C# spil (del 1) - Kom i gang med et simpelt spil

Denne artikel starter helt fra grunden (med GDI). Den viser hvordan man undgår flimmer, og hvordan man opretter en meget simpel sprite klasse, samt en klasse til at håndtere sprite kollision. Artiklen kræver grundlæggende kendskab til C# og .NET

Skrevet den **12. Feb 2009** af **sovsekoder** I kategorien **Programmering / C#** | ★★★☆☆

Indledning
Denne artikkel tager udgangspunkt i udviklingsmiljøet Visual studio 2003. VS2003 koster knapper. Jeg har derfor indsat denne indledning, der helst skulle hjælpe dem der bruger et gratis udviklings miljø (C# Developer).

Opret et nyt projektet i File -> New combine. Vælg "windows application", og skriv "CollisionSample" i *name*. Tryk ok, og projektet er oprettet. Når jeg snakker om designview, mener jeg viewet, hvor man ser formens layout. Dette view finder du i C# developer ved siden af source-tabben. Når man dobbelt klikker på et event for en kontrol, genererer VS2003 en default metode for event handleren. Dette sker også i C# Developer, dog har C# Developer ikke "_" i metode navnet.

Hvis du har VS2003, skal du starte med at lave et "windows application"-project i C#. I artiklen skal al kode inde i de "blå" bokse indtaste på den ene eller anden måde. Nogle steder er det kode, der kan pastes direkte ind, andre steder er det properties, der skal sættes på en given kontrol på formen.

Første skridt består i at lave en timer. Timeren skal sørge for at spillet kører i et fast tempo. Der er flere måder hvorpå man kan lave en sådan timer. I denne artikel bliver det en forholdsvis simpel timer, nemlig en af dem man kan trække ind på formen - og så er den i vinkel!

Step 1 - tilføj game timer.
Opret et nyt VS2003 projekt - C# "windows application"
Klik på designviewet for Form1
Find en timer i VS2003 toolboxen, og træk den ind på formen. Klik på timeren og indstil følgende properties:

```
Enabled = true
Interval = 10
```

Klik på event iconet (lynet) og dobbel klik på Tick for at få VS2003 til at generere en default timer event handler (timer1_Tick).
Inde i timer1_Tick indtastes nu følgende linie:

```
Invalidate(true);
```

Vi har nu en timer, og en event handler (timer1_Tick) der bliver kaldt hver gang der er gået 10 millisekunder. Styringen af spillet foregår i denne metode, timer1_Tick, metoden kaldes også for gameloopet. Linien der er indsat i timer1_Tick (Invalidate(true)), gør at vinduet bliver gentegnet, hver gang timer eventen bliver fyret. På denne måde er vinduet hele tiden opdateret. Næste skridt består i at læse

og vise mussemarkørens position.

Step 2 - Vis musens position i en label på formen.
Klik på designviewet af formen (Form1)
Træk en label over på formen
Klik på Formen, og klik på event ikonet i properties
Find Paint eventen og dobbelklik på denne, default metoden Form1_Paint genereres af VS2003
Vi er nu inde i Form1 kilde koden for paint metoden .
Følgende linier indtastes i Form1_Paint:

```
// Hent musens position, og indskriv koordineterne
// - i labelen.
Graphics g = e.Graphics;
Point location = PointToClient(MousePosition);
label1.Text = string.Format("{0:0000},{0:0000}", location.X, location.Y);
```

Kør programmet og se at man nu kan se musens koordinater (i den label vi trak ind på formen). Paint eventen bliver fyret når Invalidate kaldes. Da vi har puttet invalidate ind i timer_tick funktionen, bliver paint eventen fyret hver gang der er gået 10 millisekunder. Dette betyder altså at vinduet bliver tegnet 100 gange i sekundet - også kaldet 100fps (100 frames pr. second). Næste skridt er at tilføje en simpel sprite klasse, kaldet BlockSprite. Denne klasse tegner et farvet rektangel som er vores "sprite".

Step 3 - Tegn simple sprites
Vælg Add Class i Project menuen. Skriv BlockSprite I klassens navn og tryk OK. Erstat al autogenereret kode inden for namespacet med følgende kode:

```
public class BlockSprite
{
  // Variabler der giver spritens størrelse og position
  int x=0;
  int y=0;
  int width=0;
  int height = 0;
  // Spritens farveangives med variablen color.
  Color color = Color.Black;

  //BlockSprite konstruktør - opretter en ny sprite
  public BlockSprite(int x, int y, int width, int height)
  {
    this.x = x;
    this.y = y;
    this.width = width;
    this.height = height;
  }

  // Draw tegner spriten i et Grpahics-objekt (formen)
  public void Draw(Graphics g)
  {
    g.FillRectangle(new SolidBrush(color), x,y, width, height);
  }

  // Properties til at gamme og hente spritens:
```

```
  // - position
  // - størrelse
  // - farve
  public int X
  {
    get { return this.x; }
    set { this.x = value; }
  }

  public int Y
  {
    get { return this.y; }
    set { this.y = value; }
  }

  public int Height
  {
    get { return this.height; }
    set { this.height = value; }
  }

  public int Width
  {
    get { return this.width; }
    set { this.width = value; }
  }

  public System.Drawing.Color Color
  {
    get { return this.color; }
    set { this.color = value; }
  }
}
```

øverst i filen tilføjes :

```
using System.Drawing;
```

Denne linie gør os i stand til at bruge tegnefunktioner i .NET
Programmet bør nu kunne kompileres uden fejl.

Den netop indtastede klasse, BlockSprite, er istand til at tegne sprites. I denne artikel er vores sprites meget simple, det er nemlig bare farvede kasser (det kræver ikke megen indsats at putte billeder ind istedet!). BlockSprite klassen har en række properties. 4 properties til at placere spriten på skærmen, samt at angive dens størrelse: x, y, width, height. 1 property til at sætte farven på vores sprite.

Spriten bliver tegnet i funktionen Draw(Graphics g). Denne funktion skal kaldes når man ønsker at tegne spriten i vinduet.

Vi skal nu oprette nogle sprites, og sørge for at de tegnes i vinduet. Følgende linie indsættes som variabel i Form1 klassen:

```
BlockSprite[] sprites;
```

Et array af sprites. Vi indsætter nu 4 sprites i arrayet. Tilføj følgende kode i Form1's kunstruktør:

```
// Tegn med double-buffering, undgå flimmer
SetStyle(ControlStyles.UserPaint, true);
SetStyle(ControlStyles.AllPaintingInWmPaint, true);
SetStyle(ControlStyles.DoubleBuffer, true);
// Lav 4 nye sprites
sprites = new BlockSprite[4];
sprites[0] = new BlockSprite(10,10,40,60);
sprites[1] = new BlockSprite(300,100,60,20);
sprites[2] = new BlockSprite(60,200,50,80);
sprites[3] = new BlockSprite(0,0,50,50);
```

BlockSprites konstruktør gør det muligt at sætte spritens properties med det samme. Bemærk de første linier (3 SetStyle kald). Disse kald sætte bestemte style-bits på formen. Den vigtigste style er DoubleBuffer. DoubleBuffer gør at formen tegnes uden at flimre. Dette er en vigtig ting, hvis man gerne vil ha' at folk bare gider, at kigge på det super-spil man nu engang har lavet. DoubleBuffer style kræver at de to andre styles sættes for at det virker (se evt. msdn dokumentationen for SetStyle).

Tilføj følgende kode I Form1_Paint:

```
// I sprite array'et, sættes sprite 3 til en position
// - hvor spriten får centrum, der hvor musen peger
sprites[3].X = location.X-sprites[3].Width/2;
sprites[3].Y = location.Y-sprites[3].Height/2;
// Tegn alle sprites i array'et
for(int i=0; i<sprites.Length; i++)
{
  sprites[i].Draw(g);
}
```

Koden laver 2 interessante ting. 1) sprite[3]'s position styres nu af musen. Dette gøres ved at sætte x og y property'en på sprite[3]. 2) Herefter tegnes alle sprites ved at løbe array'et af sprites igennem, og kalde Draw metoden for hver sprite.

Vi har nu en form med 4 sorte kasser. Den ene kasse følger musens bevægelse, de andre kasser står bare stille. Vi har tilføjet BlockSprite klassen, som tegner disse kasser og holder styr på hvor og hvor store de er.

Sidste skridt bliver at være i stand til at detektere sprite kollisioner.

Step 4 - Check for kollisioner
Tilføj en klasse med navnet, SpriteCollision (brug Add Class i project menuen).

Kopier følgende kode ind i namespacet for projektet (og overskriv derfor den autogenererede kode inde i namespacet):

```
// Opret en delegate der kan bruges i forbindelse med
// - Listener-eventen
public delegate void CollisionDelegate(BlockSprite s1, BlockSprite s2);

public class SpriteCollision
{
  ArrayList listeners = new ArrayList();

  // AddListener - tilføjer en listener mellem sprite s1 og s2.
  // - metoden CollisionDelegate kaldes når s1 og s2 kolliderer.
  public void AddListener(BlockSprite s1, BlockSprite s2, CollisionDelegate d
)
  {
    Listener l = new Listener(s1, s2, d);
    listeners.Add(l);
  }

  // Her tjekkes alle listeners for kollisioner
  public void Check()
  {
    foreach(Listener listener in  listeners)
    {
      if(listener.Collide())
      {
        listener.FireCollisionEvent();
      }
    }
  }
}

class Listener
{
  BlockSprite sprite1;
  BlockSprite sprite2;
  event CollisionDelegate CollisionEvent;

  public Listener(BlockSprite s1, BlockSprite s2, CollisionDelegate d)
  {
    sprite1 = s1;
    sprite2 = s2;
    CollisionEvent += d;
  }

  // Collide - tjekker om to rektangler overlapper.
  // - dvs. retanglet for sprite1 og sprite2
  public bool Collide()
  {
    if(sprite1.X < (sprite2.X+sprite2.Width))
    {
      if((sprite1.X+sprite1.Width) > sprite2.X)
      {
        if(sprite1.Y < (sprite2.Y+sprite2.Height))
        {
          if((sprite1.Y+sprite1.Height) > sprite2.Y)
```

```
          {
            return true;
          }
        }
      }
    }
    return false;
  }

  public void FireCollisionEvent()
  {
    CollisionEvent(sprite1, sprite2);
  }
}
```

Tilføj et using statement i toppen af filen så ArrayList klassen kan bruges:

```
using System.Collections;
```

Programmet kan nu kompiles uden fejl. Denne klasse kræver noget forklaring.

Øverst har vi en delegate denne delegate er basis for et event der defineres senere. Overordnet fungerer det således at når to sprites er stødt sammen, så fyres dette event. På denne måde kan man abonere på et event, og derfor få at vide hvornår to sprites er støt sammen.

Den første variable der erklares i SpriteCollision klassen er en arraylist. Denne arraylist kommer til at bestå af "Listeners", altså lyttere. Disse lyttere, lytter på om to sprites er støt sammen. Hvis man harløst til at høre om sprite A og sprite B støder sammen, så tilføjer man en lytter på sprite A og B. Klassen har en metode, AddListener, der gør det muligt at tilføje en sådan lytter (Listener).

Herudover har vi metoden Check. Denne metode kaldes når man ønsker at tjekke om der er sket nogle kollisioner. I vores tilfælde kommer vi til at kalde denne metode i timer1_tick (mere om det senere), hvilket så betyder at vi hele tiden beder denne klasse om at tjekke for sammenstødte sprites.

Tilsidst har vi en Listener klasse. Denne klasse holder på 2 sprites (BlockSprites) og en event. Klassen har en metode, der tjekker om de 2 sprites er stødt sammen, hvis de er - ja, så fyres event'en. Eventen fyres af den sidste metode kaldet FireCollisionEvent.

Nu til koden der bruger denne kollisions-dektektor-klasse. Det første skridt er at oprette en variabel i form1 klassen. Tilføj følgende kode øverst i form1 klassen:

```
SpriteCollision collisions = new SpriteCollision();
```

Variablen collisions kan nu bruges til at oprette listeners, således at vi kan få en event når 2 (eller flere) sprites støder sammen. Tilføj følgende kode i Timer1_tick:

```
// Sørg for at alle sprites farves sort inden vi
// - tjekker for kollision
for(int i=0; i<sprites.Length; i++)
{
```

```
  sprites[i].Color = Color.Black;
}
// Tjek for kollision - hvis to sprites kolliderer tegnes de røde
collisions.Check();
```

Her farves alle sprites sorte, og der tjekkes efterfølgende for kollisioner. Bagtanken er nu hvis to sprites er kollideret males de røde. Hvilket gøres med følgende metode (tilføj koden til form1):

```
public void CollisionHandler(BlockSprite sprite1, BlockSprite sprite2)
{
  // Mal de to implicerede sprites røde
  sprite1.Color = Color.Red;
  sprite2.Color = Color.Red;
}
```

Vi mangler nu at tilføje denne metode til vores Listener. Tilføj følgende kode sidst i form1 konstruktøren:

```
collisions.AddListener(sprites[0], sprites[3], new  CollisionDelegate(CollisionHandler));
collisions.AddListener(sprites[1], sprites[3], new  CollisionDelegate(CollisionHandler));
collisions.AddListener(sprites[2], sprites[3], new  CollisionDelegate(CollisionHandler));
```

Vi har således tilføjet en listener til:
sprite3 og sprite0
sprite3 og sprite1
sprite3 og sprite2

Dette betyder at hvis sprite3 (som jo styres af musen) støder ind i en af de andre sprites, så farves de røde - dette sker da event handleren, CollisionHandler, forbindes ved AddListener kaldet (og som tidligere nævnt sørger denne metode for, at farve de implicerede sprites røde).

Artiklen har givet et kort (og hurtigt) indblik i, hvad der skal til at sætte en simpel sprite klasse op, samt at lave en simpel detektion for sprite kollision. Herudover er linierne trukket op til gameloopet (som er Timer1_tick metoden), hvor spil logikken ligger.

Tro det eller ej, du er ikke langt fra at lave et simpelt spil nu. De mest elementære ting er kridtet op. Er du (læseren) interesseret i at vide mere, vil jeg gerne skrive lidt mere om emnet (på samme "lave" niveau) - det vil så være:

DEL 2: Hvordan tilstandsmaskinen styrer spillet

DEL 3: Implementering af Space Invaders (med BlockSprites)!!

DEL 4: grafik og effekter - hvordan grafik og effekter tilføjes

DEL 5: musik og lyd i et spil - hvordan lyden kommer på

Kritik (positiv og negativ) modtages genrne :)

Kommentar af mjense173 d. 30. Oct 2004 | 1

Super! Godt arbejde, jeg kan ikke se hvad visualdeveloper kan sætte fingre på.
Mere af den slags!

Kommentar af charly d. 20. Jan 2005 | 2

Kommentar af finger d. 01. Nov 2004 | 3

Jeg havde nu håbet på at den indeholdt noget Direct3D kode eller mere "spil" orienteret kode. Men dog en god introduktion til tankerne bag spil programmering.

Kommentar af webcreator d. 30. Oct 2004 | 4

Faktisk en ganske god artikel. Jeg må indrømme, at det jeg bedst kunne bruge fra artiklen, var delen hvor man finder musens koordinater - det vil være mig behjælpelig i et andet projekt.

Jeg håber meget, at du fortsætter med artiklerne, for pt. kan vi ikke bruge det til noget. Det kunne være sjovt, hvis du fik lavet en afsluttende artikel, der gennemgår, hvordan man faktisk får lavet et lille spil (og helst et rimeligt brugbart et).

Godt arbejde :)

Kommentar af henrikgn d. 26. Oct 2004 | 5

Super artikel, til os c# nørder... Fortsæt endeligt det gode arbejde - glæder mig til spillet bliver brugbart :)

Kommentar af dustie d. 03. Dec 2007 | 6

Kommentar af kevinsk (nedlagt brugerprofil) d. 08. Oct 2005 | 7

Rigtig lækkert! sad nemlig og kiggede efter lidt hjælp til Collision

Kommentar af plugin- d. 01. Jun 2005 | 8

En af de bedste artikler jeg har læst indtil videre... for nybegyndere som mig selv følger man virkelig at man kommer igang... og jeg er da også så småt ved at videreudvikle dette eksempel til noget større :D

Kommentar af mmbn d. 11. Nov 2005 | 9

Super beskrivelse

Kommentar af sorenbs d. 13. Jul 2005 | 10

Super spændende læsning.
Jeg har aldrig beskæftiget mig mad andet end lidt php, og alligevel kom jeg hurtigt igang med dette "lille" projekt.
Det bør nok lige nævnes at det gratis program "Visual C# Express Edition Beta 2" kan hentes hos Microsoft

her:
http://lab.msdn.microsoft.com/express/vcsharp/default.aspx .
Så skulle alle kunne komme i gang :)

Kommentar af mogenhelge d. 25. Oct 2004 | 11

Godt pædagoisk beskrevet - det er lige til at forstå. Ser frem til næste afsnit! :)

Kommentar af vigilante d. 25. Oct 2004 | 12

Det er en god start på at få noget viden om netop det at lave spil omend det naturligvis er meget simpelt. Hvis du kan komme så langt som til at vise hvordan man laver et simpelt space invaders som du skriver, ja så er det sgu meget godt gået.

Ihvertfald klart de 5 point værd =)

Jeg er dog ikke sikker på jeg forstår hele koden, så skulle dine kommentarer næsten have været på linieniveau, derfor har jeg givet den karakteren god og ikke meget god.

Men jeg er jo også en c# noob!

Kommentar af visualdeveloper d. 17. Oct 2005 | 13

Sådan !

Kommentar af sorensen_123 d. 30. Sep 2006 | 14

Super fed artikel!
Må hellere lige læse videre og se hvad der kommer ud af det! :D

Og til dig, phil-profil, kan du ikke lige tage at slappe lidt af herinde? Artiklen er super god, så enten er det nok din hjerne der er noget galt med, at du overhovedet ikke har læst artiklen, eller måske bare er så dum at du ikke kan finde ud af det?
Da utroligt hvordan folk opfører sig inde på forums i den her tid, hvad sker der inde i folks hoveder? Ja, jeg ved det ærligt talt ikke!

Kommentar af phil-profil d. 02. Jul 2006 | 15

fucking dålig det værste der findes fuck det.
og jeg har givet 5 point!!!!

Kommentar af tobiasahlmo d. 24. Jan 2008 | 16

Nogenlunde

Kommentar af sovsekoder d. 23. Feb 2011 | 17

kig evt. på mit codeplex projekt ang. XNA og platform spil:

http://mrdev.codeplex.com/