

Kapitel 1

Einführung

In diesem Kapitel erlernen Sie anhand eines ersten Projekts den Umgang mit der Entwicklungsumgebung und den Steuerelementen. Anschließend werden Sie in der Lage sein, Ihr erstes eigenes Windows-Programm zu erstellen.

C# ist eine objektorientierte Programmiersprache, die von Microsoft im Zusammenhang mit dem .NET-Framework eingeführt wurde. Mithilfe der Entwicklungsumgebung Visual Studio 2012 können Sie unter anderem in der Sprache C# programmieren. Visual Studio 2012 ist der Nachfolger von Visual Studio 2010. Innerhalb von Visual Studio stehen Ihnen noch weitere Sprachen zur Programmentwicklung zur Verfügung.

C#

1.1 Aufbau dieses Buches

Dieses Buch vermittelt Ihnen zunächst einen einfachen Einstieg in die Programmierung mit Visual C# 2012. Die Bearbeitung der Beispiele und das selbstständige Lösen der vorliegenden Übungsaufgaben helfen dabei. Dadurch werden Sie schnell erste Erfolgserlebnisse haben, die Sie zum Weitermachen motivieren. In späteren Kapiteln werden Ihnen auch die komplexen Themen vermittelt.

Beispiele

Von Anfang an wird mit anschaulichen Windows-Anwendungen gearbeitet. Die Grundlagen der Programmiersprache und die Standardelemente einer Windows-Anwendung, wie Sie sie schon von anderen Windows-Programmen her kennen, werden gemeinsam vermittelt. Die Anschaulichkeit einer Windows-Anwendung hilft dabei, den eher theoretischen Hintergrund der Programmiersprache leichter zu verstehen.

Grundlagen

1.2 Visual Studio 2012

Express Edition

Es werden mehrere Express-Versionen von Visual Studio 2012 unter Windows 7 mit Service Pack 1 und unter Windows 8 eingesetzt. Diese freien Versionen von Visual Studio 2012 liegen dem Buch bei, Sie können sie aber auch bei Microsoft herunterladen.

Die Express-Versionen von Visual Studio 2012 bieten jeweils eine komfortable Entwicklungsumgebung. Sie umfassen einen Editor zur Erstellung des Programmcodes, einen Compiler zur Erstellung der ausführbaren Programme, einen Debugger zur Fehlersuche und vieles mehr.

Auf dem Datenträger zum Buch finden Sie die folgenden Express-Versionen von Visual Studio 2012, mit unterschiedlichen Einsatzbereichen:

- ▶ *Visual Studio Express 2012 für Desktop* zur Programmierung der Windows Forms-Anwendungen sowie der Konsolen-Anwendungen, die Sie in Kapitel 1 bis 8 sowie 10 und 11 sehen werden. Außerdem dient diese Version zur Umsetzung der WPF-Anwendungen in Kapitel 12. Mit der Desktop-Version werden wir beginnen.
- ▶ *Visual Studio Express 2012 für das Web* zur Programmierung der Internet-Anwendungen mit C# in Kapitel 9.
- ▶ *Visual Studio Express 2012 für Windows 8* zur Programmierung der Windows-Store-Apps, die nur in Windows 8 laufen und die Sie in Kapitel 13 sehen werden.

Zur Installation der Express-Versionen verweise ich auf den Anhang.

Noch eine Anmerkung in eigener Sache: Für die Hilfe bei der Erstellung dieses Buches bedanke ich mich beim Team von Galileo Press, besonders bei Anne Scheibe.

Thomas Theis

1.3 Mein erstes Windows-Programm

Anhand eines ersten Projekts werden Sie die Schritte durchlaufen, die zur Erstellung eines einfachen Programms mithilfe von Visual C# 2012 notwendig sind. Das Programm soll nach dem Aufruf zunächst aussehen wie in Abbildung 1.1.

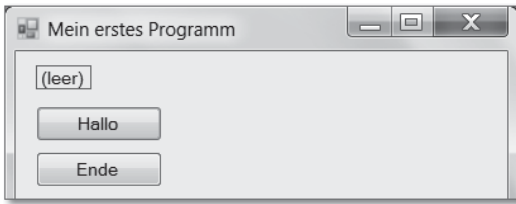


Abbildung 1.1 Erstes Programm nach dem Aufruf

Nach Betätigung des Buttons HALLO soll sich der Text in der obersten Zeile verändern, siehe Abbildung 1.2.

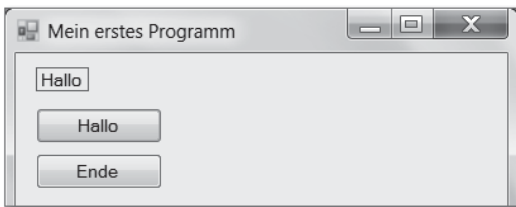


Abbildung 1.2 Nach Betätigung des Buttons »Hallo«

1.4 Visual C# 2012-Entwicklungsumgebung

Während der Projekterstellung lernen Sie Schritt für Schritt die Visual Studio 2012-Entwicklungsumgebung kennen.

1.4.1 Ein neues Projekt

Nach dem Aufruf des Programms Visual Studio Express 2012 für Windows Desktop müssen Sie zur Erstellung eines neuen C#-Projekts den Menüpunkt DATEI • NEUES PROJEKT • INSTALLIERT • VORLAGEN • VISUAL C# • WINDOWS FORMS-ANWENDUNG auswählen. Als Projektname bietet die Entwicklungsumgebung den Namen *WindowsFormsApplication1* an, dieser sollte geändert werden, zum Beispiel in *MeinErstes*.

Es erscheinen einige Elemente der Entwicklungsumgebung. Folgende Elemente sind besonders wichtig:

- Das Benutzerformular (engl.: Form) enthält die Oberfläche für den Benutzer des Programms (siehe Abbildung 1.3). Form

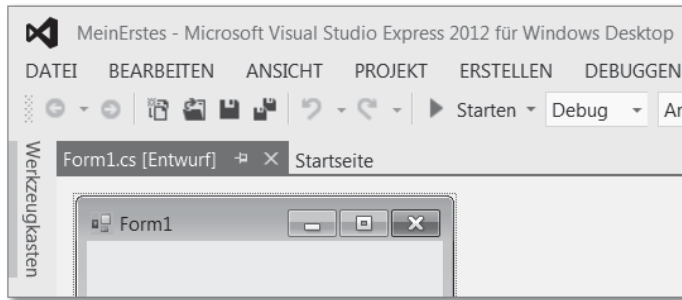


Abbildung 1.3 Benutzerformular

- Toolbox** ▶ Die WERKZEUGSAMMLUNG (engl.: TOOLBOX) enthält die Steuerelemente für den Benutzer, mit denen er den Ablauf des Programms steuern kann. Sie werden vom Programm-Entwickler in das Formular eingefügt (siehe Abbildung 1.4).

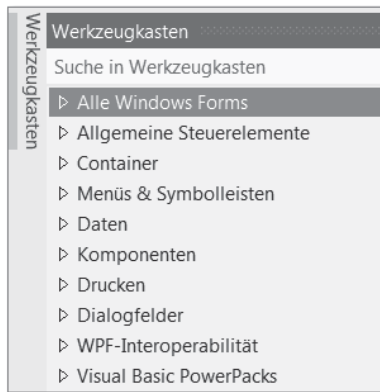


Abbildung 1.4 Toolbox mit verschiedenen Kategorien von Steuerelementen

- Eigenschaftensfenster** ▶ Das EIGENSCHAFTENFENSTER (engl.: Properties Window) dient zum Anzeigen und Ändern der Eigenschaften von Steuerelementen innerhalb des Formulars durch den Programm-Entwickler (siehe Abbildung 1.5). Ich empfehle Ihnen, sich die Eigenschaften in alphabetischer Reihenfolge anzeigen zu lassen. Dazu einfach das zweite Symbol von links, unter FORM1, betätigen.
- Projektmappen-Explorer** ▶ Der PROJEKTMAPPEN-EXPLORER (engl.: *Solution Explorer*) zeigt das geöffnete Projekt und die darin vorhandenen Elemente (siehe Abbildung 1.6).

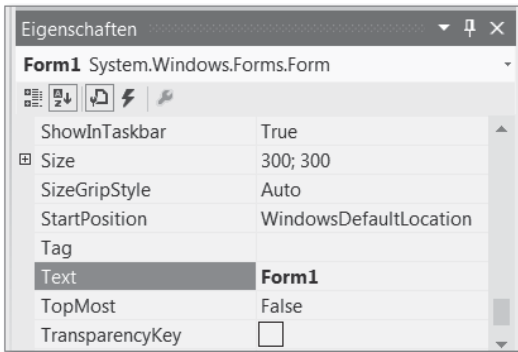


Abbildung 1.5 Eigenschaftenfenster

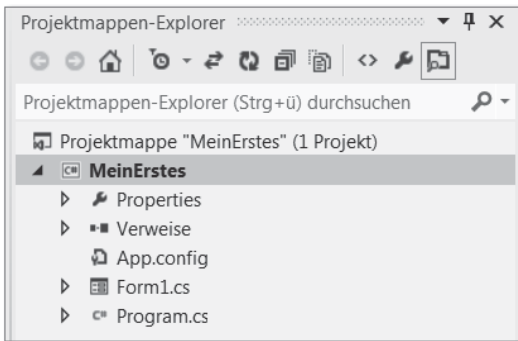


Abbildung 1.6 Projektmappen-Explorer

Sollten die TOOLBOX, das EIGENSCHAFTENFENSTER oder der PROJEKTMAPPEN-EXPLORER einmal nicht sichtbar sein, so können Sie das betreffende Element über das Menü ANSICHT einblenden. Sollte das Formular einmal nicht sichtbar sein, so können Sie es über einen Doppelklick auf den Namen (FORM1.CS) im PROJEKTMAPPEN-EXPLORER einblenden.

Anfangs schreiben Sie nur einfache Programme mit wenigen Elementen, daher benötigen Sie den PROJEKTMAPPEN-EXPLORER noch nicht. Es empfiehlt sich, das EIGENSCHAFTENFENSTER nach oben zu vergrößern.

1.4.2 Einfügen von Steuerelementen

Zunächst sollen drei Steuerelemente in das Formular eingefügt werden: ein Bezeichnungsfeld (Label) und zwei Befehlsschaltflächen (Buttons). Ein Bezeichnungsfeld dient im Allgemeinen dazu, feste oder veränderliche

Label, Button

Texte auf der Benutzeroberfläche anzuzeigen. In diesem Programm soll das Label einen Text anzeigen. Ein Button dient zum Starten bestimmter Programmteile oder, allgemeiner ausgedrückt, zum Auslösen von Ereignissen. In diesem Programm sollen die Buttons dazu dienen, den Text anzuzeigen bzw. das Programm zu beenden.

Allgemeine Steuerelemente

Um ein Steuerelement einzufügen, ziehen Sie es mithilfe der Maus von der TOOLBOX an die gewünschte Stelle im Formular. Alle Steuerelemente finden sich in der TOOLBOX unter ALLE WINDOWS FORMS. Übersichtlicher ist der Zugriff über ALLGEMEINE STEUERELEMENTE (engl.: *Common Controls*), siehe Abbildung 1.7.

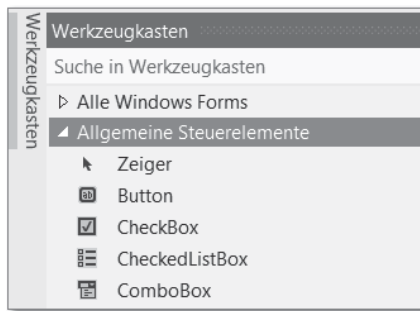


Abbildung 1.7 Toolbox, Allgemeine Steuerelemente

Steuerelement auswählen

Ein Doppelklick auf ein Steuerelement in der TOOLBOX fügt es ebenfalls in die Form ein. Anschließend können Ort und Größe noch verändert werden. Dazu wählen Sie das betreffende Steuerelement vorher durch Anklicken aus, siehe Abbildung 1.8. Ein überflüssiges Steuerelement können Sie durch Auswählen und Drücken der Taste **Entf** entfernen.

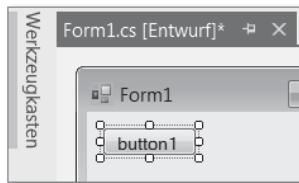


Abbildung 1.8 Ausgewählter Button

Die Größe und andere Eigenschaften des Formulars selbst können Sie auch verändern. Dazu wählen Sie es vorher durch Anklicken auf einer freien Stelle aus.


1.4.3 Arbeiten mit dem Eigenschaftfenster

Die eingefügten Steuerelemente haben zunächst einheitliche Namen und Aufschriften, diese sollten Sie allerdings zur einfacheren Programmentwicklung ändern. Es haben sich bestimmte Namenskonventionen eingebürgert, die die Lesbarkeit erleichtern. Diese Namen beinhalten den Typ (mit drei Buchstaben abgekürzt) und die Aufgabe des Steuerelements (mit großem Anfangsbuchstaben).

Ein Button (eigentlich: *Command Button*), der die Anzeige der Zeit auslösen soll, wird beispielsweise mit `cmdZeit` bezeichnet. Weitere Vorsilben sind `txt` (Textfeld/TextBox), `lbl` (Bezeichnungsfeld/Label), `opt` (Optionsschaltfläche/RadioButton), `frm` (Formular/Form) und `chk` (Kontrollkästchen/CheckBox).

`cmd, txt, lbl, ...`

Zur Änderung des Namens eines Steuerelements muss es zunächst ausgewählt werden. Das können Sie entweder durch Anklicken des Steuerelements auf dem Formular oder durch Auswahl aus der Liste am oberen Ende des EIGENSCHAFTENFENSTERS tun.

Im EIGENSCHAFTENFENSTER werden alle Eigenschaften des ausgewählten Steuerelements angezeigt. Die Liste ist zweispaltig: In der linken Spalte steht der Name der Eigenschaft, in der rechten Spalte ihr aktueller Wert. Die Eigenschaft (NAME) steht am Anfang der Liste der Eigenschaften. Die betreffende Zeile wählen Sie durch Anklicken aus und geben hier den neuen Namen ein. Nach Bestätigung mit der Taste  ist die Eigenschaft geändert, siehe Abbildung 1.9.

Eigenschaftensfenster

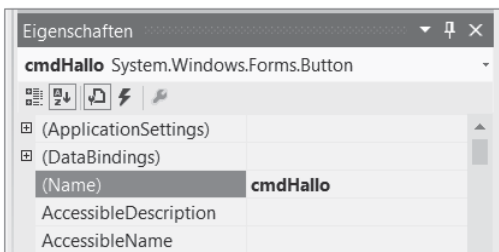


Abbildung 1.9 Button, nach der Namensänderung

Die Aufschrift von Buttons, Labels und Formularen ist in der Eigenschaft `Text` angegeben. Sobald diese Eigenschaft verändert wird, erscheint die veränderte Aufschrift in dem betreffenden Steuerelement. Auch der Name

`Text`

und die Aufschrift des Formulars sollten geändert werden. Im Folgenden sind die gewünschten Eigenschaften für die Steuerelemente dieses Programms in Tabellenform angegeben, siehe Tabelle 1.1.

Typ	Eigenschaft	Einstellung
Formular	Text	Mein erstes Programm
Button	Name	cmdHallo
	Text	Hallo
Button	Name	cmdEnde
	Text	Ende
Label	Name	lblAnzeige
	Text	(leer)
	BorderStyle	FixedSingle

Tabelle 1.1 Steuerelemente mit Eigenschaften

Startzustand Zu diesem Zeitpunkt legen Sie den Startzustand fest, also die Eigenschaften, die die Steuerelemente zu Beginn des Programms bzw. eventuell während des gesamten Programms haben sollen. Viele Eigenschaften können Sie auch während der Laufzeit des Programms durch den Programmcode verändern.

Bei einem Label ergibt die Einstellung der Eigenschaft `BorderStyle` auf `FixedSingle` einen Rahmen. Zur Änderung auf `FixedSingle` klappen Sie die Liste bei der Eigenschaft auf und wählen den betreffenden Eintrag aus, siehe Abbildung 1.10. Zur Änderung einiger Eigenschaften müssen Sie gegebenenfalls ein Dialogfeld aufrufen.

Im Label soll zunächst der Text (*leer*) erscheinen. Hierzu wählen Sie den vorhandenen Text durch Anklicken aus und ändern ihn.

Liste der Steuerelemente Sie finden alle in diesem Formular vorhandenen Steuerelemente in der Liste, die sich am oberen Ende des EIGENSCHAFTENFENSTERS öffnen lässt. Dabei zeigt sich ein Vorteil der einheitlichen Namensvergabe: Die Steuerelemente des gleichen Typs stehen direkt untereinander.

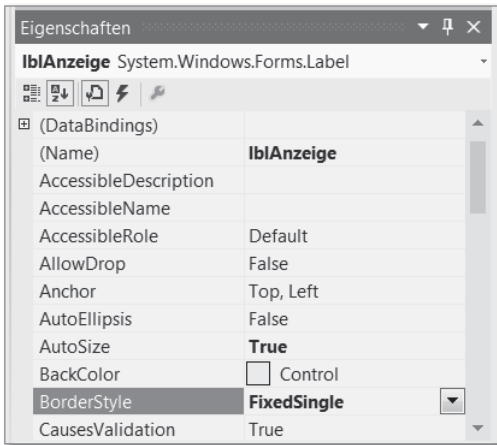


Abbildung 1.10 Label, nach der Änderung von Name und BorderStyle

1.4.4 Speichern eines Projekts

Die Daten eines Visual C#-Projekts werden in verschiedenen Dateien gespeichert. Zum Speichern des gesamten Projekts verwenden Sie den Menüpunkt **DATEI • ALLE SPEICHERN**. Diesen Vorgang sollten Sie in regelmäßigen Abständen durchführen, damit keine Änderungen verloren gehen können.

Alles speichern

Die in diesem Skript angegebenen Namen dienen als Empfehlung, um die eindeutige Orientierung und das spätere Auffinden von alten Programmen zu erleichtern.

1.4.5 Das Codefenster

Der Ablauf eines Windows-Programms wird im Wesentlichen durch das Auslösen von Ereignissen durch den Benutzer gesteuert. Er löst z. B. die Anzeige des Texts *Hallo* aus, indem er auf den Button HALLO klickt. Der Entwickler muss dafür sorgen, dass aufgrund dieses Ereignisses der gewünschte Text angezeigt wird. Zu diesem Zweck schreibt er Programmcode und ordnet diesen Code dem Ereignis zu. Der Code wird in einer *Ereignismethode* abgelegt.

Ereignis

Zum Schreiben einer Ereignismethode führen Sie am besten einen Doppelklick auf das betreffende Steuerelement aus. Es erscheint das Codefenster. Zwischen der Formularansicht und der Codeansicht können Sie

Ereignismethode

anschließend über die Menüpunkte ANSICHT • CODE bzw. ANSICHT • DESIGNER hin- und herschalten. Dies ist auch über die Registerkarten oberhalb des Formulars bzw. des Codefensters möglich, siehe Abbildung 1.11.

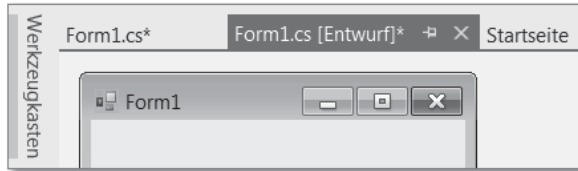


Abbildung 1.11 Registerkarten

Nach erfolgtem Doppelklick auf den Button HALLO erscheinen im Codefenster folgende Einträge:

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

namespace MeinErstes
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            private void cmdHallo_Click(
                object sender, EventArgs e)
            {

        }
    }
}
```

Listing 1.1 Projekt »MeinErstes«, Button »Hallo«, ohne Code

Zur Erläuterung:

- Sie sollten sich nicht von der Vielzahl der automatisch erzeugten Zeilen und den noch unbekannten Inhalten verwirren lassen.

- Zunächst das Wichtigste: Innerhalb der geschweiften Klammern { } der Ereignismethode `cmdHallo_Click()` wird später Ihr eigener Programmcode hinzugefügt.

Zu den anderen Bestandteilen (die für das eigene Programmieren erst später wichtig sind):

- C# ist eine objektorientierte Sprache. Ein wichtiges Element objektorientierter Sprachen sind Klassen. Klassen eröffnen weitere Programmiermöglichkeiten. Namensräume beinhalten zusammengehörige Klassen. **Namensraum**
- In obigem Listing können Sie erkennen, dass einige Programmzeilen, die mit `using` beginnen, bereits entfernt wurden. Das Schlüsselwort `using` dient zum Einbinden von Namensräumen in das aktuelle Projekt. Da wir die Klassen in diesen Namensräumen nicht benötigen, wurden die betreffenden Zeilen entfernt. **using**
- Dieses erste Projekt verfügt über einen eigenen Namensraum (engl. *namespace*), daher `namespace MeinErstes`. **namespace**
- Alle Elemente des aktuellen Formulars `Form1` stehen innerhalb der öffentlich zugänglichen Klasse `Form1`, daher `public class Form1`. Ein Teil der Elemente steht in dieser Datei, ein anderer Teil, der ebenfalls automatisch erzeugt wurde, steht in einer anderen, hier nicht sichtbaren Datei; daher der Zusatz `partial` (dt. teilweise). **public partial class**
- Die Methode `InitializeComponent()` beinhaltet Programmzeilen, die für das Aussehen und Verhalten der Steuerelemente des Programms sorgen.
- Der Zusatz `private` bedeutet, dass die Ereignismethode `cmdHallo_Click()` nur in dieser Klasse bekannt ist. Mit `void` wird gekennzeichnet, dass diese Methode nur etwas ausführt, aber kein Ergebnis zurückliefert. **private void**
- Auf weitere Einzelheiten dieser automatisch erzeugten Bestandteile wird zu einem späteren Zeitpunkt eingegangen, da es hier noch nicht notwendig ist und eher verwirren würde.

Wie bereits erwähnt: Die Ereignismethode für den Klick auf den Button HALLO heißt `cmdHallo_Click()`. Der Kopf der Methode ist sehr lang, daher wurde er für den Druck in diesem Buch auf mehrere Zeilen verteilt, wodurch auch die Lesbarkeit von Programmen erhöht wird:

```
private void cmdHallo_Click(
    object sender, EventArgs e)
```

Der anfänglich ausgeführte Doppelklick führt immer zu dem Ereignis, das am häufigsten mit dem betreffenden Steuerelement verbunden wird.

Click Dies ist beim Button natürlich das Ereignis `Click`. Zu einem Steuerelement gibt es aber auch noch andere mögliche Ereignisse.

Bei den nachfolgenden Programmen werden nicht mehr alle Teile des Programmcodes im Buch abgebildet, sondern nur noch

- die Teile, die vom Entwickler per Code-Eingabe erzeugt werden,
- und die Teile des automatisch erzeugten Codes, die wichtig für das Verständnis sind.

Den vollständigen Programmcode können Sie jederzeit betrachten, wenn Sie die Beispiel-Projekte laden bzw. ausprobieren.

1.4.6 Schreiben von Programmcode

In der Methode `cmdHallo_Click()` soll eine Befehlszeile eingefügt werden, so dass sie anschließend wie folgt aussieht:

```
private void cmdHallo_Click(
    object sender, EventArgs e)
{
    lblAnzeige.Text = "Hallo";
}
```

Listing 1.2 Projekt »MeinErstes«, Button »Hallo«, mit Code

Der Text muss in Anführungszeichen gesetzt werden, da C# sonst annimmt, dass es sich um eine Variable mit dem Namen `Hallo` handelt.

Anweisung Der Inhalt einer Methode setzt sich aus einzelnen Anweisungen zusammen, die nacheinander ausgeführt werden. Die vorliegende Methode enthält nur eine Anweisung; in ihr wird mithilfe des Gleichheitszeichens eine Zuweisung durchgeführt.

Zuweisung Bei einer Zuweisung wird der Ausdruck rechts vom Gleichheitszeichen ausgewertet und der Variablen, der Objekt-Eigenschaft oder der Steuerelement-Eigenschaft links vom Gleichheitszeichen zugewiesen. Die Zeichenkette *Hallo* wird der Eigenschaft `Text` des Steuerelements `lblAnzeige` mithilfe der Schreibweise `Steuerelement.Eigenschaft = Wert` zugewiesen. Dies führt zur Anzeige des Werts.

Nach dem Wechsel auf die Formularansicht können Sie das nächste Steuerelement auswählen, für das eine Ereignismethode geschrieben werden soll.

Innerhalb des Codefensters kann Text mit den gängigen Methoden der Textverarbeitung editiert, kopiert, verschoben und gelöscht werden.

[Code editieren](#)

In der Ereignismethode `cmdEnde_Click()` soll der folgende Code stehen:

```
private void cmdEnde_Click(
    object sender, EventArgs e)
{
    Close();
}
```

Listing 1.3 Projekt »MeinErstes«, Button »Ende«

Die Methode `Close()` dient zum Schließen eines Formulars. Da es sich um das einzige Formular dieses Projekts handelt, wird dadurch das Programm beendet und die gesamte Windows-Anwendung geschlossen.

`Close()`

Dies waren Beispiele zur Änderung der Eigenschaften eines Steuerelements zur Laufzeit des Programms durch Programmcode. Sie erinnern sich: Zu Beginn hatten wir die Start-Eigenschaften der Steuerelemente im EIGENSCHAFTENFENSTER eingestellt.

1.4.7 Kommentare

Bei längeren Programmen mit vielen Anweisungen gehört es zum guten Programmierstil, Kommentarzeilen zu schreiben. In diesen Zeilen werden einzelne Anweisungen oder auch längere Blöcke von Anweisungen erläutert, damit Sie selbst oder auch ein anderer Programmierer sie später leichter verstehen. Alle Zeichen innerhalb eines Kommentars werden nicht übersetzt oder ausgeführt.

Ein Kommentar beginnt mit der Zeichenkombination `/*`, endet mit der Zeichenkombination `*/` und kann sich über mehrere Zeilen erstrecken.

`/* Kommentar */`

Eine andere Möglichkeit ergibt sich durch die Zeichenkombination `//`. Ein solcher Kommentar erstreckt sich nur bis zum Ende der Zeile.

`// Kommentar`

Der folgende Programmcode wurde um einen Kommentar ergänzt:

```
private void cmdEnde_Click(
    object sender, EventArgs e)
{
    /* Diese Anweisung beendet
       das Programm */
    Close();
}
```

Listing 1.4 Projekt »MeinErstes«, Button »Ende«, mit Kommentar

**Code
auskommentieren**

Ein kleiner Trick: Sollen bestimmte Programmzeilen für einen Test des Programms kurzfristig nicht ausgeführt werden, können Sie sie *auskommentieren*, indem Sie die Zeichenkombination `//` vor die betreffenden Zeilen setzen. Dies geht sehr schnell, indem Sie die betreffende(n) Zeile(n) markieren und anschließend das entsprechende Symbol im linken Bereich der Symbolleiste anklicken, siehe Abbildung 1.12. Rechts daneben befindet sich das Symbol, das die Auskommentierung nach dem Test wieder rückgängig macht.



Abbildung 1.12 Kommentar ein/aus

1.4.8 Starten, Ausführen und Beenden des Programms

Programm starten

Nach dem Einfügen der Steuerelemente und dem Erstellen der Ereignismethoden ist das Programm fertig und kann gestartet werden. Dazu betätigen Sie den Start-Button in der Symbolleiste (dreieckiger Pfeil nach rechts). Alternativ starten Sie das Programm über die Funktionstaste `[F5]` oder den Menüpunkt **DEBUGGEN • DEBUGGING STARTEN**. Das Formular erscheint, das Betätigen der Buttons führt zum programmierten Ergebnis.

Programm beenden

Zur regulären Beendigung eines Programms ist der Button mit der Aufschrift *Ende* vorgesehen. Möchten Sie ein Programm während des Verlaufs abbrechen, können Sie auch den End-Button in der Symbolleiste (Quadrat) betätigen.

Fehler

Tritt während der Ausführung eines Programms ein Fehler auf, so werden Sie hierauf hingewiesen, und das Codefenster zeigt die entsprechende Ereignismethode sowie die fehlerhafte Zeile an. In diesem Fall beenden Sie das Programm, korrigieren Sie den Code und starten Sie das Programm wieder.

Es ist empfehlenswert, das Programm bereits während der Entwicklung mehrmals durch Aufruf zu testen und nicht erst, wenn das Programm vollständig erstellt worden ist. Geeignete Zeitpunkte sind zum Beispiel:

Programm testen

- ▶ nach dem Einfügen der Steuerelemente und dem Zuweisen der Eigenschaften, die Sie zu Programmbeginn benötigen
- ▶ nach dem Erstellen jeder Ereignismethode

1.4.9 Ausführbares Programm

Nach erfolgreichem Test des Programms können Sie die ausführbare Datei (.exe-Datei) auch außerhalb der Entwicklungsumgebung aufrufen. Haben Sie an den Grundeinstellungen nichts verändert und die vorgeschlagenen Namen verwendet, so findet sich die zugehörige .exe-Datei des aktuellen Projekts im Verzeichnis *Eigene Dateien\Visual Studio 2012\Projects\Mein-Erstes\MeinErstes\bin\Debug*. Das Programm kann also im Windows-Explorer direkt über Doppelklick gestartet werden.

.exe-Datei

Die Weitergabe eines eigenen Windows-Programms auf einen anderen PC ist etwas aufwendiger. Der Vorgang wird in Abschnitt A.5 beschrieben.

1.4.10 Projekt schließen, Projekt öffnen

Sie können ein Projekt schließen über den Menüpunkt DATEI • PROJEKT-MAPPE SCHLIESSEN. Falls Sie Veränderungen vorgenommen haben, werden Sie gefragt, ob Sie diese Änderungen speichern möchten.

Projekt schließen

Möchten Sie die Projektdaten sicherheitshalber zwischendurch speichern, so ist dies über den Menüpunkt DATEI • ALLE SPEICHERN möglich. Dies ist bei längeren Entwicklungsphasen sehr zu empfehlen.

Zum Öffnen eines vorhandenen Projekts wählen Sie den Menüpunkt DATEI • PROJEKT ÖFFNEN. Im darauf folgenden Dialogfeld PROJEKT ÖFFNEN wählen Sie zunächst das gewünschte Projektverzeichnis aus und anschließend die gleichnamige Datei mit der Endung *.sln*.

Projekt öffnen

1.4.11 Übung

Erstellen Sie ein Windows-Programm mit einem Formular, das zwei Buttons und ein Label beinhaltet, siehe Abbildung 1.13. Bei Betätigung des ers-

Übung ÜName

ten Buttons erscheint im Label Ihr Name. Bei Betätigung des zweiten Buttons wird das Programm beendet. Namensvorschläge: Projektname *ÜName*, Buttons *cmdMyName* und *cmdEnde*, Label *lblMyName*.

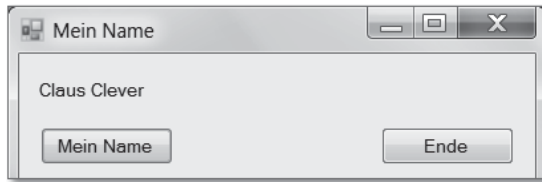


Abbildung 1.13 Übung ÜName

1.4.12 Empfehlungen für Zeilenumbrüche

Zeilenumbruch Zeilenumbrüche erhöhen die Lesbarkeit des Programmcodes. Sie können nicht an jeder Stelle einer Anweisung durchgeführt werden. Nachfolgend werden einige Stellen empfohlen:

- ▶ nach einer öffnenden Klammer, wie bereits gezeigt
- ▶ vor einer schließenden Klammer
- ▶ nach einem Komma
- ▶ nach den meisten Operatoren, also auch nach dem Zuweisungsoperator (=) hinter `lblAnzeige.Text`, siehe Abschnitt 2.2
- ▶ nach einem Punkt hinter einem Objektnamen, also auch nach dem Punkt hinter dem Objektnamen `lblAnzeige`

Auf keinen Fall dürfen Sie einen Zeilenumbruch innerhalb einer Zeichenkette durchführen.

1.5 Arbeiten mit Steuerelementen

1.5.1 Steuerelemente formatieren

Hilfslinien Zur besseren Anordnung der Steuerelemente auf dem Formular können Sie sie mithilfe der Maus nach Augenmaß verschieben. Dabei erscheinen automatisch Hilfslinien, falls das aktuelle Element horizontal oder vertikal parallel zu einem anderen Element steht.

Weitere Möglichkeiten bieten die Menüpunkte im Menü **FORMAT**. In vielen Fällen müssen vorher mehrere Steuerelemente auf einmal markiert werden, siehe Abbildung 1.14.

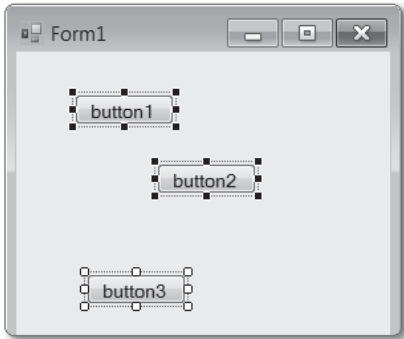




Abbildung 1.14 Mehrere markierte Elemente

Dies geschieht entweder

- ▶ durch Umrahmung der Elemente mit einem Rechteck, nachdem Sie zuvor das Steuerelement **Zeiger** ausgewählt haben, oder
- ▶ durch Mehrfachauswahl, indem Sie ab dem zweiten auszuwählenden Steuerelement die -Taste (wie für Großbuchstaben) oder die -Taste gedrückt halten.

Über das Menü **FORMAT** haben Sie anschließend folgende Möglichkeiten zur Anpassung der Steuerelemente:

Menü »Format«

- ▶ Die ausgewählten Steuerelemente können horizontal oder vertikal zueinander ausgerichtet werden (Menü **FORMAT** • **AUSRICHTEN**).
- ▶ Die horizontalen und/oder vertikalen Dimensionen der ausgewählten Steuerelemente können angeglichen werden (Menü **FORMAT** • **GRÖSSE ANGLEICHEN**).
- ▶ Die horizontalen und vertikalen Abstände zwischen den ausgewählten Steuerelementen können angeglichen, vergrößert, verkleinert oder entfernt werden (Menü **FORMAT** • **HORIZONTALER ABSTAND/VERTIKALER ABSTAND**).
- ▶ Die Steuerelemente können horizontal oder vertikal innerhalb des Formulars zentriert werden (Menü **FORMAT** • **AUF FORMULAR ZENTRIEREN**).

**Einheitliche
Abstände**

- ▶ Sollten sich die Steuerelemente teilweise überlappen, können Sie einzelne Steuerelemente in den Vorder- bzw. Hintergrund schieben (Menü **FORMAT • REIHENFOLGE**).
- ▶ Sie können alle Steuerelemente gleichzeitig gegen versehentliches Verschieben absichern (Menü **FORMAT • STEUERELEMENTE SPERREN**). Diese Sperrung gilt nur während der Entwicklung des Programms.

Abbildung 1.15 zeigt ein Formular mit drei Buttons, die alle links ausgerichtet sind und im gleichen vertikalen Abstand voneinander stehen.

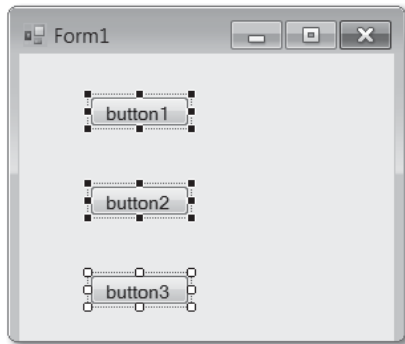


Abbildung 1.15 Nach der Formatierung

Übung

Laden Sie das Projekt *MeinErstes* aus Abschnitt 1.3, markieren Sie darin mehrere Steuerelemente, und testen Sie die einzelnen Möglichkeiten des Format-Menüs.

1.5.2 Steuerelemente kopieren

Steuerelemente kopieren

Zur schnelleren Erzeugung eines Projekts können vorhandene Steuerelemente einschließlich aller ihrer Eigenschaften kopiert werden. Markieren Sie hierzu die gewünschten Steuerelemente und kopieren Sie sie entweder

- ▶ über das Menü **BEARBEITEN • KOPIEREN** und das Menü **BEARBEITEN • EINFÜGEN** oder
- ▶ mit den Tasten **[Strg] + [C]** und **[Strg] + [V]**.

Anschließend sollten Sie die neu erzeugten Steuerelemente direkt umbenennen und an der gewünschten Stelle anordnen.

Übung

Laden Sie das Projekt *MeinErstes* aus Abschnitt 1.3 und kopieren Sie einzelne Steuerelemente. Kontrollieren Sie anschließend die Liste der vorhandenen Steuerelemente im EIGENSCHAFTENFENSTER auf einheitliche Namensgebung.

1.5.3 Eigenschaften zur Laufzeit ändern

Steuerelemente haben die Eigenschaften *Size* (mit den Komponenten *Width* und *Height*) und *Location* (mit den Komponenten *X* und *Y*) zur Angabe von Größe und Position. *X* und *Y* geben die Koordinaten der oberen linken Ecke des Steuerelements an, gemessen von der oberen linken Ecke des umgebenden Elements (meist das Formular). Alle Werte werden in Pixeln gemessen.

Size, Location

Alle diese Eigenschaften können sowohl während der Entwicklungszeit als auch während der Laufzeit eines Projekts verändert werden. Zur Änderung während der Entwicklungszeit können Sie die Eigenschaftswerte wie gewohnt im EIGENSCHAFTENFENSTER eingeben. Als Beispiel für Änderungen während der Laufzeit soll das folgende Programm (Projekt *Steuerelemente*) dienen, siehe Abbildung 1.16.

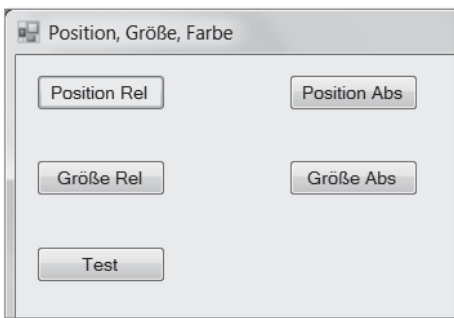


Abbildung 1.16 Position und Größe bestimmen

Es wird nachfolgend generell nur der Teil des Programmcodes angezeigt, der verändert wurde:

```
private void cmdPositionRel_Click(...)
{
    cmdTest.Location = new Point(
        cmdTest.Location.X + 20,
```

```

        cmdTest.Location.Y);
    }

    private void cmdPositionAbs_Click(...)
    {
        cmdTest.Location = new Point(100, 200);
    }

    private void cmdGrößeRel_Click(...)
    {
        cmdTest.Size = new Size(
            cmdTest.Size.Width + 20,
            cmdTest.Size.Height);
    }

    private void cmdGrößeAbs_Click(...)
    {
        cmdTest.Size = new Size(50, 100);
    }

```

Listing 1.5 Projekt »Steuerelemente«

Zur Erläuterung:

Verkürzte Darstellung

- Der Kopf der einzelnen Methoden wurde aus Gründen der Übersichtlichkeit jeweils in verkürzter Form abgebildet. Dies wird bei den meisten nachfolgenden Beispielen ebenfalls so sein, außer wenn es genau auf die Inhalte des Methodenkopfs ankommt.
- Das Formular enthält fünf Buttons. Die oberen vier Buttons dienen zur Veränderung von Position und Größe des fünften Buttons.
- Die Position eines Elements kann relativ zur aktuellen Position oder auf absolute Werte eingestellt werden. Das Gleiche gilt für die Größe eines Elements.
- Bei beiden Angaben handelt es sich um Wertepaare (X/Y bzw. Breite/Höhe).

new Point

- Zur Einstellung der Position dient die Struktur `Point`. Ein Objekt dieser Struktur liefert ein Wertepaar. In diesem Programm wird mit `new` jeweils ein neues Objekt der Struktur `Point` erzeugt, um das Wertepaar bereitzustellen.

- ▶ Bei Betätigung des Buttons POSITION ABS wird die Position des fünften Buttons auf die Werte $X=100$ und $Y=200$ gestellt, gemessen von der linken oberen Ecke des Formulars. X, Y
- ▶ Bei Betätigung des Buttons POSITION REL wird die Position des fünften Buttons auf die Werte $X = \text{cmdTest.Location.X} + 20$ und $Y = \text{cmdTest.Location.Y}$ gestellt. Bei X wird also der alte Wert der Komponente X um 20 erhöht, das Element bewegt sich nach rechts. Bei Y wird der alte Wert der Komponente Y nicht verändert, das Element bewegt sich nicht nach oben oder unten.
- ▶ Zur Einstellung der Größe dient die Struktur Size. Size
- ▶ Bei Betätigung des Buttons GRÖSSE ABS wird die Größe des fünften Buttons auf die Werte $\text{Width} = 50$ und $\text{Height} = 100$ gestellt. Width, Height
- ▶ Bei Betätigung des Buttons GRÖSSE REL wird die Größe des fünften Buttons auf die Werte $\text{Width} = \text{cmdTest.Size.Width} + 20$ und $\text{Height} = \text{cmdTest.Size.Height}$ gestellt. Bei Width wird also der alte Wert der Komponente Width um 20 erhöht, das Element wird breiter. Bei Height wird der alte Wert der Komponente Height nicht verändert, das Element verändert seine Höhe nicht.

Nach einigen Klicks sieht das Formular aus wie in Abbildung 1.17.

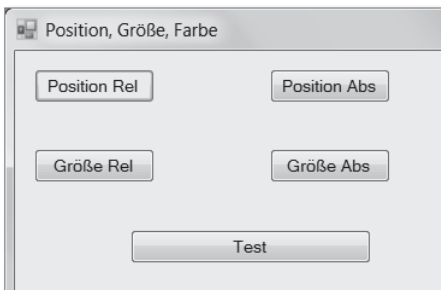


Abbildung 1.17 Veränderung von Eigenschaften zur Laufzeit

1.5.4 Vergabe und Verwendung von Namen

Beachten Sie in allen Programmen, dass jedes Steuerelement seinen eigenen, eindeutigen Namen hat und immer mit diesem Namen angesprochen werden muss. Es passiert erfahrungsgemäß besonders am Anfang häufig, dass ein Programm nicht zum gewünschten Erfolg führt, weil ein nicht vor-

handener Name verwendet wurde. In diesem Zusammenhang weise ich noch einmal auf die Namenskonventionen hin:

- Buttons sollten Namen, wie z. B. `cmdEnde`, `cmdAnzeigen`, `cmdBerechnen` usw., haben.
- Labels sollten Namen, wie z. B. `lblAnzeige`, `lblName`, `lblUhrzeit`, `lblBeginnDatum`, haben.

Diese Namen liefern eine eindeutige Information über Typ und Funktion des Steuerelements. Falls Sie beim Schreiben von Programmcode anschließend diese Namen z. B. vollständig in Kleinbuchstaben eingeben, werden sie nach Verlassen der Zeile darauf aufmerksam gemacht. Sie können schnell erkennen, ob Sie tatsächlich ein vorhandenes Steuerelement verwendet haben.

1.5.5 Verknüpfung von Texten, mehrzeilige Texte

+ und \n Es können mehrere Texte in einer Ausgabe mithilfe des Zeichens + miteinander verknüpft werden. Falls Sie eine mehrzeilige Ausgabe wünschen, können Sie einen Zeilenvorschub mithilfe des Textes "\n" (für *new line*) erzeugen.

Nachfolgend wird das Projekt *Steuerelemente* ergänzt um ein Label, in dem die aktuelle Position und Größe des Buttons angezeigt werden. Dies soll nach Betätigung des Buttons ANZEIGE geschehen:

```
private void cmdAnzeige_Click(...)
{
    lblAnzeige.Text =
        "Position: X: " + cmdTest.Location.X +
        ", Y: " + cmdTest.Location.Y + "\n" +
        "Größe: Breite: " + cmdTest.Size.Width +
        ", Höhe: " + cmdTest.Size.Height;
}
```

Listing 1.6 Projekt »Steuerelemente«, mit Anzeige

Nach einigen Klicks und der Betätigung des Buttons ANZEIGE sieht das Formular aus wie in Abbildung 1.18.



Abbildung 1.18 Anzeige der Eigenschaften

1.5.6 Eigenschaft BackColor, Farben allgemein

Die Hintergrundfarbe eines Steuerelements wird mit der Eigenschaft `BackColor` festgelegt. Dabei können Sie die Farbe zur Entwicklungszeit leicht mithilfe einer Farbpalette oder aus Systemfarben auswählen.

BackColor

Hintergrundfarben und andere Farben können Sie auch zur Laufzeit einstellen. Dabei bedienen Sie sich der Farbwerte, die Sie über die Struktur `Color` auswählen.

Color

Ein Beispiel, ebenfalls im Projekt *Steuerelemente*:

```
private void cmdFarbe_Click(...)
{
    BackColor = Color.Yellow;
    lblAnzeige.BackColor =
        Color.FromArgb(192, 255, 0);
}
```

Listing 1.7 Projekt »Steuerelemente«, mit Farben

Zur Erläuterung:

- Diese Struktur bietet vordefinierte Farbnamen als Eigenschaften, z. B. `Yellow`. Der Wert kann der Eigenschaft `BackColor` des Steuerelements zugewiesen werden, hier ist dies das Formular selbst.
- Außerdem bietet die Struktur die Methode `FromArgb()`. Diese können Sie auf verschiedene Arten aufrufen. Eine dieser Arten erwartet genau drei Parameter, nämlich die Werte für Rot, Grün und Blau, jeweils zwischen 0 und 255.

FromArgb()

Das Formular sieht nach der Änderung der Eigenschaft `Farbe` aus wie in Abbildung 1.19.

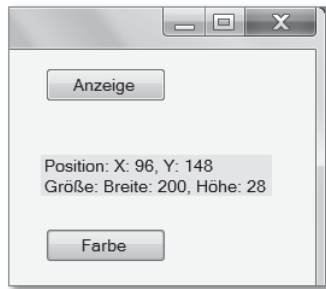


Abbildung 1.19 Nach Änderung der Eigenschaft »Farbe«

Kapitel 11

Beispielprojekte

Als weiterführende Übungsaufgaben werden in diesem Kapitel zwei lauf-fähige Beispielprojekte vorgeführt. Haben Sie den geschilderten Aufbau verstanden, können Sie später eigene Verbesserungen oder Erweiterungen einbringen.

Bei den beiden Beispielprojekten handelt es sich zum einen um das bekannte Tetris-Spiel und zum anderen um einen Vokabeltrainer.

11

11.1 Spielprogramm Tetris

Im Folgenden wird das bekannte Spielprogramm Tetris in einer vereinfachten, nachvollziehbaren Version für Visual C# realisiert und erläutert. Das Programm beinhaltet:

- ▶ ein zweidimensionales Feld
- ▶ einen Timer
- ▶ einen Zufallsgenerator
- ▶ die Erzeugung und Löschung von Steuerelementen zur Laufzeit
- ▶ die Zuordnung von Ereignismethoden zu Steuerelementen, die erst zur Laufzeit erzeugt werden

Abbildung 11.1 zeigt die Benutzeroberfläche des Programms.

11.1.1 Spielablauf

Nach Programmstart fällt ein Steuerelement vom Typ `Panel` in einer von acht möglichen Farben so weit herunter, bis es auf den Rand des Spielfelds oder auf ein anderes Panel trifft. Es kann mithilfe der drei Buttons »Links« (LI), »Rechts« (RE) und »Drop« (DR) bewegt werden. »Drop« bewirkt ein sofortiges Absenken des Panels auf die unterste mögliche Position.

Panel fällt herunter

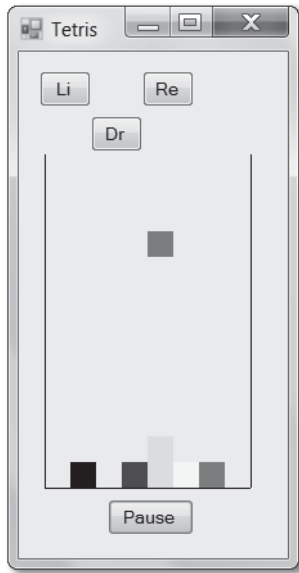


Abbildung 11.1 Tetris

- Nächstes Level** Falls sich drei gleichfarbige Panels untereinander oder nebeneinander befinden, so verschwinden sie. Panels, die sich eventuell darüber befinden, rutschen nach. Anschließend wird die Fallgeschwindigkeit der Panels erhöht, das heißt, die Schwierigkeitsstufe wird gesteigert, man gelangt zum nächsten Level.
- Ende** Sobald ein Panel nur noch in der obersten Zeile platziert werden kann, ist das Spiel zu Ende. Ziel des Spiels ist es, so viele Panels wie möglich zu platzieren. Mit dem Button PAUSE kann das Spiel unterbrochen werden, eine erneute Betätigung des Buttons lässt das Spiel weiterlaufen.

11.1.2 Programmbeschreibung

- Hilfsfeld** Der Kasten, in dem sich die fallenden Panels befinden, ist 8 Spalten breit und 13 Zeilen hoch. Als Hilfskonstruktion steht das zweidimensionale Feld F mit 10 Spalten und 15 Zeilen zur Verfügung, in dem jedes existierende Panel mit seiner laufenden Nummer vermerkt ist.

Ze / Sp	0	1	2	3	4	5	6	7	8	9
1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
2	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
3	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
4	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
5	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
6	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
7	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
8	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
9	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
10	-2	-1	-1	-1	-1	-1	-1	-1	-1	-2
11	-2	-1	-1	-1	11	-1	-1	-1	-1	-2
12	-2	-1	-1	-1	3	8	9	-1	-1	-2
13	-2	-1	0	10	2	4	5	-1	-1	-2
14	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

Tabelle 11.1 Spielfeld

Im Beispiel in Tabelle 11.1 wird der Inhalt des Felds F nach den Panels 0 bis 11, also nach zwölf gefallen Panels angezeigt. Die Panels 1, 6 und 7 hatten die gleiche Farbe, standen über- oder nebeneinander und sind deshalb schon verschwunden. Die Randelemente werden zu Spielbeginn mit dem Wert der Konstanten Rand=-2 besetzt. Alle Elemente des Felds F, die kein Panel enthalten, also leer sind, haben den Wert der Konstanten Leer=-1.

11.1.3 Steuerelemente

Es gibt zu Beginn des Programms folgende Steuerelemente:

- vier Buttons für Links, Rechts, Drop und Pause
- drei Panels als Begrenzungslinien des Spielfelds

- Timer** ► einen Timer, der das aktuelle Panel automatisch weiter fallen lässt (Startwert für den Zeitintervall: 500ms)

Im Verlauf des Programms werden weitere Steuerelemente vom Typ `Panel` hinzugefügt bzw. wieder entfernt.

11.1.4 Initialisierung des Programms

Sie müssen die Namensräume `System.Collections` (für eine `ArrayList`) und `System.Drawing` (für Positionsänderungen von Steuerelementen) einbinden.

Zu Beginn des Programms werden die klassenweit gültigen Variablen und Konstanten vereinbart und die `Form1_Load`-Methode durchlaufen:

```
public partial class Form1 : Form
{
    ....
    /* Index des aktuellen Panels */
    int PX;

    /* Gesamtes Spielfeld inkl. Randfelder */
    int[,] F = new int[15, 10];

    /* Zeile und Spalte des aktuellen Panels */
    int PZ, PS;

    /* Schwierigkeitsstufe */
    int Stufe;

    /* Eine zunächst leere Liste von Spiel-Panels */
    ArrayList PL = new ArrayList();

    /* Ein Feld von Farben für die Panels */
    Color[] FarbenFeld = {Color.Red,
        Color.Yellow, Color.Green, Color.Blue,
        Color.Cyan, Color.Magenta, Color.Black,
        Color.White};

    /* Konstanten für Status eines Feldpunktes */
    const int Leer = -1;
    const int Rand = -2;
```

```

/* Zufallsgenerator erzeugen und initialisieren */
Random r = new Random();

private void Form1_Load(...)
{
    int Z, S;

    /* Feld besetzen */
    for (Z=1; Z<14; Z++)
    {
        F[Z, 0] = Rand;
        for (S=1; S<9; S++)
            F[Z, S] = Leer;
        F[Z, 9] = Rand;
    }

    for (S=0; S<10; S++)
        F[14, S] = Rand;

    /* Initialisierung */
    Stufe = 1;
    NächstesPanel();
}
....

```

11

Listing 11.1 Projekt »Tetris«, Variablen, Konstanten, Start

Zur Erläuterung der klassenweit gültigen Variablen und Konstanten:

- ▶ Die laufende Nummer (der Index) des aktuell fallenden Panels wird in der Variablen `PX` festgehalten.
- ▶ Das gesamte Spielfeld, das in Abschnitt 11.1.2 schematisch dargestellt wurde, wird im zweidimensionalen Feld `F` gespeichert. Hilfsfeld
- ▶ Die Variablen `PZ` und `PS` beinhalten die Zeilen- und Spalten-Position des aktuell fallenden Panels innerhalb des Spielfelds.
- ▶ Die Variable `Stufe` kennzeichnet den Schwierigkeitsgrad des Spiels. Jedes Mal, wenn drei Panels, die untereinander oder nebeneinander lagen, gelöscht wurden, wird die Stufe um 1 erhöht. Dies sorgt für ein kürzeres Timer-Intervall, die Panels werden schneller. Level

- Liste von Panels**
- ▶ PL ist eine ArrayList von Steuerelementen vom Typ Panel. ArrayLists können beliebige Objekte enthalten. Dies können Variablen, Objekte eigener Klassen oder, wie hier, Steuerelemente, also Objekte vorhandener Klassen, sein. Zu Beginn ist die ArrayList leer.
 - ▶ Das Feld `FarbenFeld` enthält insgesamt acht Farben. Die Farben der Panels werden per Zufallsgenerator ermittelt.
 - ▶ Die Konstanten `Leer` und `Rand` werden erzeugt. Die Namen der Konstanten sind im Programm leichter lesbar als die Werte `-1` bzw. `-2`.
 - ▶ Für die Farbauswahl wird der Zufallsgenerator bereitgestellt.

Zur Erläuterung der `Form1_Load`-Methode:

- ▶ Die Elemente des oben beschriebenen Hilfsfelds `F` werden mit `Leer` bzw. `Rand` besetzt.
 - ▶ Die Schwierigkeitsstufe wird auf 1 gesetzt.
- Erstes Panel**
- ▶ Es wird die Methode `NächstesPanel()` aufgerufen. Sie ist in diesem Fall für die Erzeugung des ersten fallenden Panels zuständig.

11.1.5 Erzeugen eines neuen Panels

Die Methode `NächstesPanel()` dient zur Erzeugung eines neuen fallenden Panels. Dies geschieht zu Beginn des Spiels und nachdem ein Panel auf dem unteren Rand des Spielfelds oder auf einem anderen Panel zum Stehen gekommen ist. Der Code lautet:

```
private void NächstesPanel()
{
    int Farbe;
    Panel p = new Panel();

    /* Neues Panel zur ArrayList hinzufügen */
    PL.Add(p);

    /* Neues Panel platzieren */
    p.Location = new Point(100, 80);
    p.Size = new Size(20, 20);

    /* Farbauswahl für neues Panel */
    Farbe = r.Next(0,8);
    p.BackColor = FarbenFeld[Farbe];
}
```

```

/* Neues Panel zum Formular hinzufügen */
Controls.Add(p);

/* Index für späteren Zugriff ermitteln */
PX = PL.Count - 1;

/* Aktuelle Zeile, Spalte */
PZ = 1;
PS = 5;
}

```

Listing 11.2 Projekt »Tetris«, Methode NächstesPanel

Zur Erläuterung:

- ▶ Es wird ein Objekt vom Typ `Panel` neu erzeugt.
- ▶ Damit darauf auch außerhalb der Methode zugegriffen werden kann, wird ein Verweis auf dieses Panel mithilfe der Methode `Add()` der `Array-List` `PL` hinzugefügt.
- ▶ Es werden die Eigenschaften *Ort*, *Größe* und *Farbe* des neuen Panels bestimmt.
- ▶ Das Panel wird mithilfe der Methode `Add()` zu der `Collection Controls` hinzugefügt. Dies ist eine Liste der Steuerelemente des Formulars. Dadurch wird das Panel sichtbar.
- ▶ Seine laufende Nummer (der Index) wird mithilfe der Eigenschaft `Count` ermittelt. Diese Nummer wird für den späteren Zugriff benötigt.
- ▶ Die Variablen `PZ` und `PS`, die die Position des aktuell fallenden Panels im Spielfeld `F` angeben, werden gesetzt.

Neues Listen-
element

Neues Steuer-
element

11.1.6 Der Zeitgeber

In regelmäßigen Zeitabständen wird das `Timer-Ereignis` erzeugt und damit die Ereignismethode `timT_Tick()` aufgerufen. Diese sorgt dafür, dass sich das aktuelle Panel nach unten bewegt, falls dies noch möglich ist:

```

private void timT_Tick(...)
{
    /* Falls es nicht mehr weiter geht */
    if (F[PZ + 1, PS] != Leer)

```

```

{
    /* Oberste Zeile erreicht */
    if (PZ == 1)
    {
        timT.Enabled = false;
        MessageBox.Show("Das war's");
        return;
    }

    F[PZ, PS] = PX;      // Belegen
    AllePrüfen();
    NächstesPanel();
}
else
{
    /* Falls es noch weiter geht */
    Panel p = (Panel) PL[PX];
    p.Top = p.Top + 20;
    PZ = PZ + 1;
}
}

```

Listing 11.3 Projekt »Tetris«, Zeitgeber

Zur Erläuterung:

- Zunächst wird geprüft, ob sich unterhalb des aktuellen Panels noch ein freies Feld befindet.
 - Ist dies nicht der Fall, so hat das Panel seine Endposition erreicht.
- Endposition**
- Befindet sich diese Endposition in der obersten Zeile, so ist das Spiel zu Ende. Der Timer wird deaktiviert, anderenfalls würden weitere Panels erzeugt. Es erscheint eine Meldung, und die Methode wird unmittelbar beendet. Will der Spieler erneut beginnen, so muss er das Programm beenden und neu starten.
 - Befindet sich die Endposition nicht in der obersten Zeile, so wird die Panelnummer im Feld F mit der aktuellen Zeile und Spalte vermerkt. Dies dient zur Kennzeichnung eines belegten Feldelements.
- Prüfen**
- Die Methode AllePrüfen() wird aufgerufen (siehe unten), um festzustellen, ob es drei gleichfarbige Panels über- oder nebeneinander gibt. Anschließend wird das nächste Panel erzeugt.

- Befindet sich unterhalb des Panels noch ein freies Feld, so kann das Panel weiter fallen. Seine Koordinaten und die aktuelle Zeilennummer werden verändert.

Weiter fallen

11.1.7 Panels löschen

Die Methode `AllePrüfen()` ist eine rekursive Methode, mit deren Hilfe festgestellt wird, ob es drei gleichfarbige Panels nebeneinander oder übereinander gibt. Ist dies der Fall, werden diese Panels entfernt und die darüberliegenden Panels rutschen nach.

Rekursive Methode

Möglicherweise befinden sich nun wieder drei gleichfarbige Panels nebeneinander oder übereinander, es muss also wiederum geprüft werden. Dies geschieht so lange, bis keine drei gleichfarbigen Panels nebeneinander oder übereinander gefunden werden.

Die Methode `AllePrüfen()` bedient sich intern der beiden Methoden `NebenPrüfen()` und `ÜberPrüfen()`:

```
private void AllePrüfen()
{
    int Z, S;
    bool Neben, Über;
    Neben = false;
    Über = false;

    /* Drei gleiche Panels nebeneinander ? */
    for(Z=13; Z>0; Z--)
    {
        for(S=1; S<7; S++)
        {
            Neben = NebenPrüfen(Z, S);
            if (Neben) break;
        }
        if (Neben) break;
    }

    /* Drei gleiche Panels übereinander ? */
    for(Z=13; Z>2; Z--)
    {
        for(S=1; S<9; S++)
        {
```

```

        Über = ÜberPrüfen(Z, S);
        if (Über) break;
    }
    if (Über) break;
}

if (Neben || Über)
{
    /* Schneller */
    Stufe = Stufe + 1;
    timT.Interval = 5000 / (Stufe + 9);

    /* Eventuell kann jetzt noch eine Reihe
       entfernt werden */
    AllePrüfen();
}
}
/* Falls 3 Felder nebeneinander besetzt */
private bool NebenPrüfen(int Z, int S)
{
    int ZX, SX;
    bool ergebnis = false;

    if (F[Z, S] != Leer &&
        F[Z, S + 1] != Leer &&
        F[Z, S + 2] != Leer)
    {
        Panel p = (Panel) PL[F[Z, S]];
        Panel p1 = (Panel) PL[F[Z, S + 1]];
        Panel p2 = (Panel) PL[F[Z, S + 2]];

        /* Falls drei Farben gleich */
        if (p.BackColor == p1.BackColor &&
            p.BackColor == p2.BackColor)
        {
            for(SX=S; SX<S+3; SX++)
            {
                /* PL aus dem Formular löschen */
                Control c = (Control) PL[F[Z, SX]];
                Controls.Remove(c);
            }
        }
    }
}

```

```

        /* Feld leeren */
        F[Z, SX] = Leer;

        /* Panels oberhalb des entladenen
           Panels absenken */
        ZX = Z - 1;
        while (F[ZX, SX] != Leer)
        {
            Panel px =
                (Panel) PL[F[ZX, SX]];
            px.Top = px.Top + 20;

            /* Feld neu besetzen */
            F[ZX + 1, SX] = F[ZX, SX];
            F[ZX, SX] = Leer;
            ZX = ZX - 1;
        }

    }
    ergebnis = true;
}
}
return ergebnis;
}

/* Falls drei Felder übereinander besetzt */
private bool ÜberPrüfen(int Z, int S)
{
    int ZX;
    bool ergebnis = false;

    if (F[Z, S] != Leer && F[Z - 1, S] != Leer &&
        F[Z - 2, S] != Leer)
    {
        Panel p = (Panel) PL[F[Z, S]];
        Panel p1 = (Panel) PL[F[Z - 1, S]];
        Panel p2 = (Panel) PL[F[Z - 2, S]];

        /* Falls drei Farben gleich */
        if (p.BackColor == p1.BackColor &&
            p.BackColor == p2.BackColor)

```

```

    {
        /* 3 Panels entladen */
        for (ZX=Z; ZX>Z-3; ZX--)
        {
            /* PL aus dem Formular löschen */
            Control c = (Control) PL[F[ZX, S]];
            Controls.Remove(c);
            /* Feld leeren */
            F[ZX, S] = Leer;
        }
        ergebnis = true;
    }
}
return ergebnis;
}

```

Listing 11.4 Projekt »Tetris«, Panels löschen

Zur Erläuterung:

- Die Variablen `Neben` und `Über` kennzeichnen die Tatsache, dass drei gleichfarbige Panels neben- oder übereinander gefunden wurden. Sie werden zunächst auf `false` gesetzt.
- Nebeneinander** ► Zunächst wird geprüft, ob sich drei gleichfarbige Panels nebeneinander befinden. Dies geschieht, indem für jedes einzelne Feldelement in der Methode `NebenPrüfen()` geprüft wird, ob es selbst und seine beiden rechten Nachbarn mit einem Panel belegt sind, und ob diese Panels gleichfarbig sind. Die Prüfung beginnt beim Panel unten links und setzt sich bis zum drittletzten Panel der gleichen Zeile fort. Anschließend werden die Panels in der Zeile darüber geprüft usw.
- Panels löschen** ► Sobald eine Reihe gleichfarbiger Panels gefunden wurde, werden alle drei Panels mithilfe der Methode `Remove()` aus der Collection der Steuerelemente des Formulars gelöscht, d. h., sie verschwinden aus dem Formular. Ihre Position im Feld `F` wird mit `-1` (= `Leer`) besetzt. Nun müssen noch alle Panels, die sich eventuell oberhalb der drei Panels befinden, um eine Position abgesenkt werden. Die Variable `Neben` wird auf `true` gesetzt. Die doppelte Schleife wird sofort verlassen.
- Übereinander** ► Analog wird nun in der Methode `ÜberPrüfen()` geprüft, ob sich drei gleichfarbige Panels übereinander befinden. Ist dies der Fall, so werden

sie aus der Collection der Steuerelemente des Formulars gelöscht. Ihre Positionen im Feld `F` werden mit `-1` besetzt. Über den drei Panels können sich keine weiteren Panels befinden, die entfernt werden müssten.

- Falls durch eine der beiden Prüfungen eine Reihe gefunden und entfernt wurde, so wird die Schwierigkeitsstufe erhöht und das Timer-Intervall verkürzt. Nun muss geprüft werden, ob sich durch das Nachrutschen von Panels wiederum ein Bild mit drei gleichfarbigen Panels über- oder nebeneinander ergeben hat. Die Methode `AllePrüfen()` ruft sich also so lange selbst auf (rekursive Methode), bis keine Reihe mehr gefunden wird.

Rekursiv

11.1.8 Panels seitlich bewegen

Mithilfe der beiden Ereignismethoden `cmdLinks_Click()` und `cmdRechts_Click()` werden die Panels nach links bzw. rechts bewegt, falls dies möglich ist:

```
private void cmdLinks_Click(...)
{
    if (F[PZ, PS - 1] == Leer)
    {
        Panel p = (Panel) PL[PX];
        p.Left = p.Left - 20;
        PS = PS - 1;
    }
}

private void cmdRechts_Click(...)
{
    if (F[PZ, PS + 1] == Leer)
    {
        Panel p = (Panel) PL[PX];
        p.Left = p.Left + 20;
        PS = PS + 1;
    }
}
```

Listing 11.5 Projekt »Tetris«, Panels seitlich bewegen

Zur Erläuterung:

- Es wird geprüft, ob sich links bzw. rechts vom aktuellen Panel ein freies Feldelement befindet. Ist dies der Fall, so wird das Panel nach links bzw. rechts verlegt und die aktuelle Spaltennummer verändert.

Seitlich

11.1.9 Panels nach unten bewegen

Die Ereignismethode `cmdUnten_Click()` dient zur wiederholten Bewegung der Panels nach unten, falls dies möglich ist. Diese Bewegung wird so lange durchgeführt, bis das Panel auf die Spielfeldbegrenzung oder auf ein anderes Panel stößt. Der Code lautet:

```
private void cmdUnten_Click(...)
{
    while (F[PZ + 1, PS] == Leer)
    {
        Panel p = (Panel) PL[PX];
        p.Top = p.Top + 20;
        PZ = PZ + 1;
    }
    F[PZ, PS] = PX;      // Belegen
    AllePrüfen();
    NächstesPanel();
}
```

Listing 11.6 Projekt »Tetris«, Panels nach unten bewegen

Zur Erläuterung:

- Nach unten** ► Es wird geprüft, ob sich unter dem aktuellen Panel ein freies Feldelement befindet. Ist dies der Fall, so wird das Panel nach unten verlegt und die aktuelle Zeilennummer verändert. Dies geschieht so lange, bis das Panel auf ein Hindernis stößt.
- Anschließend wird das betreffende Feldelement belegt. Es wird geprüft, ob nun eine neue Reihe von drei gleichfarbigen Panels existiert und das nächste Panel wird erzeugt.

11.1.10 Pause

- Spiel anhalten** Abhängig vom aktuellen Zustand wird durch Betätigen des Buttons PAUSE in den Zustand *Pause* geschaltet oder wieder zurück:

```
private void cmdPause_Click(...)
{
    timT.Enabled = !timT.Enabled;
}
```

Listing 11.7 Projekt »Tetris«, Pause

Zur Erläuterung:

- Der Zustand des Timers wechselt zwischen `Enabled = true` und `Enabled = false`.

11.2 Lernprogramm Vokabeln

In diesem Abschnitt wird ein kleines, erweiterungsfähiges Vokabel-Lernprogramm (Projekt Vokabeln) vorgestellt. Es beinhaltet:

- eine Datenbank als Basis
- ein Hauptmenü
- die Nutzung einer ArrayList
- einen Zufallsgenerator
- eine Benutzerführung, abhängig vom Programmzustand
- Lesen einer Textdatei

11.2.1 Benutzung des Programms

Nach dem Start erscheint die Benutzeroberfläche, siehe Abbildung 11.2.

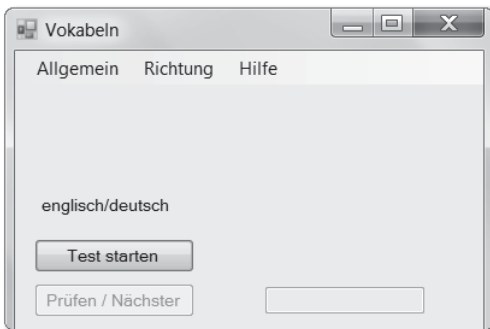


Abbildung 11.2 Projekt »Vokabeln«, Benutzeroberfläche