keywaarden onthouden
upkey = up.Text;
downkey = down.Text;
leftkey = left.Text;
rightkey = right.Text;

// formulier
ClientSize = new Size(320, 150);
Controls.Add(button2);
Controls.Add(button1);
Controls.Add(muziek);
Controls.Add(preview);
Controls.Add(level);
Controls.Add(levelname);
Controls.Add(lijn);
Controls.Add(lijnname);
Controls.Add(up);
Controls.Add(draainame);
Controls.Add(down);
Controls.Add(downname);
Controls.Add(right);
Controls.Add(rightname);
```

```
        Controls.Add(left);
        Controls.Add(leftname);
        this.Text = "Opties";
    }

    public void upchanged(object sender, System.EventArgs e)
    {
        down.Items.Clear();
        down.Items.AddRange(keys);
        left.Items.Clear();
        left.Items.AddRange(keys);
        right.Items.Clear();
        right.Items.AddRange(keys);

        //nieuwe waarde verwijderen
        down.Items.Remove(up.Text);
        down.Items.Remove(left.Text);
        down.Items.Remove(right.Text);
        left.Items.Remove(up.Text);
        left.Items.Remove(down.Text);
        left.Items.Remove(right.Text);
        right.Items.Remove(up.Text);
        right.Items.Remove(down.Text);
        right.Items.Remove(left.Text);
    }

    public void downchanged(object sender, System.EventArgs e)
    {
        //oude waarde herinlezen
        up.Items.Clear();
        up.Items.AddRange(keys);
        left.Items.Clear();
        left.Items.AddRange(keys);
        right.Items.Clear();
        right.Items.AddRange(keys);

        //nieuwe waarde verwijderen
        up.Items.Remove(down.Text);
        up.Items.Remove(left.Text);
        up.Items.Remove(right.Text);
        left.Items.Remove(up.Text);
        left.Items.Remove(down.Text);
        left.Items.Remove(right.Text);
        right.Items.Remove(up.Text);
        right.Items.Remove(down.Text);
        right.Items.Remove(left.Text);
    }

    public void rightchanged(object sender, System.EventArgs e)
    {
        //oude waarde opnieuw toevoegen
        up.Items.Clear();
        up.Items.AddRange(keys);
        down.Items.Clear();
        down.Items.AddRange(keys);
        left.Items.Clear();
        left.Items.AddRange(keys);
    }
}
```

```

        //nieuwe waarde verwijderen
        up.Items.Remove(down.Text);
        up.Items.Remove(left.Text);
        up.Items.Remove(right.Text);
        down.Items.Remove(up.Text);
        down.Items.Remove(left.Text);
        down.Items.Remove(right.Text);
        left.Items.Remove(up.Text);
        left.Items.Remove(down.Text);
        left.Items.Remove(right.Text);
    }

    public void leftchanged(object sender, System.EventArgs e)
    {
        up.Items.Clear();
        up.Items.AddRange(keys);
        down.Items.Clear();
        down.Items.AddRange(keys);
        right.Items.Clear();
        right.Items.AddRange(keys);

        //nieuwe waarde verwijderen
        up.Items.Remove(down.Text);
        up.Items.Remove(left.Text);
        up.Items.Remove(right.Text);
        down.Items.Remove(up.Text);
        down.Items.Remove(left.Text);
        down.Items.Remove(right.Text);
        right.Items.Remove(up.Text);
        right.Items.Remove(down.Text);
        right.Items.Remove(left.Text);
    }
}

```

B.2.6 Over.cs

```

// Namespace Declaratie
using System;
using System.Drawing;
using System.Windows.Forms;

// Class Declaratie
class Over : Form
{
    // Label aanmaken
    private Label labell;
    private LinkLabel link1;
    private PictureBox picture1 = new PictureBox();

    // Constructor
    public Over()
    {
        // Titelbalk tekst
        this.Text = "Over";

        // Lettertypen aanpassen
        this.Font = new Font( "MS Sans Serif", 10 );
    }
}

```

```

        // picture1 initialisatie
        picture1.Image =
System.Drawing.Image.FromFile(Application.StartupPath+"\\picture.jpg");
        picture1.Location = new Point(10,10);
        picture1.Size = new Size(80,80);

        // labell1 initialisatie
        labell1 = new Label();
        labell1.Location = new Point( 100,10 );
        labell1.Size = new Size( 298, 80 );
        labell1.Text = "Ontworpen door:\n Tim Langens";
        labell1.TextAlign = ContentAlignment.MiddleLeft;

        // link1 initializatie
        link1 = new LinkLabel();
        link1.Location = new Point( 10, 90 );
        link1.Size = new Size( 298, 30);
        link1.Text = "http://users.pandora.be/langens/ikke";
        link1.TextAlign = ContentAlignment.MiddleLeft;

        // event toevoegen voor link1
        link1.LinkClicked += new
LinkLabelLinkClickedEventHandler(this.link1_Clicked);

        // labels aan het formulier toevoegen
        this.Controls.Add( labell1 );
        this.Controls.Add( link1 );
        this.Controls.Add( picture1);
    }

    // link1 linken
    protected void link1_Clicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
        link1.LinkVisited = true;
        System.Diagnostics.Process.Start(
"http://users.telenet.be/langens/ikke" );
    }
}

```

B.2.7 Tetris.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Drawing.Drawing2D;

public class tetris : Form
{
    private System.ComponentModel.Container components = null;

    // constanten
    public const int UP = 20;
    public const int DOWN = UP + 380;
    public const int LEFT = 100;
    public const int RIGHT = LEFT + 220;
}

```

```

        public const int STARTX = LEFT + 100; // x waarde voor het punt waar
de nieuwe blok begint
        public const int STARTY = UP; // y waarde voor het punt waar
de nieuwe blok begint
        public const int WIDTH = 12; // De breedte van het speelveld
        public const int HEIGHT = 20; // De hoogte van het speelveld
        public const int VBX = 30;
        public const int VBY = 140;
        public const int SND_LOOP = 0x0008; // loop the sound
until next sndPlaySound
        public const int SND_ASYNC = 0x0001; // play asynchronously

// variabelen
private MainMenu mainMenu;
private ContextMenu contextMenu;
private MenuItem menuItem1;
private Block aBlock, bBlock;
private Timer aTimer;
private Field aField = new Field( new Point( LEFT, UP ));
private int score;
private int level = 0; //gaat omhoog per lijn --> per 10 lijnen
level hoger
private int huidiglevel = 1;
private int beginlevel = 1; // zegt welke level er is ingegeven in
het optieslevel
private int beginlijn = 0;
private bool pauze, muziek = false, preview = true, gameover = true;
private HighScoreTable highScoreTable = new HighScoreTable();
private string up = "Up", down = "Down", right = "Right", left =
"Left";

[DllImport("winmm.dll")]
private static extern bool PlaySound( string lpszName, int hModule,
int dwFlags);

// Constructor
public tetris()
{
    InitializeComponent();

    // Main Menu bouwen
    MenuItem Game = mainMenu.MenuItems.Add("&Spel");
    MenuItem Info = mainMenu.MenuItems.Add("&Info");
    Game.MenuItems.Add( new MenuItem("&Nieuw", new
EventHandler(this.Nieuw_Clicked), Shortcut.F2));
    Game.MenuItems.Add( new MenuItem("&Pauze", new
EventHandler(this.Pauze_Clicked), Shortcut.F3));
    Game.MenuItems.Add( new MenuItem("-"));
    Game.MenuItems.Add( new MenuItem("&Highscore", new
EventHandler(this.Highscore_Clicked), Shortcut.CtrlH));
    Game.MenuItems.Add( new MenuItem("&Opties", new
EventHandler(this.Opties_Clicked), Shortcut.CtrlO));
    Game.MenuItems.Add( new MenuItem("-"));
    Game.MenuItems.Add( new MenuItem("&Afsluiten", new
EventHandler(this.Sluiten)));
    Info.MenuItems.Add( new MenuItem("&Over", new
EventHandler(this.Over_Clicked), Shortcut.CtrlA));

```

```

        // opmaak van het formulier
        this.Text = "Tetris";
        this.Size = new Size(400,500);
        this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level1.bmp");

        // muziek afspelen
        if(muziek)

                PlaySound(Application.StartupPath+"\\tetris.wav",0,
SND_LOOP | SND_ASYNC);

        // highscore laden
        highScoreTable.Load( Application.StartupPath+"\\highscore.txt"
);

        // buffer tegen het flikkeren
        this.SetStyle(ControlStyles.UserPaint |
ControlStyles.AllPaintingInWmPaint | ControlStyles.DoubleBuffer, true);
    }

    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if( components != null )
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

    // Initialiseer form componenten
    private void InitializeComponent()
    {
        Random r = new Random(unchecked((int)DateTime.Now.Ticks));
        this.components = new System.ComponentModel.Container();
        this.aTimer = new Timer(this.components);

        // aTimer
        this.aTimer.Enabled = false;
        this.aTimer.Interval = 500;
        this.aTimer.Tick += new System.EventHandler(this.aTimer_tick);

        // mainMenu initialiseren
        mainMenu = new MainMenu();
        this.Menu = mainMenu;

        // contextMenu initialisatie
        this.contextMenu1 = new ContextMenu();
        this.menuItem1 = new MenuItem();

        // contextMenu1
        this.contextMenu1.MenuItems.AddRange(new MenuItem[]
{this.menuItem1});

```

```

        // menuItem1
        this.menuItem1.Index = 0;
        this.menuItem1.Text = "Over";

        // ContextMenu
        this.AutoScaleBaseSize = new Size(5, 13);
        this.ClientSize = new Size(292, 266);
        this.ContextMenu = this.contextMenu1;

        //initialiseer blocks
        aBlock = new Block( new Point( STARTX,STARTY ), Color.White );
        bBlock = new Block( new Point( VBX,VBY ), Color.White);

        //zet pauze aan
        pauze = true;
    }

    protected override void OnPaint( PaintEventArgs e )
    {
        // Get Graphics Object
        Graphics g = e.Graphics;

        // Font bepalen
        Font f = new Font( "Verdana", 10 );

        // Brush bepalen
        SolidBrush b = new SolidBrush( Color.Blue );

        // Teken de scorebox
        GraphicsPath scorepath = new GraphicsPath();
        Rectangle scorebox = new Rectangle(10,20,80,95);
        scorepath.AddRectangle(scorebox);
        PathGradientBrush pgbs = new PathGradientBrush(scorepath);
        pgbs.SurroundColors = new Color[] { Color.White };
        g.FillRectangle(pgbs,scorebox);
        g.DrawRectangle(new Pen(Color.Black),scorebox);
        g.DrawString("Score:",f,b,20,20);
        g.DrawString(score.ToString(),f,b,20,35);
        g.DrawString("lijnen:",f,b,20,50);
        g.DrawString(level.ToString(),f,b,20,65);
        g.DrawString("level:",f,b,20,80);
        g.DrawString(huidiglevel.ToString(),f,b,20,95);

        // Teken de previewbox
        GraphicsPath previewpath = new GraphicsPath();
        Rectangle previewbox = new Rectangle(10,120,80,110);
        previewpath.AddRectangle(previewbox);
        PathGradientBrush pgbp = new PathGradientBrush(previewpath);
        pgbp.SurroundColors = new Color[] { Color.White };
        g.DrawRectangle(new Pen(Color.Black),previewbox);
        g.FillRectangle(pgbp,previewbox);
        g.DrawString("Preview:",f,b,20,120);
        if(preview)
            bBlock.paint( g );

        // Teken het field
    }

```

```
aField.paint( g );

// Teken het blok
if(!pauze)
    aBlock.paint( g );

// Pauze tekenen
if(pauze & aBlock.shape.Length > 1)
{
    f = new Font("Verdana",50);
    b.Color = Color.Tomato;
    g.DrawString("Pauze",f,b,LEFT,100);
}
}

// de block verplaatsen

private void aTimer_tick(object sender, System.EventArgs e)
{
    if(!gameover)
        MoveDown();
    else
        gedaan();
}

protected override void OnKeyDown(KeyEventArgs e)
{
    if(!pauze)
    {
        string strKeyPress = e.KeyCode.ToString();
        if(down == strKeyPress)
        {
            MoveDown();
        }
        else
        {
            if(up == strKeyPress)
            {
                Draai();
            }
            else
            {
                if(left == strKeyPress)
                {
                    MoveLeft();
                }
                else
                {
                    if(right == strKeyPress)
                    {
                        MoveRight();
                    }
                }
            }
        }
    }
    base.OnKeyDown(e);
}
```

```

    }
}

public void MoveDown()
{
    bool move = true;
    int i = 0, j = 0;

    if( (aBlock.getLocation().Y + ((aBlock.height-1)*20)) < DOWN )
    {
        while(move & i < aBlock.height)
        {
            while(j < aBlock.width)
            {
                if (aField.blocks[(((aBlock.location.X -
LEFT) / 20) + j), (((aBlock.location.Y - UP) / 20) + i + 1)] &&
aBlock.shape[j,i])
                    move = false;
                j++;
            }
            j = 0;
            i++;
        }
    }
    else
        move = false;

    if(move)
    {
        aBlock.moveTo( aBlock.getLocation().X,
aBlock.getLocation().Y + 20 );
        Invalidate(new Rectangle(aBlock.getLocation().X,
aBlock.getLocation().Y - 20, aBlock.width * 20, (aBlock.height + 1) * 20));
    }
    else
    {
        if(!GameOver())
            nieuwblok();
        else
            gedaan();
    }
}

public void MoveRight()
{
    bool move = true;
    int i = 0, j = 0;

    if( aBlock.location.X < (RIGHT - ((aBlock.width - 1) * 20) ) )
    {
        while(move & i < aBlock.height)
        {
            while(j < aBlock.width)
            {

```

```

        if (aField.blocks[(((aBlock.location.X -
LEFT) / 20) + j + 1), (((aBlock.location.Y - UP) / 20) + i)] &
aBlock.shape[j,i])
            move = false;
            j++;
        }
        j = 0;
        i++;
    }
}
else
    move = false;

    if(move)
    {
        aBlock.moveTo( aBlock.getLocation().X + 20,
aBlock.getLocation().Y );
        Invalidate(new Rectangle(aBlock.getLocation().X - 20,
aBlock.getLocation().Y,(aBlock.width + 1) * 20,(aBlock.height) * 20));
    }
}

public void MoveLeft()
{
    bool move = true;
    int i = 0, j = 0;

    if( aBlock.location.X > LEFT )
    {
        while(move & i < aBlock.height)
        {
            while(j < aBlock.width)
            {
                if (aField.blocks[(((aBlock.location.X -
LEFT) / 20) + j - 1), (((aBlock.location.Y - UP) / 20) + i)] &
aBlock.shape[j,i])
                    move = false;
                    j++;
            }
            j = 0;
            i++;
        }
    }
    else
        move = false;

    if(move)
    {
        aBlock.moveTo( aBlock.getLocation().X - 20,
aBlock.getLocation().Y );
        Invalidate(new Rectangle(aBlock.getLocation().X,
aBlock.getLocation().Y,(aBlock.width + 1) * 20, aBlock.height * 20));
    }
}

public void Draai()

```

```

    {
        bool draai = true;
        if(aBlock.height != aBlock.width)
        {
            if(aBlock.height == 4)
            {
                if((aBlock.location.X <= (RIGHT - 40)) &
(aBlock.location.X >= (LEFT + 20)))
                {
                    for(int i = (((aBlock.location.X - LEFT) /
20) - 1); i < (((aBlock.location.X - LEFT) / 20) + aBlock.height - 1) &
draai;i++)
                        for(int j = ((aBlock.location.Y - UP)
/ 20 + 1); j < (((aBlock.location.Y - UP) / 20) + aBlock.height + 1) &
draai;j++)
                            if(aField.blocks[i,j])
                                draai = false;
                    if(draai)
                    {
                        aBlock.Turn();
                        Invalidate( new
Rectangle(aBlock.location.X, aBlock.location.Y - 20, aBlock.width * 20,
aBlock.width * 20));
                    }
                }
            }
            else
            {
                if((aBlock.location.X <= (RIGHT - ((aBlock.height-
1)*20))) & (aBlock.location.Y <= (DOWN - ((aBlock.width-1)*20))))
                {
                    if(aBlock.height > aBlock.width)
                    {
                        for(int i = ((aBlock.location.X -
LEFT) / 20); i < (((aBlock.location.X - LEFT) / 20) + aBlock.height) &
draai;i++)
                            for(int j = ((aBlock.location.Y
- UP) / 20); j < (((aBlock.location.Y - UP) / 20) + aBlock.height) &
draai;j++)
                                if(aField.blocks[i,j])
                                    draai = false;
                    if(draai)
                    {
                        aBlock.Turn();
                        Invalidate( new
Rectangle(aBlock.location.X, aBlock.location.Y, aBlock.width * 20,
aBlock.width * 20));
                    }
                }
            }
            else
            {
                for(int i = ((aBlock.location.X -
LEFT) / 20); i < (((aBlock.location.X - LEFT) / 20) + aBlock.width) &
draai;i++)

```

```

        for(int j = ((aBlock.location.Y
- UP) / 20); j < (((aBlock.location.Y - UP) / 20) + aBlock.width) &
draai;j++)
            if(aField.blocks[i,j])
                draai = false;
            if(draai)
            {
                aBlock.Turn();
                if(aBlock.height == 4)
                    Invalidate(new
Rectangle(aBlock.location.X - 20, aBlock.location.Y, aBlock.height * 20,
aBlock.height * 20));
                else
                    Invalidate(new
Rectangle(aBlock.location.X, aBlock.location.Y, aBlock.height * 20,
aBlock.height * 20));
            }
        }
    }
}

public void LijnDetect()
{
    bool lijn = true;
    byte aantal = 0;

    aTimer.Enabled = false;

    for(int i = 0 ;i < HEIGHT ; i++)
    {
        for(int j = 0;(j < WIDTH & lijn);j++)
        {
            if(!aField.blocks[j,i])
                lijn = false;
        }
        if(lijn)
        {
            aantal++;
            level++;
            score += (10 * (huidiglevel) * aantal);
            MoveLijn(i);
        }
        lijn = true;
    }
    if(aantal > 0)
    {
        if((level/10) + 1 > huidigelevel & aTimer.Interval > 50)
        {
            huidigelevel++;
            ChangeLevel();
        }
        else
            Invalidate(new Rectangle(LEFT,UP, WIDTH * 20,
HEIGHT * 20));
    }
}

```

```
        Invalidate(new Rectangle(10,20,80,95));
        aTimer.Enabled = true;
    }

    public void ChangeLevel()
    {
        switch(huidiglevel)
        {
            case 2:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level2.bmp");
                aTimer.Interval = 450;
                break;
            case 3:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level3.bmp");
                aTimer.Interval = 400;
                break;
            case 4:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level4.bmp");
                aTimer.Interval = 350;
                break;
            case 5:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level5.bmp");
                aTimer.Interval = 300;
                break;
            case 6:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level6.bmp");
                aTimer.Interval = 250;
                break;
            case 7:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level7.bmp");
                aTimer.Interval = 200;
                break;
            case 8:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level8.bmp");
                aTimer.Interval = 150;
                break;
            case 9:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level9.bmp");
                aTimer.Interval = 100;
                break;
            case 10:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level10.bmp");
                aTimer.Interval = 50;
                break;
            default:
                this.BackgroundImage =
System.Drawing.Image.FromFile(Application.StartupPath+"\\level11.bmp");
                aTimer.Interval = 500;
        }
    }
}
```

```
                break;
            }
            this.Validate();
        }
        public void MoveLijn(int lijn)
        {
            for(int i = lijn ; i > 0; i--)
                for(int j = 0; j < WIDTH;j++)
                {
                    aField.blocks[j,i] = aField.blocks[j,i-1];
                    aField.kleur[j,i] = aField.kleur[j,i-1];
                }
        }

        public bool GameOver()
        {
            for(int i = 0; i < WIDTH & !gameover; i++)
                if(aField.blocks[i,0])
                {
                    gameover = true;
                }
            return gameover;
        }

        public void gedaan()
        {
            aTimer.Enabled = false;
            if(gameover & aBlock.kleur != Color.White)
            {
                MessageBox.Show("Uw score:\n" + score.ToString());

                // highscore controleren
                CheckHighScore();
            }

            //score en lijnen terug op nul zetten
            score = 0;
            level = 0;

            // alles terug leeg maken
            aField = new Field(new Point( LEFT, UP ));
            aBlock = new Block(new Point(STARTX,STARTY),Color.White);
            bBlock = aBlock;

            // zet het juiste aantal lijnen klaar
            if(beginlijn > 0)
                BouwHoog();

            huidiglevel = beginlevel;
            ChangeLevel();

            gameover = true;

            // opnieuw tekenen
            Invalidate();
        }
    }
```

```

public void nieuwblok()
{
    Random r = new Random(unchecked((int)DateTime.Now.Ticks));
    for(int i = 0; i < aBlock.width; i++)
        for(int j = 0; j < aBlock.height; j++)
            if(aBlock.shape[i, j])
            {
                aField.blocks[((aBlock.location.X - LEFT) /
20) + i, ((aBlock.location.Y - UP) / 20) + j] = true;
                aField.kleur[((aBlock.location.X - LEFT) /
20) + i, ((aBlock.location.Y - UP) / 20) + j] = aBlock.kleur;
            }
        LijnDetect();
        Invalidate(new Rectangle(aBlock.getLocation().X,
aBlock.getLocation().Y, aBlock.width, aBlock.height));
        aBlock = bBlock;
        aBlock.setLocation(new Point(STARTX, STARTY));
        bBlock = new Block( new Point( VBX, VBY ), r.Next(1,8) );
        Invalidate(new Rectangle(aBlock.getLocation().X,
aBlock.getLocation().Y, aBlock.width*20, aBlock.height*20));

        //hertekenen previewbox wanneer preview gevraagd wordt
        if(preview)
            Invalidate(new Rectangle(10,120,80,100));
        else
            score++;
    }

public void nieuwspel()
{
    Random r = new Random(unchecked((int)DateTime.Now.Ticks));

    // stop timer
    this.aTimer.Enabled = false;

    // alles terug leeg maken
    if(gameover == false)
        gedaan();
    aBlock = new Block(new Point( STARTX, STARTY ), r.Next(1,8));
    bBlock = new Block( new Point( VBX, VBY ), r.Next(1,8) );

    // zet pauze uit
    pauze = false;

    // hertekenen
    Invalidate();

    // zet gameover uit
    gameOver = false;

    // start timer
    aTimer.Enabled = true;
}

// functie om een starthoogte van blokken in te bouwen

```

```

private void BouwHoog()
{
    Random y = new Random();
    for(int i = HEIGHT - 1; i >= HEIGHT - beginlijn; i--)
    {
        for(int j = 0; j < WIDTH; j++)
        {
            if(y.Next() % 2 == 0)
                aField.blocks[j,i] = true;
            else
                aField.blocks[j,i] = false;
        }
        score += 100*(HEIGHT - i);
    }
}

// functie om het spel stil te leggen of te hervatten
private void PauzeGame()
{
    if(aBlock.shape.Length > 1)
    {
        aTimer.Enabled = !aTimer.Enabled;
        pauze = !pauze;
        Invalidate(new
Rectangle(aField.location.X,aField.location.Y,WIDTH * 20,DOWN));
    }
}

// menufunctie's

// Nieuw Menu item handler
private void Nieuw_Clicked(object sender, System.EventArgs e)
{
    nieuwspel();
}

// Pauze Menu item handler
private void Pauze_Clicked(object sender, System.EventArgs e)
{
    PauzeGame();
}

// Sluiten Menu item handler
private void Sluiten(object sender, System.EventArgs e)
{
    this.Close();
}

protected override void
OnClosing(System.ComponentModel.CancelEventArgs e)
{
    this.aTimer.Enabled = false;
    if(!gameover & aBlock.kleur != Color.White)
    {
        MessageBox.Show("Uw score:\n" + score.ToString());
        this.CheckHighScore();
    }
}

```

```

        base.OnClosing (e);
    }

    // Over Menu item handler
    private void Over_Clicked(object sender, System.EventArgs e)
    {
        PauzeGame();
        Over myOver = new Over();
        myOver.Show();
    }

    // Opties Menu item handler
    private void Opties_Clicked(object sender, System.EventArgs e)
    {
        if(!pauze & !gameover)
            PauzeGame();
        using( Opties aForm = new
Opties(muziek,preview,beginlevel,beginlijn,up,down,right,left) )
        {
            aForm.StartPosition = FormStartPosition.CenterScreen;

            if( aForm.ShowDialog() == DialogResult.OK )
            {
                if(!aForm.muziek.Checked & muziek)
                    PlaySound("NULL",0,1);
                if(aForm.muziek.Checked & !muziek)

                    PlaySound(Application.StartupPath+"\\tetris.wav",0, SND_LOOP |
SND_ASYNC);

                muziek = aForm.muziek.Checked;
                if(preview != aForm.preview.Checked)
                {
                    Invalidate(new Rectangle(10,120,80,100));
                    preview = aForm.preview.Checked;
                }
                beginlevel = Int32.Parse(aForm.level.Text);
                if(beginlijn != Int32.Parse(aForm lijn.Text))
                {
                    beginlijn = Int32.Parse(aForm lijn.Text);
                    if(huidiglevel == beginlevel)
                    {
                        gedaan();
                    }
                }
                // keuze van toetsen maken.
                up = aForm.up.Text;
                down = aForm.down.Text;
                right = aForm.right.Text;
                left = aForm.left.Text;

                if(beginlevel != huidiglevel)
                {
                    PauzeGame();
                    if(score > 0)
                        gameover = true;
                    gedaan();
                }
            }
        }
    }

```

```
    }
}

// Highscore Menu item handler
private void Highscore_Clicked(object sender, System.EventArgs e)
{
    this.showHighScore();
}

private void showHighScore()
{
    HighScore HighScoreForm = new HighScore(highScoreTable);
    HighScoreForm.StartPosition = FormStartPosition.CenterScreen;
    HighScoreForm.Show();
}

private void CheckHighScore()
{
    highScoreTable.Load( Application.StartupPath+"\\highscore.txt"
);
    if( highScoreTable.GetIndexOfScore( score, huidiglevel ) > -1
)
    {
        string name = "";
        using( Gescoord aForm = new Gescoord() )
        {
            aForm.StartPosition =
FormStartPosition.CenterScreen;

            if( aForm.ShowDialog() == DialogResult.OK )
            {
                name = aForm.textBox1.Text;
                highScoreTable.Update( name, score,
huidiglevel );

                showHighScore();
            }
        }
    }
}

// De main functie van het programma
[STAThread]
public static void Main()
{
    Application.Run( new tetris() );
}
}
```