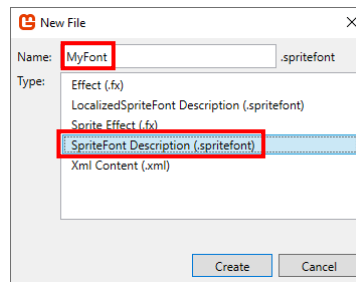


MonoGame – Text och Ljud

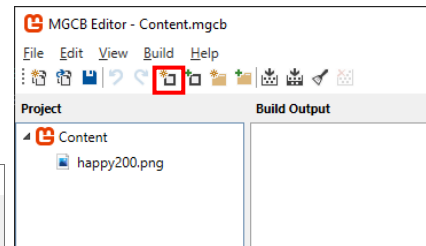
Den här guiden förutsätter inte mycket.

1. För att lägga in text i programmet behöver man skapa en s.k. spritefont via content editorn, detta är i princip ett typsnitt.

Tryck på **New Item** och välj **SpriteFont Description**, namnges i guiden till **MyFont**.

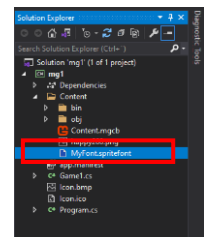


Figur 2



Figur 1

2. Man kan göra inställningar typsnittet via Visual Studio, klicka på **MyFont** i **Solution Explorer**. Vill man ha svenska behöver man ändra till 246

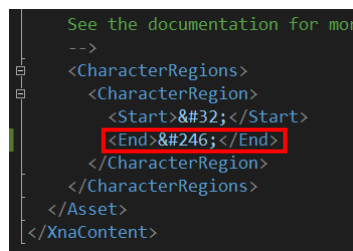


Figur 3

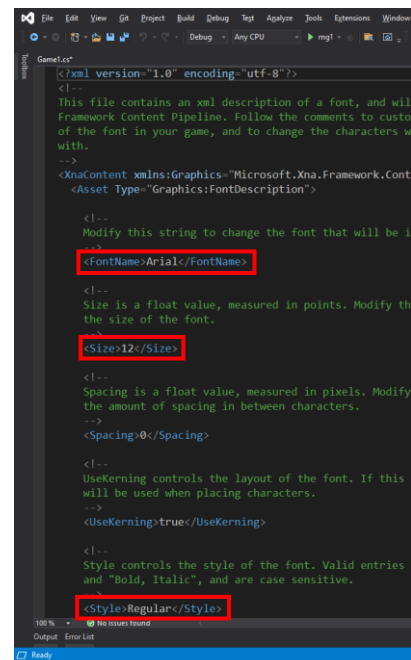
Själv typsnittet är **Arial** från början, kan ändras genom att skriva namnet på ett annat typsnitt som är installerat på datorn. T.ex. Calibri.

Går även att ändra **storlek** och **stil**. Se figur 5.

Behöver man använda svenska tecken så som å, ä och ö behöver man ändra CharacterRegion, **längst ned**, så att den slutar på **246**, se figur 4.



Figur 4

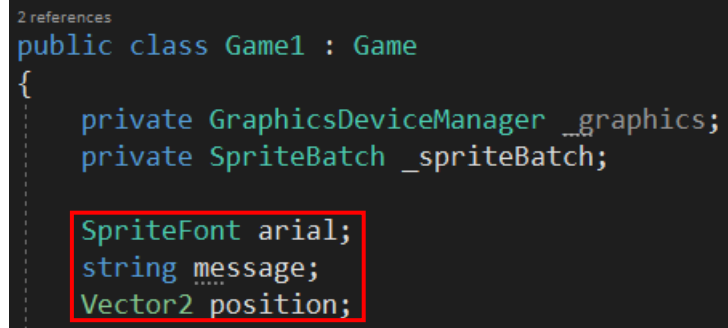


Figur 5

3. Deklarerar variabler för typsnittet, texten som ska skrivas ut och dess position.

Namnges till **arial**, **message** och **position**.

Notera: Detta görs i vanlig ordning utanför metoderna men inne i klassen Game1.

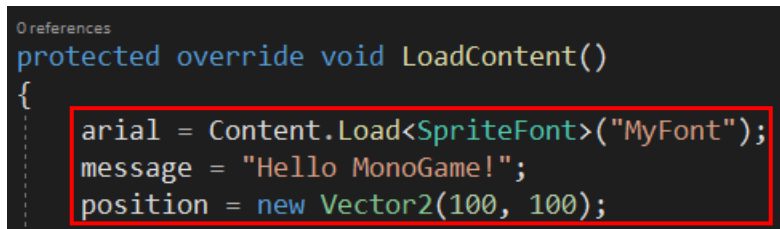
A screenshot of a code editor showing the Game1 class. The class is public and inherits from Game. It has two private fields: GraphicsDeviceManager _graphics and SpriteBatch _spriteBatch. Below these, three variables are declared: SpriteFont arial, string message, and Vector2 position. These three lines are enclosed in a red rectangular box. The code is color-coded: public, class, and Game1 are blue; Game is green; private, GraphicsDeviceManager, and _graphics are blue; SpriteBatch and _spriteBatch are blue; SpriteFont, arial, string, message, Vector2, and position are blue; and the semicolons are green.

```
2 references
public class Game1 : Game
{
    private GraphicsDeviceManager _graphics;
    private SpriteBatch _spriteBatch;

    SpriteFont arial;
    string message;
    Vector2 position;
}
```

Figur 6

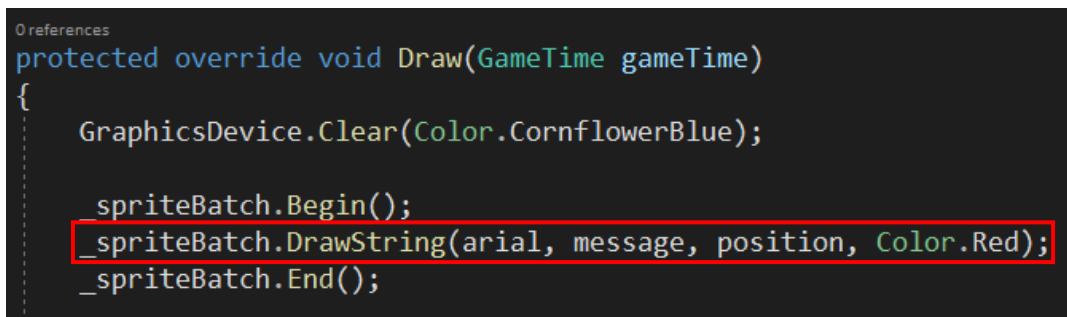
4. Tilldelar variablerna innehåll i metoden **LoadContent()**.

A screenshot of a code editor showing the LoadContent method. The method is protected and overrides the base method. It contains three lines of code: arial = Content.Load<SpriteFont>("MyFont");, message = "Hello MonoGame!";, and position = new Vector2(100, 100);. These three lines are enclosed in a red rectangular box. The code is color-coded: protected, override, void, and LoadContent() are blue; Content, Load, and SpriteFont are blue; "MyFont" is in quotes; Hello MonoGame! is in quotes; new, Vector2, 100, and 100 are blue; and the semicolons are green.

```
0 references
protected override void LoadContent()
{
    arial = Content.Load<SpriteFont>("MyFont");
    message = "Hello MonoGame!";
    position = new Vector2(100, 100);
}
```

Figur 7

5. Meddelandet ritas ut med hjälp av metoden **DrawString()**. Använder färgen röd bara för att.

A screenshot of a code editor showing the Draw method. The method is protected and overrides the base method. It contains four lines of code: GraphicsDevice.Clear(Color.CornflowerBlue);, _spriteBatch.Begin();, _spriteBatch.DrawString(arial, message, position, Color.Red);, and _spriteBatch.End();. The last line is enclosed in a red rectangular box. The code is color-coded: protected, override, void, and Draw are blue; gameTime is green; GraphicsDevice, Clear, Color, and CornflowerBlue are blue; _spriteBatch, Begin, DrawString, arial, message, position, Color, and Red are blue; and the semicolons are green.

```
0 references
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);

    _spriteBatch.Begin();
    _spriteBatch.DrawString(arial, message, position, Color.Red);
    _spriteBatch.End();
}
```

Figur 8

6. Ljud i MonoGame kan vara av typen **Song** eller **SoundEffect**.

För att kunna använda dem lägger vi till två nya namespaces. **Media** och **Audio**.

Deklarerar även en variabel av varje typ. **bgMusic** och **mySound**.

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using Microsoft.Xna.Framework.Audio;

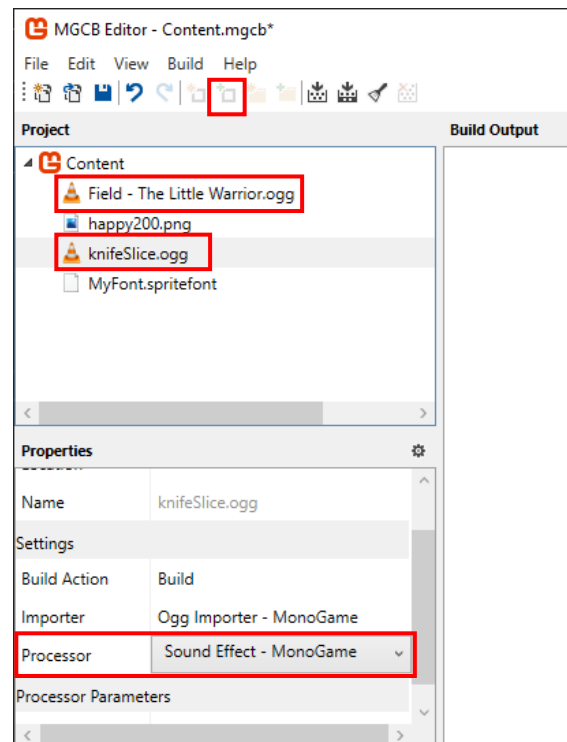
namespace mg1
{
    2 references
    public class Game1 : Game
    {
        private GraphicsDeviceManager _graphics;
        private SpriteBatch _spriteBatch;

        Song bgMusic;
        SoundEffect mySound;
    }
}
```

Figur 9

7. Ljudfiler laddas in via content editorn, använd **Existing Item**.

Notera att de är av typen **Song** från början, för att kunna använda en ljudfil som en **Sound Effect** måste inställningen ändras, se figur 10.



Figur 10

8. Ljudfiler behöver sparas till variabler precis som bilder.

```
0 references
protected override void LoadContent()
{
    bgMusic = Content.Load<Song>("Field - The Little Warrior");
    mySound = Content.Load<SoundEffect>("knifeSlice");
}
```

Figur 11

9. Ljud av typen **Song** hanteras med hjälp av klassen **MediaPlayer** och behöver initialt bara startas upp, kan göras direkt i metoden **LoadContent()** efter tilldelningen av variablerna.

```
0 references
protected override void LoadContent()
{
    bgMusic = Content.Load<Song>("Field - The Little Warrior");
    mySound = Content.Load<SoundEffect>("knifeSlice");

    MediaPlayer.Play(bgMusic);
    MediaPlayer.IsRepeating = true;
    MediaPlayer.Volume = 0.5f;
}
```

Figur 12

Tolkning:

Starta upp ljudfilen som finns i variabeln *bgMusic*, ställ in spelaren på att repetera om ljudfilen tar slut och ställ in volymen på 50 %.

10. Ljud av typen **SoundEffect** kan köras i metoden **Update()** vid lämpligt tillfälle med hjälp av metoden **Play()**.

```
0 references
protected override void Update(GameTime gameTime)
{
    keyboardOld = keyboard;
    keyboard = Keyboard.GetState();
    mouse = Mouse.GetState();

    if (keyboard.IsKeyDown(Keys.Space) && keyboardOld.IsKeyUp(Keys.Space))
    {
        mySound.Play();
    }
}
```

Figur 13

Tolkning: Spela upp ljudfilen som finns i variabeln *mySound*, en gång, om användaren trycker på mellanslag.