EUROPA-FACHBUCHREIHE
Programmierung für die
IT-Ausbildung

C++ für IT-Berufe

```
#include <iostream>
using namespace std;

int main()
{
     cout << "Informationsteil:" << endl;
     cout << "     - Standard-C++" << endl;
     cout << "     - C++ unter WinRT:" << endl;
     cout << "          ~ C++11/CX Grundlagen"<<endl;
     cout << "          ~ Windows Store-Apps" << endl;
     cout << endl;
     cout << "Aufgabenpool" << endl;
     cout << endl;
     cout << "Lernsituationen" << endl;
     cout << endl;
     cout << endl;
     return 0;
}
```

2. Auflage

VERLAG EUROPA-LEHRMITTEL · Nourney, Vollmer GmbH & Co. KG
Düsselberger Str. 23 42781 Haan-Gruiten

Europa-Nr.: 85498

Verfasser:
Dirk Hardy, 46049 Oberhausen

Verlagslektorat:
Alexander Barth

2. Auflage 2013

Druck 5 4 3

Alle Drucke derselben Auflage sind parallel einsetzbar, da sie bis auf die Behebung von Druckfehlern untereinander unverändert sind.

ISBN 978-3-8085-8550-4

http://www.europa-lehrmittel.de
Satz: Reemers Publishing Services GmbH, Krefeld
Umschlag: braunwerbeagentur, 42477 Radevormwald
Umschlagfotos: matttilda-fotolia.com, Gina Sanders-fotolia.com
Druck: Medienhaus Plump, 53619 Rheinbreitbach

Vorwort

Die Entwicklung von Anwendungen geht heutzutage weit über das reine Programmieren hinaus. Die Anforderungen an den **Fachinformatiker** bzw. Informatiker mit der **Fachrichtung Anwendungsentwicklung** sind sehr komplex geworden. Es reicht nicht mehr aus, mit einer speziellen Programmiersprache zu arbeiten. Vielmehr sind umfangreiche Kenntnisse in mehreren Programmiersprachen, in Datenbankmanagementsystemen und weiteren Gebieten wie der Client-Server-Programmierung, Web-Programmierung oder auch App-Programmierung nötig. Das Gemeinsame dieser verschiedenen Aspekte ist die Objektorientierung bzw. die Hinwendung zur Objektorientierung. Die modernen Programmiersprachen sind objektorientiert, die Datenbankmanagementsysteme werden in Zukunft verstärkt objektorientiert sein. Mischformen wie objektrelationale Datenbanken sind bereits im Einsatz. Eine gute Grundlage für diese Anforderungen ist die Erlernung der Sprache C++. Die Sprache ist plattformunabhängig und objektorientiert. Sie ist schwerer zu erlernen als andere Sprachen, bietet allerdings dafür einige Vorteile:

- Geschwindigkeit
- Weite Verbreitung
- Sehr viele Tools und Bibliotheken
- Compiler und Entwicklungswerkzeuge auf unterschiedlichen Plattformen
- Grundlage vieler weiterer Sprachen wie Java, Javascript, Perl oder auch PHP und C#

Die letzte ISO-Standardisierung der Sprache C++ fand 2011 statt. Damit verfügt C++ über viele neue Konzepte, die in anderen modernen Programmiersprachen (wie C#) bereits umgesetzt worden sind. Beispielsweise kann **C++11** (so die offizielle Bezeichnung) nun auch mit *Lambda-Ausdrücken* (einem Konzept der anonymen Funktionen) umgehen. Die vorliegende zweite Auflage dieses Buches nimmt diese Neuerungen in einem eigenen Kapitel auf. Darauf aufbauend bietet das Buch auch einen Einstieg in die komponentenbasierte Erweiterung **C++/CX** sowie die **App-Programmierung** unter Windows 8.

Den Lesern dieses Buches bleibt viel Erfolg bei der Erlernung von C++ zu wünschen und mit einem Zitat des Erfinders von C++ (Bjarne Stroustrup) zu schließen: „Die Frage *Wie schreibt man ein gutes C++ -Programm?* hat sehr viel Ähnlichkeit mit der Frage *Wie schreibt man gute englische Prosa?* Darauf gibt es zwei Antworten: Wisse, was Du sagen willst und Übe. Halte Dich an gute Vorbilder. Beide Antworten gelten für C++ ebenso wie für Englisch – und sind ebenso schwer zu befolgen."

Für Anregungen und Kritik zu diesem Buch sind wir Ihnen dankbar (gerne auch per E-Mail).

Dirk Hardy
E-Mail: Hardy@DirkHardy.de

Im September 2013

Verlag Europa-Lehrmittel
E-Mail: Info@Europa-Lehrmittel.de

Aufbau des Buches

Das vorliegende Buch möchte die Sprache C++ möglichst anschaulich, praxis- und unterrichtsnah vermitteln. Damit verfolgt dieses Buch einen **praktischen Ansatz**. Es ist die Ansicht des Autors, dass gerade in der schulischen Ausbildung der Zugang zu den komplexen Themen der Programmierung verstärkt durch anschauliche und praktische Umsetzung vorbereitet werden muss. Anschließend können allgemeine und komplexe Aspekte der Programmierung oder auch der Softwareentwicklung besser verstanden und umgesetzt werden.

Das Buch ist in **drei Teile** getrennt. Der **erste Teil** des Buches dient als **Informationsteil** und bietet eine **systematische Einführung in die Sprache C++**. Es werden sowohl die strukturierte Programmierung als auch die objektorientierte Programmierung behandelt.

Ein ausführlicher Einstieg in die **App-Programmierung** rundet den Informationsteil ab. Dabei werden zuerst die Spracherweiterung C++11/CX für die Programmierung der Windows Runtime-API und dann die Grundlagen der Microsoft Store-Apps betrachtet.

Der **zweite Teil** des Buches ist eine **Sammlung von Übungsaufgaben**. Nach der Erarbeitung der entsprechenden Kenntnisse aus dem Informationsteil können die Aufgaben aus diesem Teil zur weiteren Auseinandersetzung mit den Themen dienen und durch verschiedene Schwierigkeitsgrade auch die Differenzierung im Unterricht ermöglichen.

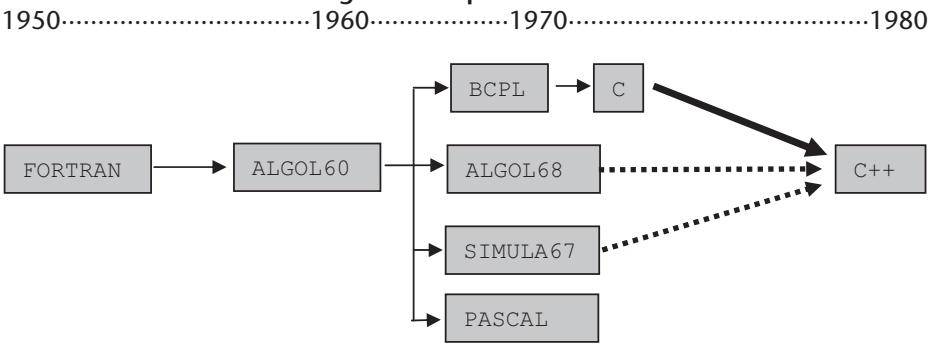
Der **dritte Teil** des Buches beinhaltet **Lernsituationen** basierend auf dem Lernfeld Entwickeln und Bereitstellen von Anwendungssystemen aus dem Rahmenlehrplan für die IT-Berufe (speziell Fachinformatiker-Anwendungsentwicklung). Lernsituationen konkretisieren sich aus den Lernfeldern und sollen im Idealfall vollständige Handlungen darstellen (Planen, Durchführen, Kontrollieren). Aus diesem Grund werden die Lernsituationen so angelegt, dass neben einer Planungsphase nicht nur die Durchführung (Implementation des Programms) im Blickpunkt steht, sondern auch geeignete Testverfahren zur Kontrolle des Programms bzw. des Entwicklungsprozesses in die Betrachtung einbezogen werden. Die Lernsituationen können aber auch als **Projektideen** verstanden werden.

Das Buch ist für alle berufsbezogenen Ausbildungsgänge im IT-Bereich konzipiert. Durch die differenzierten Aufgabenstellungen kann es in allen IT-Berufen (speziell Fachinformatiker), aber auch von den informationstechnischen Assistenten genutzt werden.

Als Entwicklungswerkzeuge werden in diesem Buch die *Express-Editionen* ***Visual Studio 2012 für Desktop sowie Visual Studio 2012 für Windows 8*** von Microsoft genutzt. Diese Entwicklungsumgebungen sind kostenfrei als Download im Internet verfügbar. Die Links zu der benutzten Software und alle Quelltexte des Buches finden Sie ständig aktualisiert auf der Seite **http://www.dirkhardy.de** unter der Rubrik „Programmierbücher“.

Dirk Hardy September 2013

1.1.3 Kleiner Stammbaum der Programmiersprachen



C++ im Überblick:

- Entwickelt von Bjarne Stroustrup in den 80er-Jahren.
- Basiert auf der Sprache C und hat Konzepte der Sprachen SIMULA67 und ALGOL68 übernommen.
- C++ ist eine objektorientierte Sprache.
- Sie ist plattformunabhängig, systemnah und sehr schnell.
- Viele Betriebssysteme wurden in C++ programmiert.
- Die Sprache ist sehr weit verbreitet und kommt in technischen, kaufmännischen und wissenschaftlichen Anwendungen zum Einsatz.

1.2 Bestandteile eines C++-Programms

1.2.1 Was ist ein Programm?

Ein Programm besteht aus einer Folge von Anweisungen an den Computer. Diese Anweisungen (Befehle) werden in einer bestimmten Form gegeben – und zwar in einer speziellen Sprache (Programmiersprache).

Diese Programmiersprache wird vom Computer nicht direkt verstanden, sondern muss erst „übersetzt" werden, und zwar in eine für den Computer verständliche Sprache, den **Maschinencode.**

Nach der Übersetzung kann das Programm gestartet werden.

```
Anweisung 1;
Anweisung 2;
Anweisung 3;
Anweisung 4;
Anweisung 5;
:
:
:
Anweisung N;
```

Eine endliche Folge von eindeutigen Anweisungen an den Computer nennt man **Algorithmus.**

1.2.2 C++-Quellcode

Die Anweisungen an den Computer in der Sprache C++ werden in Dateien gespeichert. Der Inhalt dieser Dateien wird als **Quellcode** bezeichnet.

Normalerweise haben C++-Quellcode-Dateien die Endungen

- `cpp` für Cplusplus
- `h bzw. hpp` für Header bzw. Header-plusplus.

2 Das erste C++-Programm

2.1 Ausgabe auf dem Bildschirm

Für die Ausgabe auf dem Bildschirm sind einige Vorbereitungen zu treffen.

Neben dem Kennenlernen der Entwicklungsumgebung, damit überhaupt ein Programm geschrieben werden kann, müssen noch weitere Punkte geklärt werden:

- Wie werden ein C++-Projekt und eine Quellcode-Datei angelegt?
- Welche Bibliothek muss für die Ausgabe auf dem Bildschirm eingebunden sein?
- Wie sieht das „Hauptprogramm“ aus?
- Mit welcher Anweisung wird eine Bildschirmausgabe erreicht?

Die folgenden Kapitel geben Aufschluss über diese Punkte.

2.1.1 Ein C++-Projekt in Visual Studio 2012 anlegen

Die integrierte Entwicklungsumgebung ***Visual Studio2012 für Desktop*** ist eine komfortable Umgebung, um C++-Programme zu entwickeln. Besonders erfreulich ist der Umstand, dass die Umgebung kostenfrei als *Express Edition* im Internet bereit steht. Ein C++-Programm besteht aus einer oder mehreren Quellcodedateien. Diese Dateien werden in einem Projekt organisiert.

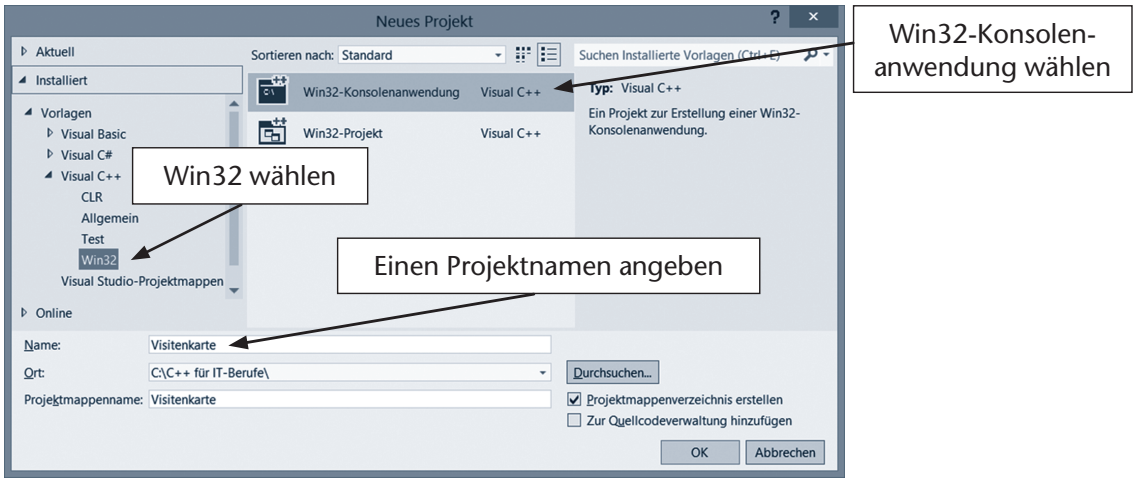
Visual C++ unterscheidet im Prinzip vier verschiedene Projektarten:

- Win32-Projekt (eine Windows-Anwendung)
- **Win32-Konsolenanwendung (ähnlich einem DOS-Programm)**
- CLR-Konsolenanwendung (die Konsolenanwendung unter .NET)
- Klassenbibliothek (Sammlung von Klassen unter .NET)

In diesem Buch ist hauptsächlich eine Projektform von Bedeutung: **die Win32-Konsolenanwendung**. Diese Form ist ausreichend, um die Programme in C++ auf dem Bild-schirm darzustellen. In den späteren Kapiteln wird dann ein Umstieg auf eine weitere Entwicklungsumgebung nötig sein, die allerdings nur unter Windows 8 lauffähig ist. Die Konsolenanwendung ist natürlich nicht so ansprechend wie ein Windows-Programm oder eine App, aber um die Sprache C++ zu lernen völlig ausreichend.

Anlegen eines neuen Projektes:

- Starten Sie ***Visual Studio Express 2012 für Desktop***
- Wählen Sie den Menüpunkt *Datei → Neues Projekt*



5 Selektion und Iteration

Die Selektion (Auswahl) und die Iteration (Wiederholung) sind zwei sehr wichtige Konstrukte in einer Programmiersprache. In den vorherigen Kapiteln sind schon einige Probleme ohne diese Konstrukte gelöst worden, aber große und immer komplexer werdende Programme können ohne Selektion und Iteration nicht auskommen.

5.1 Die Selektion

5.1.1 Darstellung der Selektion mit einem Programmablaufplan

Problemstellung:

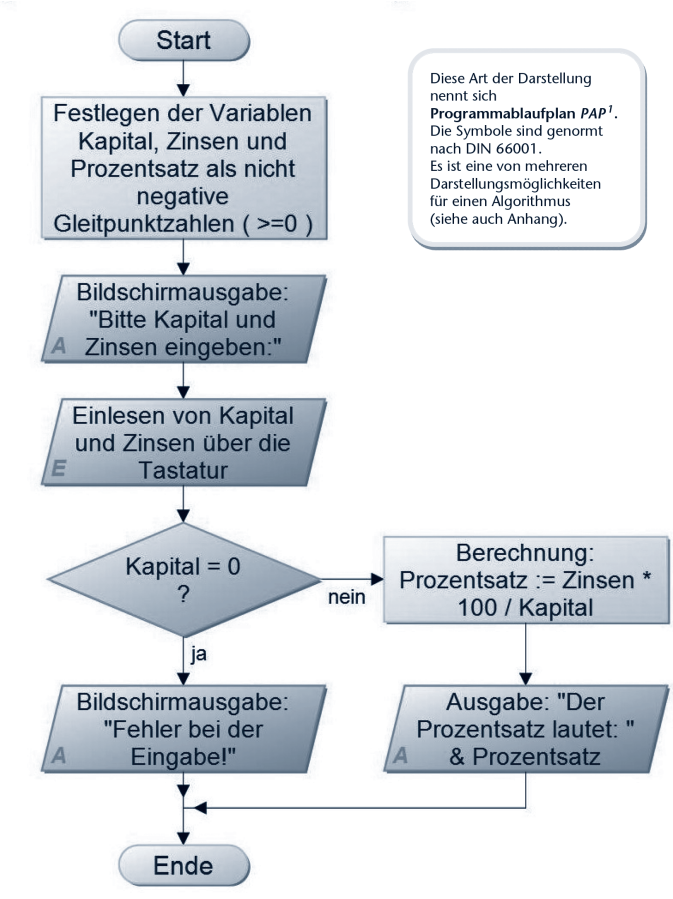
Es soll eine Berechnung des Prozentsatzes bei gegebenem Kapital und Zinsen durchgeführt werden.

Formel: $p = \frac{Z * 100}{K}$

Das Programm kann einen Fehler verursachen, wenn für das Kapital der Wert Null eingegeben wird (die Division durch Null ist verboten).

Lösungsmöglichkeit:

Das Programm erkennt, ob eine Null eingegeben wurde, und führt dann keine Berechnung durch. Das Problem wird durch eine Dokumentationstechnik, den Programmablaufplan, zuerst schematisch erfasst. Anschließend wird dann auf die Umsetzung in C++ eingegangen.



1 Der Programmablaufplan wurde mit dem Programm *PapDesigner* erstellt. Diese Software wurde speziell für die Ausbildung im IT-Bereich entwickelt und steht kostenfrei im Internet zum Download bereit.

Die Vorstellung eines zweidimensionalen Arrays entspricht einer Tabelle. Tabellen sind allgemein bekannt: Sie bestehen aus Zeilen und Spalten, die den entsprechenden Index des Arrays widerspiegeln.

Beispiel:

Es wird ein zweidimensionales Array angelegt.

```
float  Tab  [ 3 ] [ 4 ];
```

entspricht der Anzahl der Spalten

entspricht der Anzahl der Zeilen

	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Zeile 0	1.5	7	12.33	77.5
Zeile 1	124	99.99	453	67.89
Zeile 2	12	90.2	2727.5	22

```
Tab  [ 1 ] [ 2 ] == 453
```

Spaltenangabe

Zeilenangabe

Dreidimensionale Arrays kann man sich als eine Sammlung von Tabellenblättern vorstellen, die hintereinander angeordnet sind.

Beispiel:

Es wird ein dreidimensionales Array angelegt.

```
float  Tab  [ 3 ] [ 3 ] [ 4 ];
```

Blatt 2	Spalte 0	Spalte 1	Spalte 2	Spalte 3
Blatt 1	**Spalte 0**	**Spalte 1**	**Spalte 2**	**Spalte 3**
Blatt 0	**Spalte 0**	**Spalte 1**	**Spalte 2**	**Spalte 3**
Zeile 0	1.5	7	12.33	77.5
Zeile 1	124	99.99	453	67.89
Zeile 2	12	90.2	2727.5	22

				25.3
			45	99.2
			45.55	10
			78.6	

```
Tab  [ 1 ] [ 1 ] [ 3 ] == 45.55
```

Spaltenangabe

Zeilenangabe

Tabellenblatt

Nach der dritten Dimension hört die menschliche Vorstellungskraft auf. Die mehrdimensionalen Arrays mit mehr als drei Dimensionen sind dann auch nicht mehr so konkret vorstellbar wie beispielsweise die Tabellen, aber dennoch sind sinnvolle Einsätze dieser Arrays denkbar.

10 Das Klassenkonzept in C++

Die Sprache C++ ist eine Hybridsprache. Sie ist sowohl zur strukturierten Programmierung als auch zur objektorientierten Programmierung geeignet. Die ersten neun Kapitel dieses Buches waren eine Einführung in die strukturierte Programmierung mit C++. Diese Grundlagen sind wichtig, um das Konzept der objektorientierten Programmierung verstehen und umzusetzen zu können.

Unter objektorientierter Programmierung kann eine spezielle Art der Programmierung verstanden werden, die versucht, gewisse Gegebenheiten möglichst realitätsnah umzusetzen.

Im Mittelpunkt der objektorientierten Programmierung steht das Objekt bzw. die Klasse. Eine Klasse kann als eine Art Bauplan betrachtet werden, mit dem Objekte gebildet werden können. Die Begriffe Objekt und Klasse werden nun näher betrachtet.

Was ist ein Objekt?

Ein Objekt ist eine softwaretechnische Repräsentation eines realen oder gedachten, klar abgegrenzten Gegenstandes oder Begriffs.

Das Objekt erfasst alle Aspekte des Gegenstandes durch Attribute (Eigenschaften) und Methoden.

Was sind Attribute und Methoden?

Attribute sind die Eigenschaften des Objektes. Sie beschreiben den Gegenstand vollständig. Attribute sind geschützt gegen Manipulation von außen (das nennt man Kapselung).

Methoden beschreiben die Operationen, die mit dem Objekt (bzw. seinen Attributen) durchgeführt werden können. Von außen erfolgt der Zugriff auf Attribute durch die Methoden.

Was ist eine Klasse?

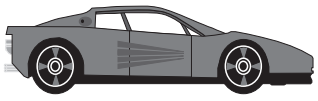
Unter einer Klasse versteht man die softwaretechnische Beschreibung eines Bauplanes für ein Objekt.

Aus einer Klasse können dann Objekte abgeleitet (gebildet) werden.

Diese etwas abstrakten, aber wichtigen Begriffsdefinitionen sollen nun anhand von Beispielen veranschaulicht werden.

Beispiel:

Diese Rennwagen sind konkrete Objekte. Sie haben Attribute wie Farbe, Leistung in KW oder Hubraum.



Name:	Lotus
Farbe:	grau
KW:	250
Hubraum:	4 Liter

Name:	Spider XL
Farbe:	schwarz
KW:	300
Hubraum:	5 Liter

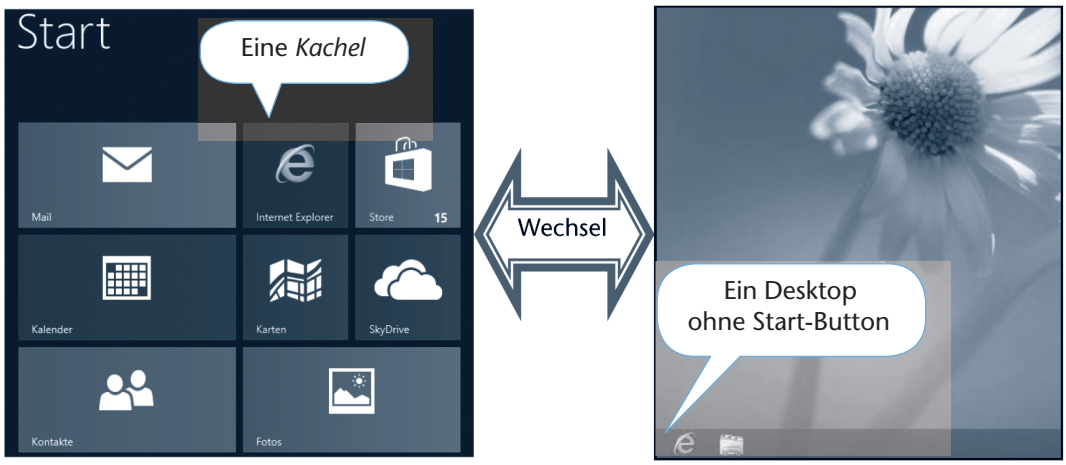
16 C++ und Windows Store Apps

16.1 Applikationen unter Windows 8

16.1.1 Neue Windows-Konzeption

Mit Windows 8 brachte Microsoft 2012 ein neues Betriebsystem heraus, das sich konzeptionell deutlich von den vorherigen Systemen unterscheidet. Auf der einen Seite ist es auf die Anforderungen der mobilen Geräte (Smartphones und Tablet-PCs) vorbereitet, indem es eine völlig veränderte Oberfläche anbietet, auf denen die Anwendungen in Form von *Kacheln* dargestellt werden. Diese Kacheln dienen auf den mobilen Geräten zur manuellen Eingabesteuerung (Touchscreen). Die Anwendungen, die mit solchen Kacheln gestartet werden, heißen *Windows Store-Apps* und zeichnen sich unter anderem dadurch aus, dass sie den ganzen Bildschirm ausfüllen und nicht mehr von einem Fensterrahmen begrenzt werden. Zusätzlich sind diese Apps ausschließlich über den Windows Store zu beziehen und werden in einer völlig anderen Form installiert als die konventionellen Windows-Applikationen. Die Entwicklung solcher Apps[1] wird das Thema der folgenden Kapitel sein.

Auf der anderen Seite bietet Windows 8 aber auch noch eine klassische Desktop-Oberfläche, die ähnlich wie die Oberfläche von Windows 7 (oder auch Windows XP) zu nutzen ist. Es fehlt nur der gewohnte Start-Button, der aber einfach mit kostenfreien Tools nachzuinstallieren ist.



16.1.2 Windows Runtime

Die Entwicklung von Windows Store Apps basiert auf einer völlig neuen Schnittstelle, der **Windows Runtime API** (API = **A**pplication **P**rogramming **I**nterface). Diese objektorientierte Schnittstelle basiert auf dem COM[2]-Modell und wurde in C++ geschrieben. Im Gegensatz zu der .NET-Technologie steht die Windows Runtime für eine native Programmentwicklung. Das bedeutet, dass die Programmentwicklung sofort auf eine Schnittstelle zugreift, die Bestandteil des Betriebssystems ist. Damit entfällt eine Zwischenschicht wie unter .NET. Ein großer Vorteil ist die schnelle Geschwindigkeit solcher Anwendungen. Der native Zugriff auf die Windows Runtime geschieht deshalb auch mit C++ (genauer gesagt mit der Erweiterung der Sprache C++/CX, dazu später mehr). Aber auch für andere Sprachen wie C# oder Visual Basic ist der Zugriff über bestimmte Klassen (*Wrapper*[3]-Klassen) ebenfalls möglich – allerdings müssen dann Übersetzungen von der .NET-Technologie (dem ver-

1 Dazu müssen das Betriebssystem Windows 8 und das Visual Studio Express für Windows 8 installiert sein.
2 COM (**C**omponent **O**bject **M**odel) ist eine Technik von Microsoft, mit der Softwarekomponenten erstellt werden können. Dabei ist die Wahl der Programmiersprache unerheblich.
3 Wrapper-Klassen „umhüllen“ andere Klassen und bieten damit die Möglichkeit bestimmte Funktionalitäten auf ein anderes System zu übertragen.

16.2 C++11

16.2.1 Der neue Standard C++11

Der letzte C++-Standard liegt mehr als 10 Jahre zurück (ISO[5] 1999). Im Jahr 2011 war es dann endlich soweit und der neue Standard (deshalb auch C++11) wurde veröffentlicht. Die Sprache wurde von Grund auf überholt und es entstand eine moderne objektorientierte Sprache. Bislang galt C++ immer als schwer zu erlernen. Mit den neuen Standards sind einige Hürden für den Programmieranfänger gesenkt worden. Außerdem tritt C++ jetzt aus dem Schattendasein hervor, in das es seit den Markteinführungen von Java und C# gedrängt wurde – Microsoft stellt C++11 (bzw. die Erweiterung *C++11/CX*) mit dem neuen Betriebssystem Windows 8 wieder in den Mittelpunkt der Entwicklung.

Die folgende Tabelle zeigt einige wesentliche Neuerungen von C++11[6], die im Anschluss auch anhand von Beispielen erklärt werden.

Neuerung	Beschreibung
Der `auto`-Typ	Eine `auto`-Variable erhält seinen Datentyp durch die Initialisierung. Wird die `auto`-Variable beispielsweise mit einem nummerischen Wert initialisiert, so ersetzt der Compiler das `auto` durch ein `double`.
Intelligente Zeiger (*smart-pointer*)	Diese Art von Zeigern erleichtert das Handling mit dynamischer Speicherreservierung.
Lambda-Ausdrücke	Diese Ausdrücke verkörpern im Prinzip anonyme Funktionen und helfen beispielweise auf sehr elegante Weise bei der Suche in STL-Arrays.
Die neue `for`-Schleife	In vielen modernen Programmiersprachen bereits ein Standard (beispielweise `foreach` in C#): die neue `for`-Schleife erlaubt einfaches Iterieren durch ein Array.

16.2.2 Der auto-Typ

Dieser neue Typ vereinfacht die Definition von Variablen. Der Entwickler muss sich keine Gedanken mehr über den Datentyp machen, wenn er beispielsweise den Rückgabewert einer Funktion speichern will. Er setzt die Variable einfach auf den `auto`-Typ, wie das folgende Beispiel zeigt:

Beispiel:

```
#include <iostream>
using namespace std;

double Funktion()
{
   return 10.5;
}

int main()
{
   auto x = 11.5;
   auto y = Funktion();
   auto z = x + y;

   cout << z << endl;
   return 0;
}
```

Durch die Initialisierung erhält die `auto`-Variable ihren Typ.

16.2.3 Intelligente Zeiger

Die dynamische Speicherreservierung liegt auch unter C++11 in den Händen des Entwicklers, da C++11 kein verwalteter Code ist und deshalb auch kein *garbage-collector* wie in Java oder C# für die Aufräumarbeiten sorgt. Dementsprechend ist die Speicherreservierung immer eine Quelle von Fehlern. Die intelligenten Zeiger sorgen dafür, dass dieses Problem entschärft wird. Dazu werden neue Arten von Zeigern zu Verfügung gestellt: `unique_ptr` und `shared_ptr`.

5 ISO (**I**nternational **S**tandardization **O**rganization oder besser International Organization for Standardization) ist eine internationale Vereinigung, die in allen Bereichen (außer Elektronik und Elektrik) Normierungen vornimmt.

6 Die Neuerungen von C++ können natürlich nur dann angewendet werden, wenn die Entwicklungsumgebung entsprechend modern ist (beispielsweise das *Visual Studio Express 2012 für Desktop*).

16.4.3 Layout-Container

Das Inhaltskonzept von XAML sieht vor, dass alle Steuerelemente letztendlich einem Container zugewiesen werden. XAML bietet einige Layout-Container an, deren Eigenschaften hier kurz vorgestellt werden:

Layout-Container	Beschreibung
`StackPanel`	Die Elemente werden einfach untereinander oder nebeneinander angeordnet. Es können beliebig viele Elemente angefügt werden.
`WrapPanel`	Die Steuerelemente werden zeilenweise angeordnet. Wenn eine Zeile keinen Platz mehr hat, dann wird auf die nächste Zeile verteilt.
`DockPanel`	Die Elemente werden an verschiedenen Seiten „angedockt" (`Left, Top, Right, Bottom`).
`Grid`	Die Steuerelemente werden in einer Art Tabelle angeordnet. Zusätzlich kann ein Steuerelement über mehrere Tabellenzellen definiert werden.
`Canvas`	Dieser Container erlaubt eine Anordnung nach festen Koordinaten und entspricht am ehesten der bekannten GUI-Programmierung.

Beispiel:

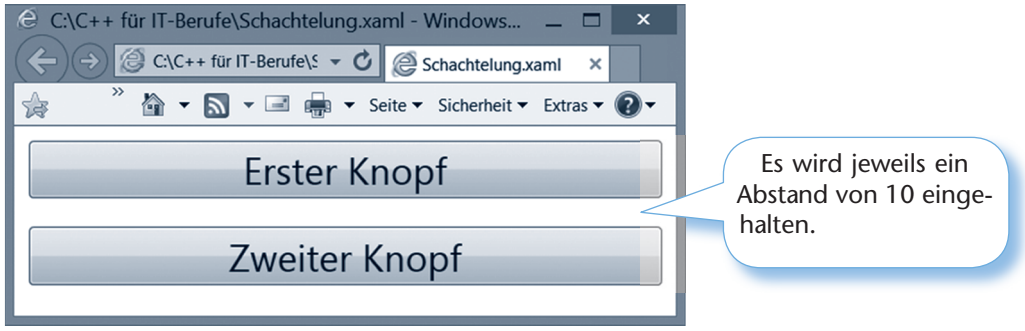
Das folgende Beispiel zeigt, wie in einem `StackPanel` zwei `Buttons` eingebettet werden:

```
<StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
presentation">

   <Button Name="knopf1" FontSize = "20pt" Margin = "10,10,10,10">
         Erster Knopf
   </Button>
   <Button Name="knopf2" FontSize = "20pt" Margin = "10,10,10,10">
         Zweiter Knopf
   </Button>

</StackPanel>
```

Nach den Starten des obigen *Loose-XAML* erscheinen die Buttons einwandfrei „gestapelt" – entsprechend dem Layout des `StackPanel`-Containers:



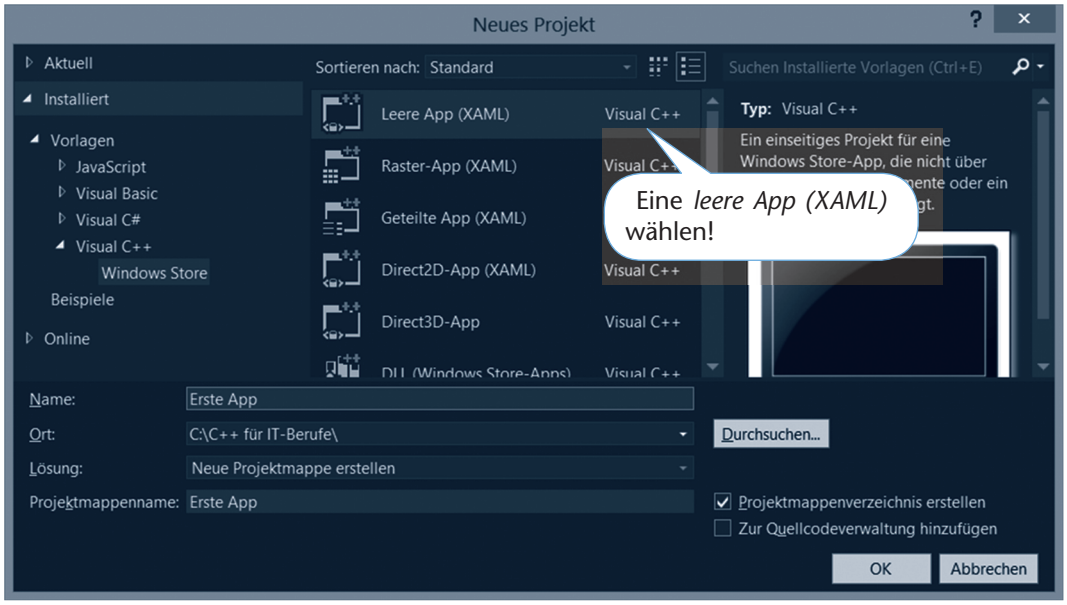
An diesen ersten Beispielen ist erkennbar, dass XAML hervorragend geeignet ist, um die visuelle Gestaltung einer GUI-Anwendung vorzunehmen. Die Logik der Anwendung wird dann in den *Code-behind*-Dateien in einer Sprache wie C++ umgesetzt.

17 Windows Store App-Entwicklung

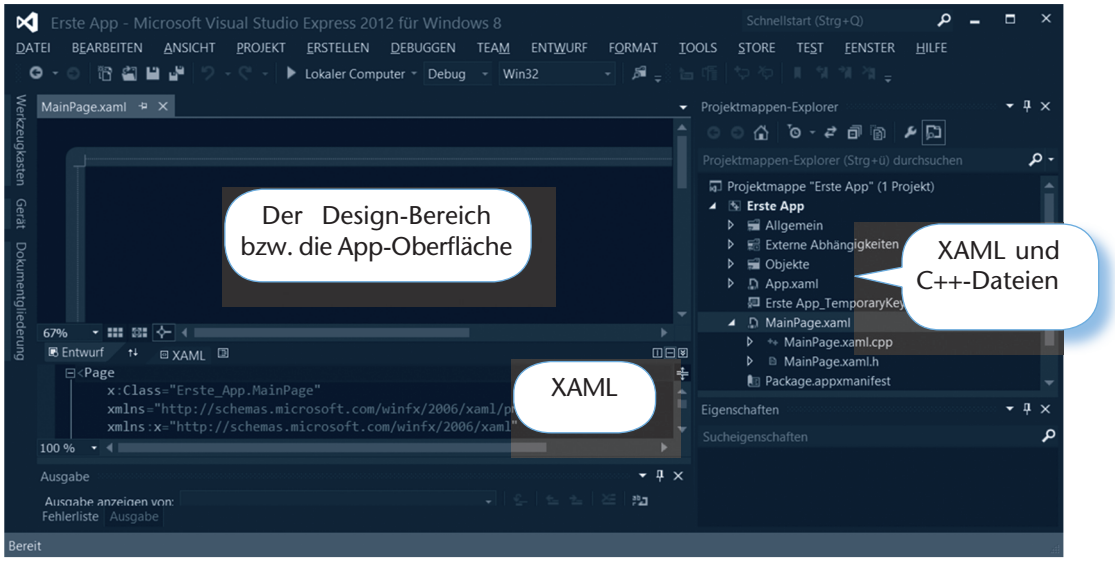
17.1 Die erste Windows Store App

17.1.1 Ein leeres App-Projekt anlegen

Mit der kostenfreien Entwicklungsumgebung *Visual Studio 2012 Express für Windows 8* werden nun Windows Store Apps entwickelt. Nach den Grundlagen zu C++/CX und XAML aus dem letzten Kapitel konzentrieren sich die Ausführungen nun auf die eigentliche App-Entwicklung mit dem Visual Studio. Die Entwicklungsumgebung hilft bei der Erstellung der GUI mit einem automatischen XAML-Generator. Die Logik hinter der GUI wird dann in den *Code-behind*-Dateien implementiert. Der erste Schritt ist das Anlegen eines neuen Projekts unter Visual Studio 2012:



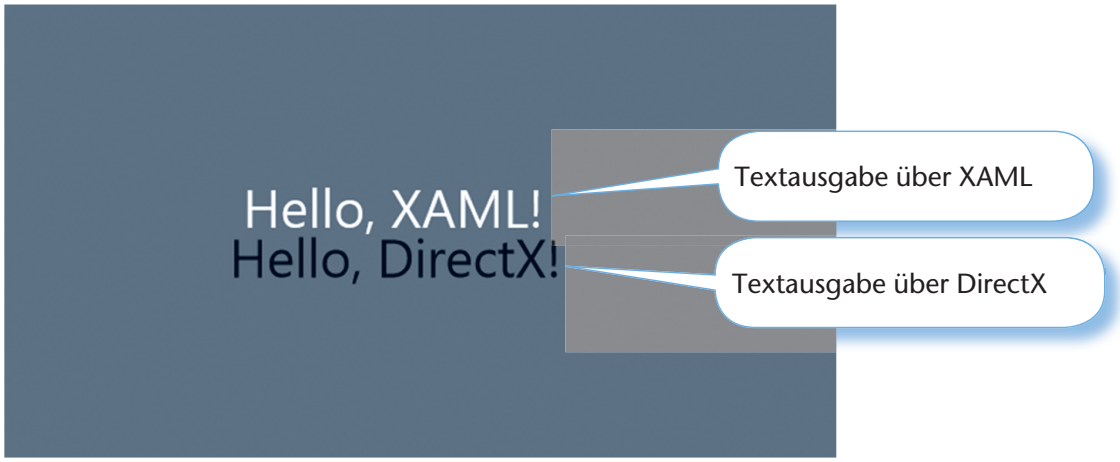
Nach dem Bestätigen mit „OK“ erscheint das das neue App-Projekt in der Entwicklungsumgebung:



Die Entwicklungsumgebung erstellt einige Quellcode-Dateien, die als Basis für die Anwendung dienen:

Datei	Beschreibung
BasicTimer.h	Enthält eine Klasse als Basis für die Zeitsteuerung.
DirectXBase.h/cpp	Diese Dateien enthalten die abstrakte Basisklasse `DirectXBase`. Sie dient als Grundlage zur Vererbung und liefert die wesentlichen Attribute und Methoden.
DirectXPage.xaml **DirectXPage.xaml.h/cpp**	Die XAML-Datei, die die Oberfläche beschreibt. Sie enthält ein besonderes Element: `SwapChainBackgroundPanel` Damit kann direkt auf die Oberfläche gezeichnet (gerendert) werden. Die *Code-behind*-Dateien sorgen für Logik im Hintergrund. Sie enthalten die Definition der Klasse `DirectXPage`, die beispielsweise für die Ereignissteuerung zuständig ist.
SimpleTextRenderer.h/cpp	Die Dateien zeigen eine einfache Umsetzung einer Klasse, die von der Basisklasse `DirectXBase` erbt. Sie enthält die wesentlichen Methoden und kann als Ausgangsbasis der eigenen Entwicklung verwendet werden.

Nach dem Starten erscheint der folgende Bildschirm:



Die App ist in der Lage den DirectX-Text mit der Maus (oder einem Zeigegerät) auf dem Bildschirm zu verschieben:

Die Speicherung der Matrix soll mithilfe der dynamischen Speicherreservierung erfolgen. Dabei sollen folgende Funktionalitäten in der Klasse umgesetzt werden:

`Methode Einlesen():`

Der Benutzer kann die Dimension der Matrix angeben und anschließend wird dynamisch Platz reserviert und die Werte eingelesen.

`Methode Ausgabe():`

Die Matrix wird (formatiert) auf dem Bildschirm ausgegeben.

`Methode Transponieren():`

Die Matrix wird transponiert. Das bedeutet, dass alle Matrixwerte aij ihre Indizes vertauschen (aij = aji).

Beispiel einer Transponierung:

$$\begin{bmatrix} 1 & 7 & 2 \\ 3 & 2 & 2 \\ 5 & 6 & 3 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 3 & 5 \\ 7 & 2 & 6 \\ 2 & 2 & 3 \end{bmatrix}$$

Zusätzlich sollen folgende Operatoren überladen werden:

- **`Gleichheitsoperator =`**
- **`Additionsoperator +`**
- **`Multiplikationsoperator *`**

Erläuterungen zur Addition und Multiplikation:

Zwei Matrizen werden addiert, indem jedes Element der einen Matrix zu dem Element mit denselben Indizes der anderen Matrix addiert wird.

Beispiel:

$$\begin{bmatrix} 1 & 7 & 2 \\ 3 & 2 & 2 \\ 5 & 6 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 6 & 5 \\ 7 & 1 & 6 \\ 2 & 2 & 7 \end{bmatrix} = \begin{bmatrix} 1+2 & 7+6 & 2+5 \\ 3+7 & 2+1 & 2+6 \\ 5+2 & 6+2 & 3+7 \end{bmatrix} = \begin{bmatrix} 3 & 13 & 7 \\ 10 & 3 & 8 \\ 7 & 8 & 10 \end{bmatrix}$$

Die Multiplikation ist etwas komplizierter (siehe auch Aufgabe 7.4):

Die erste Zeile der ersten Matrix wird mit der ersten Spalte der zweiten Matrix elementeweise multipliziert und anschließend werden die Produkte addiert. Das Endergebnis ist das erste Element der Multiplikationsmatrix.

Danach wird die erste Zeile der ersten Matrix mit der zweiten Spalte der zweiten Matrix elementeweise multipliziert und anschließend werden die Produkte addiert. Das Endergebnis ist das zweite Element der Multiplikationsmatrix usw.

Am anschaulichsten ist es mit einem einfachen Beispiel:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \bullet \begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 1\cdot 3+2\cdot 5 & 1\cdot 4+2\cdot 6 \\ 3\cdot 3+4\cdot 5 & 3\cdot 4+4\cdot 6 \end{bmatrix} = \begin{bmatrix} 13 & 16 \\ 29 & 36 \end{bmatrix}$$

Hinweise:

- Denken Sie an den Standardkonstruktor, den Copy-Konstruktor und den Destruktor sowie die korrekte Freigabe des reservierten Speichers.
- Achten Sie auf modulare Programmgestaltung.
- Für die dynamische Reservierung ist ein Zeiger auf einen Zeiger (Typ `int` oder `double`) nötig.

```
void MainPage::TICTACTOEPointerPressed(Platform::Object^ sender,
  Windows::UI::Xaml::Input::PointerRoutedEventArgs^ e)
{
  auto wer = (TextBlock^) sender;

  const wchar_t *wo = wer->Name->Data();
  int x = wo[1]-48;
  int y = wo[3]-48;
  :
}
```

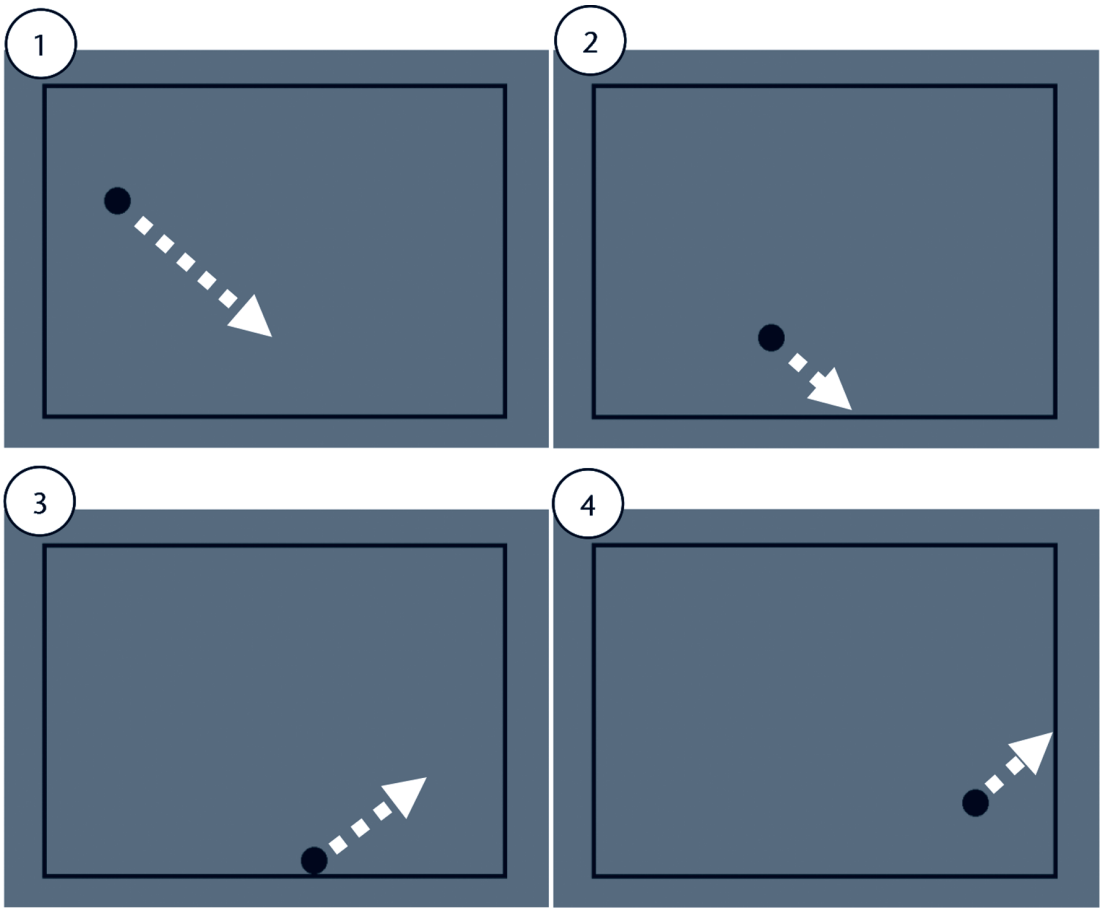
Textblock identifizieren

Position filtern

Aufgabe 17.2

Schreiben Sie eine Direct2D-App (XAML), die einen Ball (bzw. Kreis) in einem vordefinierten Rechteck springen läßt. Stößt der Ball an die Seiten des Rechtecks, so ändert er seine Richtung um 90°. Wenn der Benutzer auf die linke Maustaste (bzw. ein anderes Zeigegerät) drückt, dann soll sich die Ballgeschwindigkeit erhöhen.

Das fertige Programm könnte dann so aussehen:



Lernsituation 8:

Entwicklung einer Grafik-App für einfache Skizzen

Ausgangssituation:
Die Firma **ProSource** hat einige Kunden aus dem handwerklichen Bereich. Um die Kundenbindung zu stärken, möchte **ProSource** ihren Kunden eine kostenfreie Grafik-App zu Verfügung stellen. Damit sollen die Monteure vor Ort einfache Skizzen auf ihren Tablet-PCs anfertigen können, die dann beispielsweise später als Grundlage eines Kostenvoranschlages dienen können. Mit dem Einsatz der neusten App-Technik möchte **ProSource** seinen Kunden auch demonstrieren, dass bei Prosource mit aktueller Technik entwickelt wird.

Als Auszubildender der Firma **ProSource** erhalten Sie nun den Auftrag die Grafik-App zu entwickeln. In einem ersten Prototyp soll die App elementare Grafikobjekte wie Linien, Ellipsen und Rechtecke zeichnen können.

Arbeitsschritte in Einzel- oder Partnerarbeit:

Planung:
Für den ersten Prototyp hat der Teamleiter der Softwareentwicklung bereits einige Vorüberlegungen getroffen:

Die App soll über einige Steuerelemente in der sogenannten `AppBottomBar` verfügen. Das ist der untere Bereich der App, der sich bei einem Klick auf die rechte Maustaste in diesem Bereich öffnet: