C# SFML - Screens in SFML

In unserem Framework haben wir bereits mehrere Screens. Wir haben zwar nur in der MainScreen gearbeitet, aber es gibt auch den GameOverScreen, GameWinScreen und den ScoreScreen. Auf diese konnten wir mit vorgegebenen Methoden wechseln.

Zuerst erstellen wir uns einen neuen Screen. Um zu wissen, wie eine Screen Klasse aufgebaut sein muss, sehen wir uns MainScreen an:

using System;

using System.Collections.Generic;

using System.Text;

Using Anweisungen

using SFML.Window;

using SFML.Audio;

using SFML.Graphics;

using SFML.System;

using SFML\_GAME.entities;

Angabe der Parent Klasse  
(haben wir noch nicht behandelt)

using SFML\_GAME.framework;

namespace SFML\_GAME.screens

{

class MainScreen : Screen

{

// Deklariere hier Objekte oder Klassenvariablen!

// Setup, wird immer einmal zu Beginn eines Screens aufgerufen

// Hier Startwerte setzen!

public override void setup()

{

setup() und loop()

}

// Loop, wird jeden Frame (60 mal die Sekunde) aufgerufen

public override void loop()

{

}

}

}

Eine Screen Klasse muss immer das ": Screen" hinter der Klassendeklaration besitzen. Das macht sie erst zu einem Screen. Dies hat mit Vererbung zutun, was wir später näher behandeln.

Dazu brauchen wir immer die Methoden "setup()" und "loop()".  
Außerdem sollten die Using Anweisungen übernommen werden.

Hier eine frisch erstellte Klasse namens NewScreen:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace SFML\_GAME\_2.screens

{

class NewScreen

{

}

}

Hier können wir jetzt unsere oben genannten Elemente einfügen.  
using System;

setup() und loop()

using System.Collections.Generic;

Using Anweisungen

using System.Text;

using SFML.Window;

using SFML.Audio;

using SFML.Graphics;

using SFML.System;

Angabe der Parent Klasse  
(haben wir noch nicht behandelt)

using SFML\_GAME.entities;

using SFML\_GAME.framework;

namespace SFML\_GAME\_2.screens

{

class NewScreen : Screen

{

public override void setup()

{

}

public override void loop()

{

}

}

}

NewScreen ist jetzt ein neue leerer Screen. Wir können ihn jetzt wie MainScreen mit Element füllen.

using System;

using System.Collections.Generic;

using System.Text;

using SFML.Window;

using SFML.Audio;

using SFML.Graphics;

using SFML.System;

using SFML\_GAME.entities;

using SFML\_GAME.framework;

namespace SFML\_GAME\_2.screens

{

class NewScreen : Screen

{

Entity example;

public override void setup()

{

example = new Entity(this);

}

public override void loop()

{

example.loop();

example.draw();

}

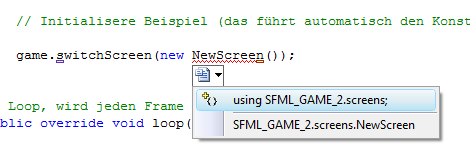
}

}

Um jetzt zwischen Screens wechseln zu können, brauchen wir die Methode "game.switchScreens( .. )" in die Klammern kommt der Screen, zu dem wir wechseln wollen. Für unseren NewScreen also:

game.switchScreen(new NewScreen());

Dabei kommt es wahrscheinlich zu einem Error, dass die Klasse MainScreen unseren NewScreen nicht kennt. Dafür einfach mit Maus auf das Kontext Menü warten und die entsprechende Using Anweisung anklicken.



Jetzt können wir zu anderen Screens wechseln.

## SubScreens

Um aus einen Screen, einen SubScreen zu machen, ist eigentlich nicht viel nötig. Man muss nur folgendes beachten:

Aus dem ": Screen", sollte man ein ": SubScreen" machen. Außerdem sollte "loop()" zu "subloop()" unbenannt werden.

Letztlich muss man noch folgenden Code in die Klasse einfügen:

public NewScreen(Screen parentScreen)

: base(parentScreen)

{

}

Wobei "NewScreen" durch den entsprechenden Klassennamen ersetzt werden muss.

Unsere NewScreen SubScreen Klasse sieht jetzt so aus:

using System;

using System.Collections.Generic;

using System.Text;

using SFML.Window;

using SFML.Audio;

using SFML.Graphics;

using SFML.System;

using SFML\_GAME.entities;

using SFML\_GAME.framework;

namespace SFML\_GAME.screens

{

class NewScreen : SubScreen

{

public NewScreen(Screen parentScreen)

: base(parentScreen)

{

}

public override void setup()

{

}

public override void subloop()

{

}

}

}

Wenn wir jetzt zu unserer NewScreen Klasse wechseln wollen mit:

game.switchScreen(new NewScreen());

Gibt es einen Error, weil ein SubScreen brauch einen HauptScreen, auf dem er angezeigt wird. Deswegen geben wir ein "this" mit, damit es auf den MainScreen projeziert wird.

game.switchScreen(new NewScreen(this));

Jetzt werden alle Inhalte von NewScreen auf MainScreen projeziert. Beide Screens sind gleichzeitig aktiv.

Wenn man jetzt wieder zurück auf den MainScreen wechseln will, gibt es 2 Möglichkeiten.

Mit

game.switchScreen(new MainScreen());

wird ein neuer MainScreen erzeugt, dabei geht der alte verloren.

Mit

this.goBack();

gelangt man wieder zum vorherigem MainScreen, mit allen Objekten so wie sie waren.