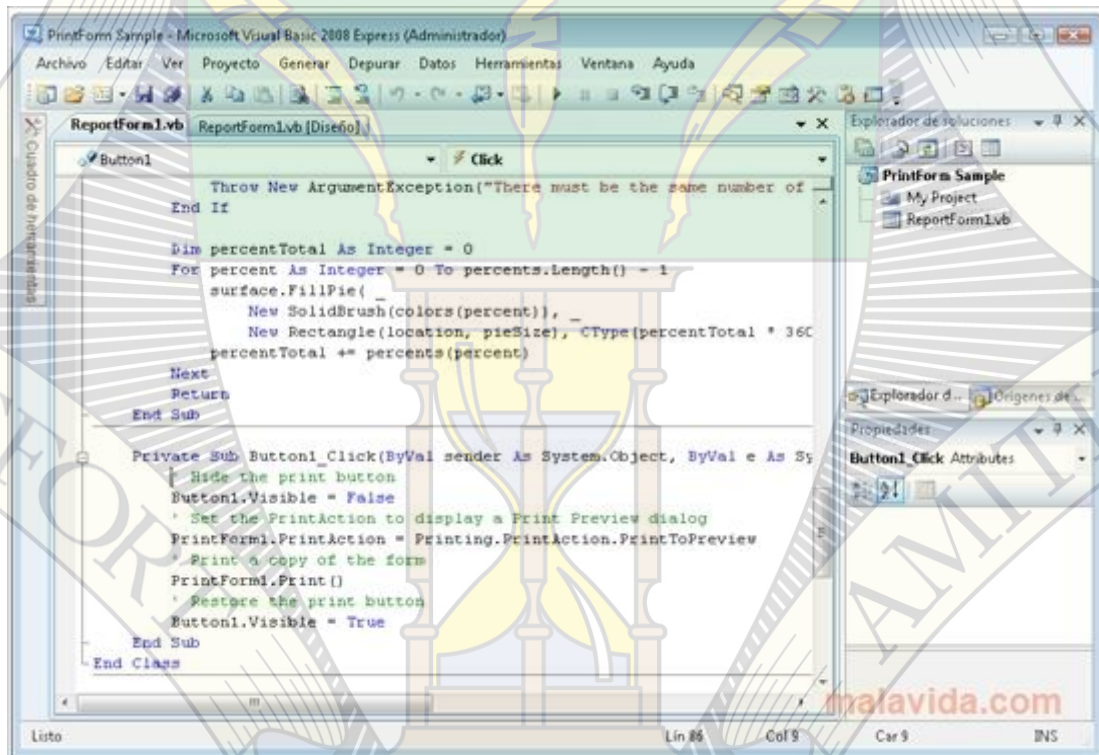


Visual Basic



Nick Sébastien

6_Tec_2

Semester 2: 2014

Kohl Séverine

Inhaltsverzeichnis :

1. Einleitung

2. Hauptteil

2.1. Übersicht

- a. Was ist Visual Basic
- b. Geschichtlicher Hintergrund
- c. Compiler (Umwandlung)

2.2. Einführung zum Schreiben eines Programmes

- a. Wichtige Wörter und Begriffe
- b. Userform
- c. Der Debugger

2.3. Einige von mir geschriebene Skripte (Programme)

- a. Ein Taschenrechner
- b. Primzahlenprogramm

3. Schluss

4. Quellenverzeichnis

1.Einleitung

Ich habe dieses Thema gewählt, weil ich bemerkt habe, dass heutzutage fast nichts ohne ein Computerprogramm funktioniert. Ich fragte mich immer wieder, was in einem Computer passiert, wenn man z.B.mit der Maus auf eine Icone klickt. Es fasziniert und interessiert mich, was heute in den technischen Geräten vor sich geht.

Anfangs arbeitete ich mit cmd, bis ich erfuhr, dass cmd nur einfache Komandos interpretiert.

Ich wollte jedoch mehr wissen und beschäftige mich seither mit Visual Basic (VB). Dies interessiert mich so sehr, dass ich ein Travail Personnel zu diesem Thema schreiben wollte um noch mehr dazuzulernen. In dieser Arbeit werde ich auch einige von mir geschriebene Skripte (Programme) beschreiben

2. Hauptteil

2.1 Übersicht

a. Was ist Visual Basic?

Visual Basic ist eine objektorientierte Programmiersprache die 1991 von Microsoft entwickelt worden ist. Ein großer Vorteil ist die einfache Erlernbarkeit. Hierbei werden bewusst englische Wörter anstatt Symbole benutzt. Einige Beispiele sind “And” statt “&&”, “Or” statt “||”, “Mod” statt “&”,....

Visual Basic Programme können in der Regel mit geringem Zeitaufwand erstellt werden. Man kann sie allerdings nur auf Windowsgeräten nutzen. Man kann gewisse Applicationen programmieren, um diese später zu nutzen, wie zum Beispiel ein Progammm zum Erkennen von Primzahlen, (siehe Kapilel 2.3.)

b. Geschichtlicher Hintergrund

Microsoft begann 1991 mit der Entwicklung von Visual Basics. Bis heute gibt es 6 Hauptversionen; die aktuelle ist die 6.0. In den späten 90er Jahren verlor Visual Basic jedoch an Bedeutung gegenüber anderen Hochsprachen (=Assemblerprogrammiersprachen) wie C und C++.

Visual Basic bot eine viel eingeschränkte Funktionalität und Erweiterbarkeit, wodurch es nicht für den professionellen Einsatz geeignet war. Der spätere historische Erfolg beruhte auf der einfachen Erlernbarkeit und der unkomplizierten Ausbaufähigkeit der Programmbibliothek.

Kritiker jedoch, bemängeln die mangelhafte Funktionalität. Oft ist die Rede von langsamer Ausführungsgeschwindigkeit bei leistungsschwachen Rechnern, (aus den 90er Jahren) welche jedoch heute bei moderneren Rechnern behoben ist.



c. Compiler

Der in Visual Basic benutzte Compiler heisst “ vbcomp ” .
Er wurde von Microsoft entwickelt.

Ein Compiler ist ein Computerprogramm, das ein zum Beispiel in Visual Basic geschriebenes Programm, in eine Form transformiert, die von einem Computer verstanden werden kann.

Ein Compiler umschreibt ein Programm aus seiner formalen Quellsprache in ein semantisches Äquivalent. Compiler sind spezielle Transformatoren die Programmcodes aus problemorientierten Programmiersprachen, sogenannten Hochsprachen, in eine Assemblersprache, maschinell ausführbare Maschinensprache oder Bytecode überführen. Der Vorgang der Transformierung wird auch als Kompilierung oder Umwandlung bezeichnet. Im umgekehrten Sinne nennt man diesen Vorgang Dekompilierung.

Die Entwicklung des Compilers

Der erste Compiler namens A-0 wurde 1949 von der Mathematikerin Grace Hopper entwickelt. Bis zu diesem Zeitpunkt reichten Programmierer ausschließlich Nullen und Einsen hintereinander, um ein Programm zu schreiben. Um diesen Prozess zu vereinfachen, entwickelte Grace Hopper eine Methode, die es ermöglichte, Programme und ihre Unterprogramme in einer mehr an der menschlichen als der maschinellen Sprache orientierten Weise auszudrücken. Am 3. Mai 1952 stellte Hopper den ersten Compiler A-0 vor. Anfang 1955 präsentierte Hopper bereits einen Prototyp des Compilers B-0, der nach englischen, französischen oder deutschen Anweisungen Programme erzeugte. Hopper nannte

ihren Vortrag zum ersten Compiler „The Education of a Computer“ („Die Ausbildung eines Computers“). Die Geschichte des Compilerbaus wurde von den jeweils aktuellen Programmiersprachen und Hardwarearchitekturen geprägt. Weitere frühe Meilensteine sind 1957 der erste FORTRAN-Compiler und 1960 der erste COBOL-Compiler. Viele Architekturmerkmale heutiger Compiler wurden aber erst in den 1960er Jahren entwickelt. Bis heute gibt es wenige Änderungen.

Die Arbeitsweise

Die prinzipiellen Schritte bei der Übersetzung eines Quellcodes in einen Zielcode lauten:

- Syntaxprüfung:

Es wird geprüft, ob der Quellcode ein gültiges Programm darstellt, also der Syntax der Quellsprache entspricht. Sie besteht aus der lexikalischen und der syntaktischen Analyse.

- Codeerzeugung:

Die optimierte Zwischendarstellung wird in entsprechende Befehle der Zielsprache übersetzt. Hierbei können weitere, zielsprachenspezifische Optimierungen vorgenommen werden.

Lexikalische Analyse

Die lexikalische Analyse zerteilt den eingelesenen Quelltext in lexikalische Einheiten (Tokens) verschiedener Typen, zum Beispiel Schlüsselwörter, Bezeichner, Zahlen, Zeichenketten

oder Operatoren. Dieser Teil des Compilers heißt Scanner oder Lexer.

Syntaktische Analyse

Die syntaktische Analyse überprüft, ob der eingelesene Quellcode ein korrektes Programm der zu übersetzenden Quellsprache ist, das heißt der Syntax (Grammatik) der Quellsprache entspricht. Dabei wird die Eingabe in einen Syntaxbaum umgewandelt. Der syntaktische Analysierer wird auch als Parser bezeichnet. Falls der Quellcode nicht zur Grammatik der Quellsprache passt, gibt der Parser einen Syntaxfehler aus.

2.2. Einführung zum Schreiben eines Programmes

a. Wichtige Wörter und Begriffe

Ein Beispiel von einem kleinem Testprogramm :

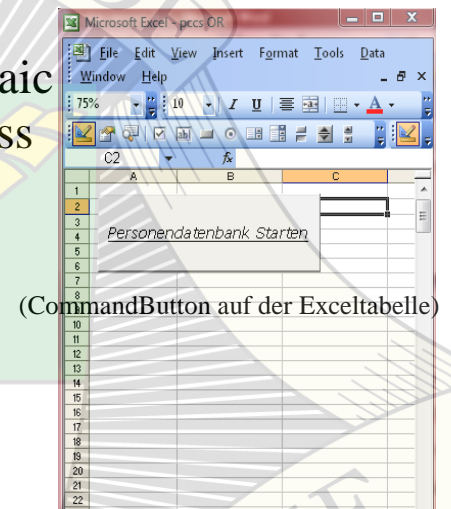
(In Visual Basic heisst die Schrift mit der man schreibt “Lucia Console”. Man schreibt normalerweise mit einer Schriftgrösse von 10 Punkten. Dies sieht dann folgendermassen aus:)

```
Private Sub CommandButton1_Click()  
    Do  
        MsgBox "Guten Morgen", vbExclamation  
        Name = InputBox("Wie heisst du?")  
        frage = MsgBox("OK, " & Name & ", isst du gerne    Pizza?",  
                        vbYesNo)  
        If frage = vbYes Then  
            MsgBox "Ich auch"  
        Else  
            MsgBox "Ich aber :)", vbExclamation  
        End If  
        arreter = MsgBox("Programm beenden?", vbYesNo)  
    Loop Until arreter = vbYes  
End Sub
```


In diesem Programm sagt der Computer zuerst “Guten Morgen”. Anschließend fragt er dich nach deinem Namen. Den schreibt er mit einem OK in eine MsgBox. Nun kann man entscheiden ob man den Programm nochmal abspielen will oder ihn beenden will.

- Private Sub CommandButton1_Click()

Mit Privat Sub_Click() fängt jedes Visual Basic Programm an. Dieser sagt dem Compiler, dass dort das Programm anfängt. Der Command-button1 steht dort, weil dieses Programm in einem Button gespeichert ist. Das heißt, dass auf der ExcelTabelle ein Button ist auf den man klicken kann um das Programm zu starten.



(CommandButton auf der Exceltabelle)

- Do...Loop until

Dies ist, wenn man im Programm zu einer bestimmten Stelle zurückkehrt(Das Programm setzt einen dorthin zurück.) z.B. bei einem Taschenrechner, man hat etwas gerechnet, dann fragt er dich ob du nochmal rechnen willst, wenn nein dann hört das Programm auf andernfalls kommt der Loop until zum Einsatz und man kann nochmal rechnen.

- Name = MsgBox

Eine MsgBox ist eine Meldebox und der “Name” ist der Name von dieser. Eine MsgBox kann man aber so nennen, wie man will, z.B. ao, Hallo, qwrtz,

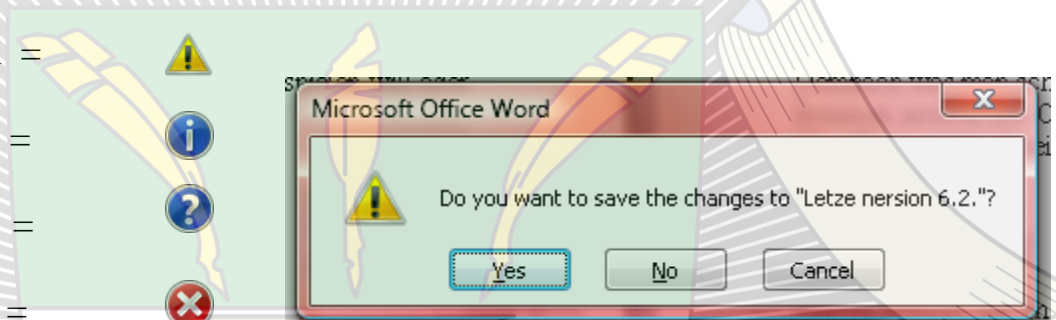
- (“...”)

Diese sorgen dafür, dass der Compiler daraus nichts macht, denn er soll dies einfach nur so in die MsgBox schreiben, ohne irgendetwas zu ändern.

- (“...”, vb...)

Hinter den ,vb kann man folgendes schreiben:

, vbExclamation =
, vbInformation =
, vbQuestion =
, vbCritical =



Je nachdem was man schreibt setzt er das folgende Zeichen in die MsgBox. Die kommen auch im Computer vor z.B. wenn man etwas in ein WordDokument schreibt und es schließt ohne zu speichern; dann setzt er den ,vbExclamation ein.

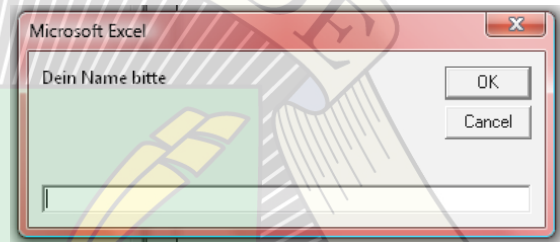
- If...then...else...end if

Hierbei können wir im laufendem Programm eine Entscheidung treffen. Mit If fängt es an, dies heißt auf Deutsch: wenn. Then ist dann, das bedeutet also: wenn...dann. Else ist ansonst, also wenn ... dann ..ansonst... . End if das kommt immer am Ende von einem if. Wenn man den nicht schreibt, gibt der computer eine Fehlermeldung aus. Ein Beispiel auf deutsch:

Wenn ac = "Ja" dann
 MsgBox "Hallo"
Ansonst
 MsgBox "Hey"
Ende dieser Entscheidungen

- InputBox

Dies ist das Gleiche wie eine MsgBox, nur hier gibt es eine weiße Zeile, auf die man etwas schreiben kann.



- Trulu = true

Dies ist das gleiche wie Do...Loop until. Man kann vor dem = schreiben was man will aber hinterher muss immer true stehen (Ich bevorzuge den Do...Loop until)

- End Sub

Dies sagt dem Compiler, dass dort das Programm fertig ist.

- ("...", vbYesNo)

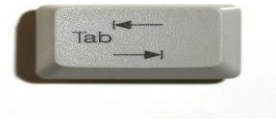
Es gibt mehrere solche Formeln aber alles was dort dann passiert geht auch nur, wenn man es programmiert:

, vbYesNo : Dieser macht, dass der Computer zwei Button in die MsgBox reinsetzt, einer auf dem Ja steht und einer auf dem Nein steht.

,... es gibt sehr viele solcher Buttons, die ich jetzt nicht alle aufzähle.

Die Schreibweise

Man versetzt die Zeilen, damit man sich besser zurechtfindet. In solchen kleineren Programmen ist das noch egal und stört nicht aber in größeren könnte dies sehr nützlich werden. Die Tasten werden mit der Tab-Taste verschoben. Damit hat man einen besseren Überblick, wo z.B. ein If anfängt und wo er aufhört. Ein Beispiel von einem kleinem Programm (die → stehen jedes mal für ein Tab):



```
Private Sub CommandButton1_Click()
```

```
→ If ao = vbYes Then
```

```
→ → MsgBox "Hallo"
```

```
→ Else
```

```
→ → MsgBox "Hey"
```

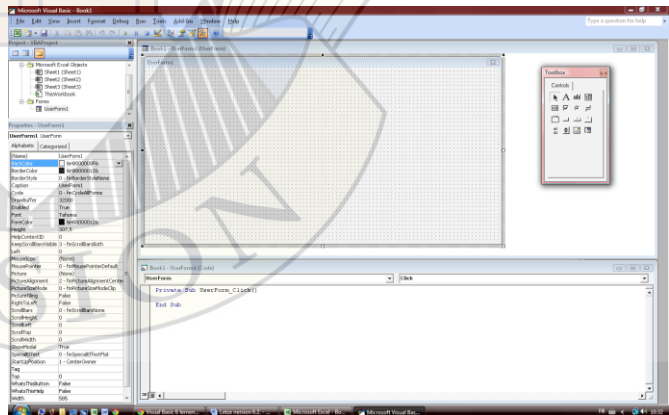
```
→ End If
```

```
End Sub
```

b. UserForm

Beim Visual Basic muss man nicht Seiten und Seiten mit Text programmieren, dies kann man auch einfacher mit Userformen erledigen. In diesem Fall sieht der Bildschirm folgendermaßen aus:

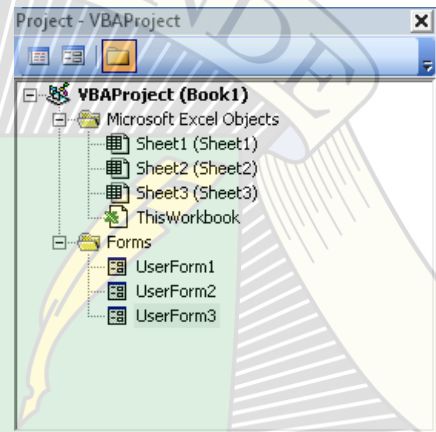
Diese Seite wird auch Hauptfenster genannt, da dies die sogenannte Schaltzentrale ist. Hiermit lassen sich die anderen Fenster aktivieren sowie Programme öffnen,



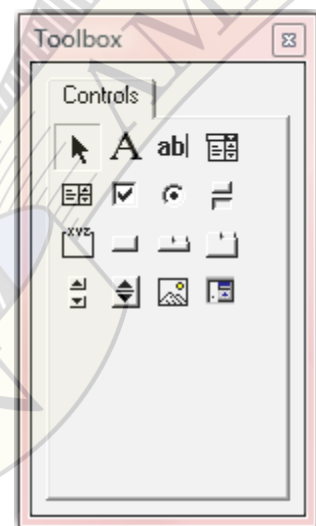
starten und stoppen. Oberhalb der UserForm befindet sich eine Leiste um das Programm auszuführen, anzuhalten oder zu stoppen.

Das Projektfenster besitzt alle Dateien, die für die Erstellung eines Programmes benötigt werden. In diesem Fenster werden drei Sorten von Dateien angezeigt:

- Die Formulare (UserForm)
- Programmdateien
- Klassendefinitionen



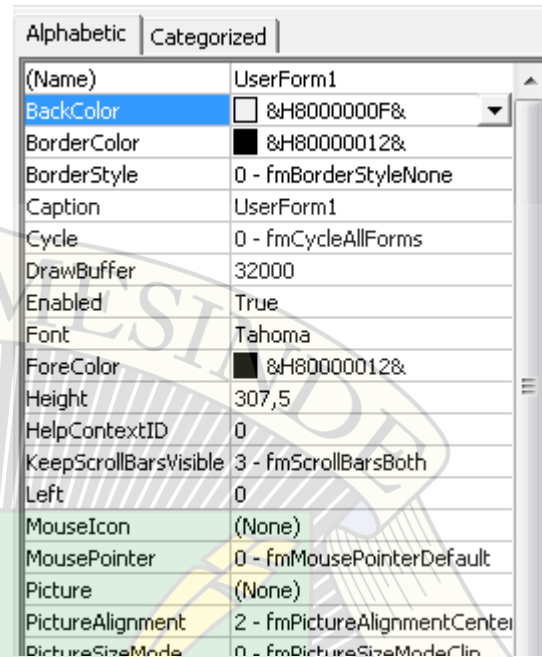
In der Toolbox werden die Steuerelemente von Visual Basic dargestellt, die auf die UserForm aufgebracht werden können; dies wären das Label, die Textbox, die Combobox, die Listbox, die CheckBox, der OptionButton, der ToggleButton, die Frame, der Commandbutton, die Tabstrip, die Mutlpage, die Scrollbar, der Spinbutton, Image und den RefEdit.



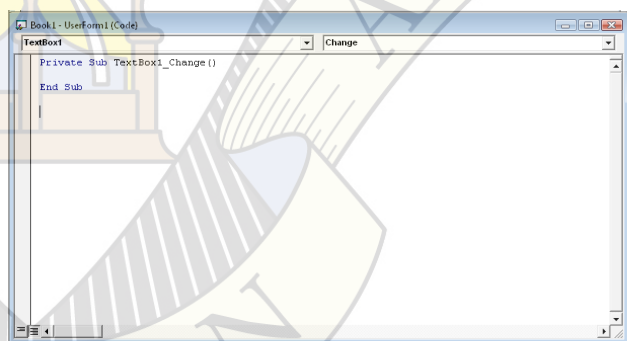
(Einige dieser Steuerelemente werde ich im Kapitel 2.3. erklären.)

Das Eigenschaftsfenster (Properties/ Property)

Das Fenster Property besitzt die Eigenschaften des ausgewählten Elements. Mit Hilfe des Property kann man die Eigenschaften von diesem Element ändern. So kann man z.B. den Namen, die Farbe, das Aussehen,... des Elements ändern. Ein Teil der Eigenschaften kann man auch während der Ausführung mit VisualBasic Befehlen geändert werden.



Das Programmcode-Fenster;
in diesem werden die
Programmcode der
einzelnen Formulare,
Steuerelemente und Dateien
erstellt und bearbeitet. Auf
diese Seite kommt man mit
einem Doppelklick auf das Element, das man programmieren
möchte.



c. Der Debugger

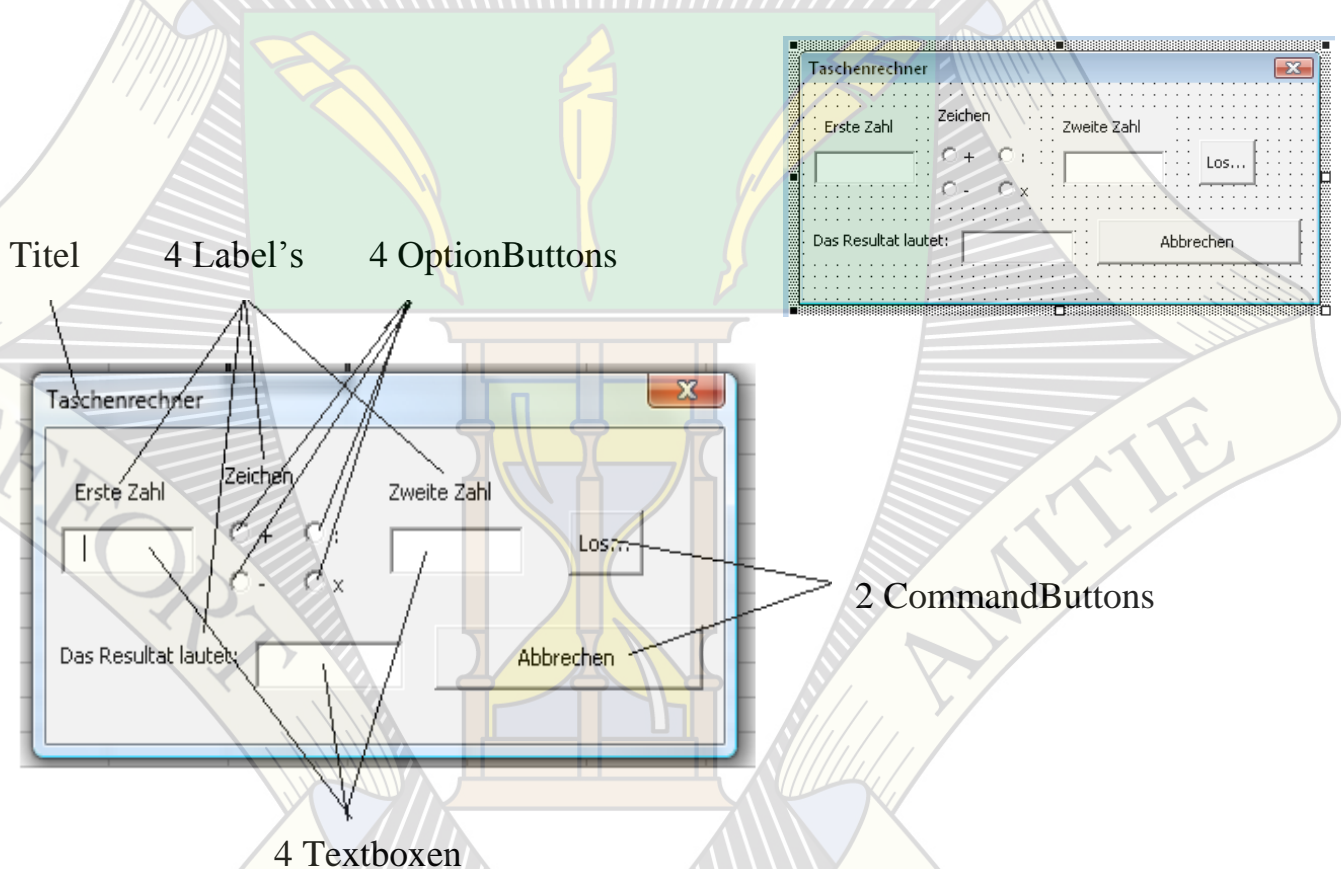
Wenn man anschließend ein fertiges Programm hat, kann man um es zu testen die F5 Taste drücken oder man klickt auf den Button in dem man das Programm geschrieben hat.

Wenn aber ein Fehler (kein Syntaxfehler, sondern bei einer JaNein-Frage; z.B. man drückt Ja aber er geht trotzdem zu Nein) im Programm ist, dann kann man diesen mit dem Debugger finden.

Um jede Anweisung des Programms während der Ausführung einzeln zu überprüfen –z.B. ob richtig gerechnet oder an der richtigen Stellen verzweigt wird–, gibt es den sogenannten Debugger. Er ermöglicht, den Inhalt einer Variablen oder den Programmablauf zu kontrollieren, um so Fehlermodule zu lokalisieren und Fehler zu finden. Der Name des Tools kommt von “debuggen”=“entmotten”. Dem ersten Anschein nach haben die Insekten nichts mit dem hier vorgestellten Tool zu tun. Dieser Begriff ist zur Zeit geprägt worden als die Computer noch nicht auf der Basis von Transistortechniken bestanden. Diese Relais schalteten über ein Magnetfeld einen mechanischen Kontakt. Wenn sich nun in so einem Kontakt eine Motte verklemmt hatte, mußte das fehlerhafte Relais gefunden und der Computer entmottet werden. Den Debugger benötigen wir heute lediglich zur Fehleranalyse des Programmablaufs.

2.3. Einige von mir geschriebene Skripte (Programme)

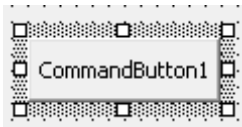
a. Ein Taschenrechner



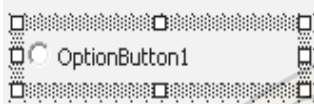
Ein Label ist dazu da, um einen Text auf ein Formular zu setzen.



Eine Textbox dient dazu, dass man während der Ausführung des Programms etwas auf das Formular schreiben kann.



Hiermit hat man die Möglichkeit auf ein Button auf zu klicken.



Wie der Name Bereits sagt: "OptionButton" dort hat man also eine Möglichkeit, dort kann man etwas auswählen

In diesem Taschenrechner ist nur in den 2 CommandButtons etwas programmiert: Im Abbrechen steht einfach nur

Unload me

Dieser Befehl bedeutet immer dass er dieses Formular schliessen soll.

Der "Los..." Button klingt kompliziert, ist aber ganz einfach. Dort drin ist folgendes programmiert:

```
If OptionButton1 Then
    TextBox1.Text = Z1 + Z2
End If
If OptionButton2 Then
    TextBox1.Text = Z1 / Z2
End If
If OptionButton3 Then
    TextBox1.Text = Z1 - Z2
End If
If OptionButton4 Then
    TextBox1.Text = Z1 * Z2
End If
```

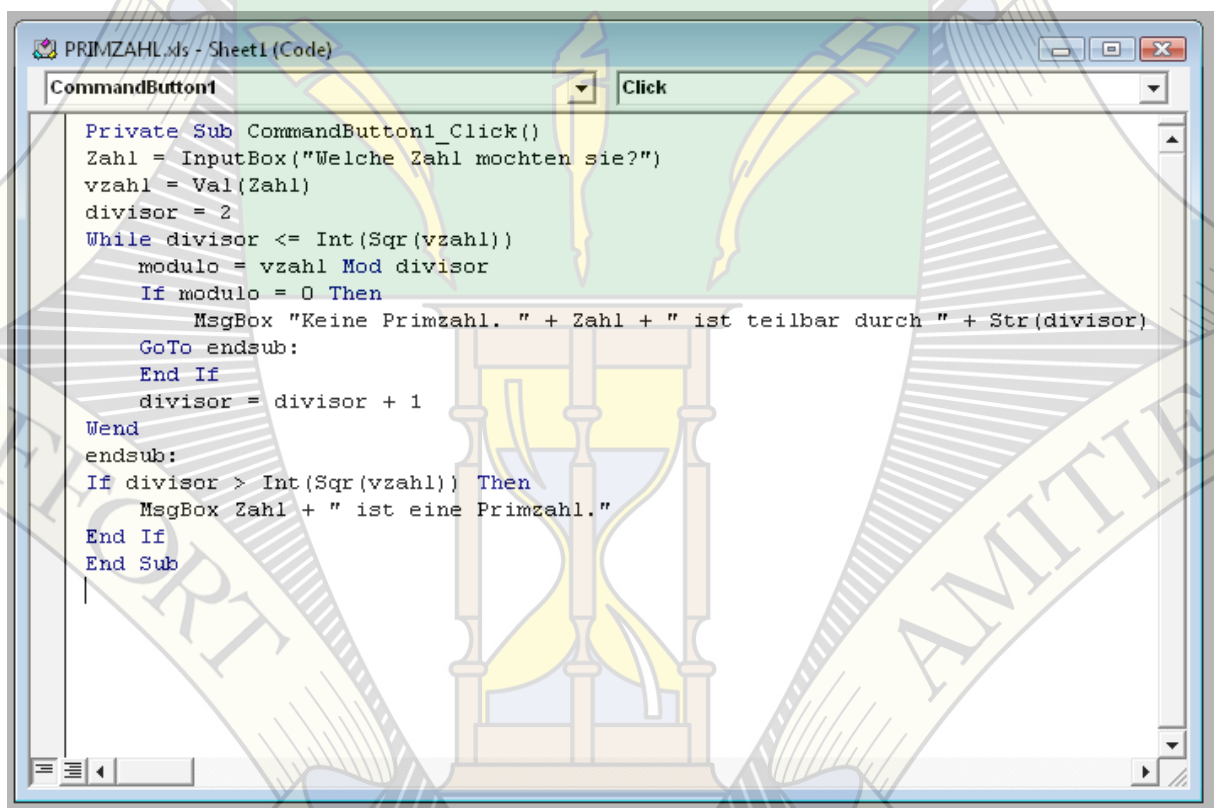
Auf deutsch würde dies heissen:“

Wenn optionbutton1 dann ‘Also das Resultat von
 Schreib in textbox1 $Z1+Z2$ ‘Z1 und Z2. Z1 und
Ende dieser Wahl ‘Z2 sind die
Wenn option button2 dann ‘Namen der 2 Textboxen.
 Schreib in textbox1 $Z1/Z2$ ‘/ ist ein Zeichen des
Ende dieser Wahl Computer ‘und bedeutet :
Wenn option button3 dann
 Schreib in textbox1 $Z1-Z2$
Ende dieser Wahl
Wenn option button4 dann ‘* ist auch ein Zeichen
Schreib in textbox1 $Z1*Z2$ ‘des Computers und
Ende dieser Wahl. ‘bedeutet “mal”



b. Primzahlen

Dieses Programm erlaubt es dem Rechner uns mitzuteilen ob eine x beliebige Zahl eine Primzahl ist oder nicht. Wenn nicht durch was sie teilbar ist.



```
Private Sub CommandButton1_Click()  
Zahl = InputBox("Welche Zahl mochten sie?")  
vzahl = Val(Zahl)  
divisor = 2  
While divisor <= Int(Sqr(vzahl))  
    modulo = vzahl Mod divisor  
    If modulo = 0 Then  
        MsgBox "Keine Primzahl. " + Zahl + " ist teilbar durch " + Str(divisor)  
        GoTo endsub:  
    End If  
    divisor = divisor + 1  
Wend  
endsub:  
If divisor > Int(Sqr(vzahl)) Then  
    MsgBox Zahl + " ist eine Primzahl."  
End If  
End Sub
```

Nun werde ich jede einzelne Zeile beschreiben:

- Privat Sub CommandButton1_Click()

Wurde bereits in Kapitel 2.2 a beschrieben.

- Zahl = InputBox ("Welche Zahl mochten sie?")

Hier wird man nach einer Zahl gefragt, die das Programm dann anschlie"ss"end ausrechnet.

- vzahl = Val(Zahl)

Wenn man dem Programm eine Zahl eingibt ist es immer noch "etwas". Hiermit verwandelt er die Variable in einen Wert.

- divisor = 2

Damit lege ich in einer Definition fest dass der Divisor 2 ist.

- While divisor <= Int (Sqr(vzahl))

...
Wend

Der while steht wegen einer Schleife dort. Der Divisor ist die vorhin definierte Variable. Der Rest bedeutet dass er die gerundete Quadratwurzel von der angegebenen Zahl ausrechnen soll. Diese Schleife läuft so lange bis der Divisor über der gerundeten Quadratwurzel ist.

- modulo = vzahl Mod divisor

modulo ist wieder eine Variable. Sie wird durch vzahl Mod divisor definiert. Dies heisst dass er den Rest der Division zwischen vzahl, dem Wert der eingegebenen Zahl, und des Divisor ergibt

- If modulo = 0 Then

Wenn der modulo 0 ist bedeutet das, dass die Rechnung auf - geht somit ist dies dann keine Primzahl. Es gibt also min einen Divisor, und zwar u.a. den der jetzt versucht wurde.

- MsgBox "Keine Primzahl. " + Zahl + " ist teilbar durch" + Str(divisor)

Also hier zeigt der Computer dass es keine Primzahl ist und durch was sie teilbar ist. Der ...Str(...) verwandelt nur den Wert zurück in eine Zahl.

- GoTo endsub:

Dies ist der alternative Ausweg aus der Schleife falls er eine Lösung gefunden hat.

- divisor = divisor + 1

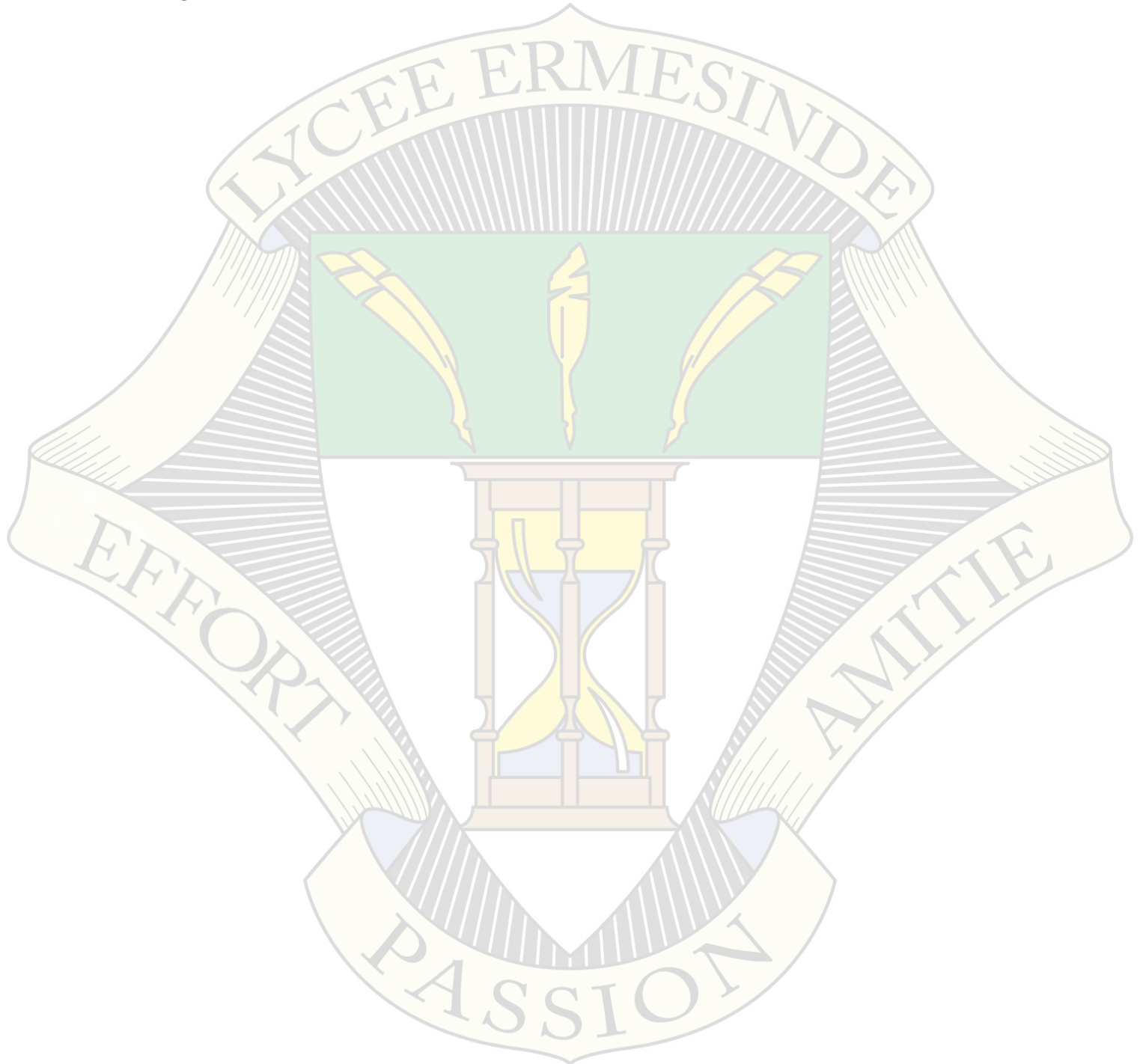
Wenn das Programm hier ankommt heißt das, dass 2 kein Divisor ist. Nun addieren wir also +1 zum derzeitigen Divisor, wird bei der ersten Runde also 3. Nun versucht er, wenn die oben genannte Kondition erfüllt ist, es nochmal, und dies bis er über der abgerundeten Quadratwurzel liegt.

- If divisor > Int(Sqr(vzahl)) Then

Sobald der Divisor über der abgerundeten Quadratwurzel liegt kommt er aus der Schleife raus und kommt zur folgender Meldebox...

- MsgBox Zahl + " ist eine Primzahl."

Dies ist die letzte Meldebox und sie gibt an, dass es sich bei der gesuchten Zahl um eine Primzahl handelt.

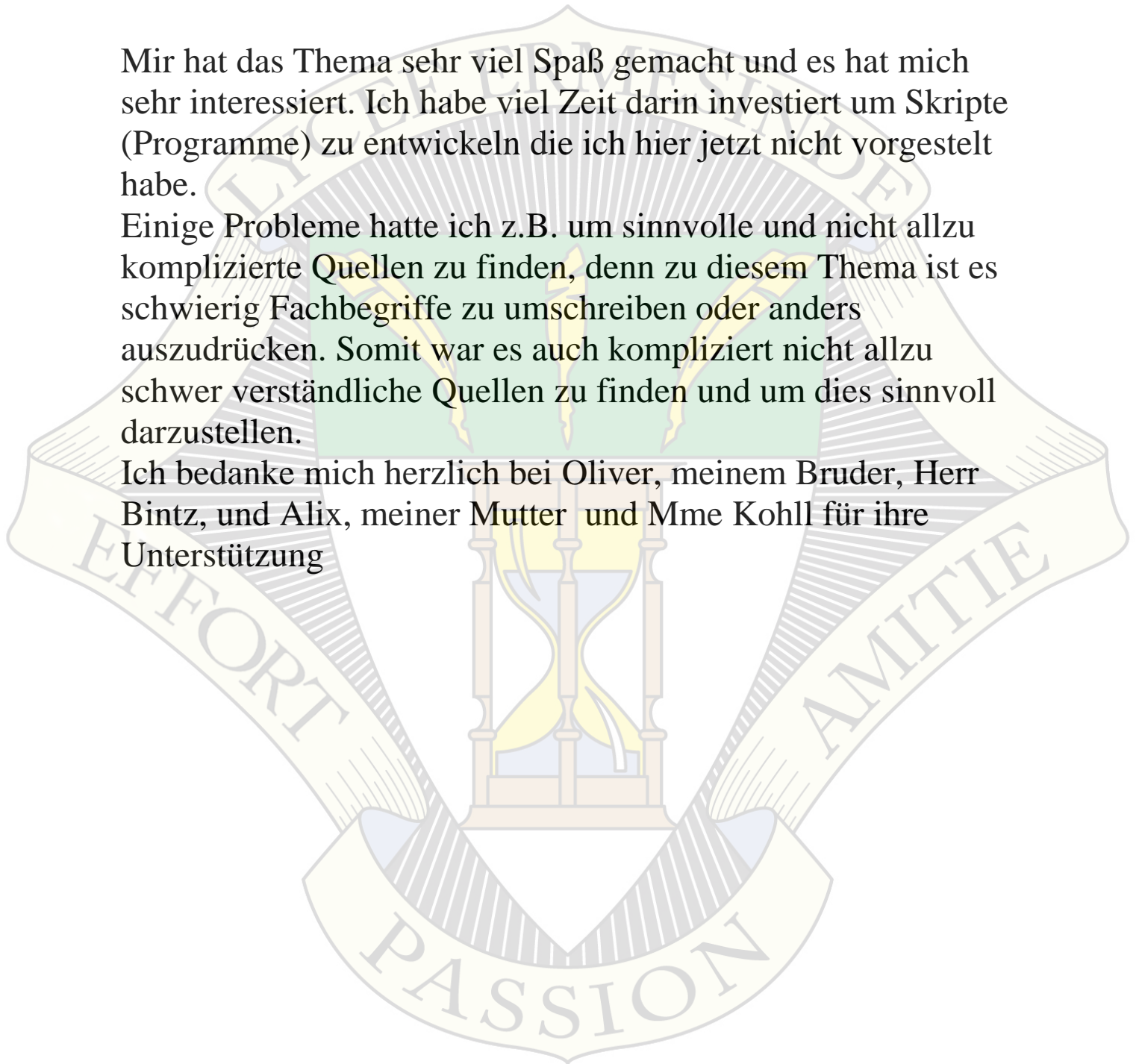


3.Schluss

Mir hat das Thema sehr viel Spaß gemacht und es hat mich sehr interessiert. Ich habe viel Zeit darin investiert um Skripte (Programme) zu entwickeln die ich hier jetzt nicht vorgestellt habe.

Einige Probleme hatte ich z.B. um sinnvolle und nicht allzu komplizierte Quellen zu finden, denn zu diesem Thema ist es schwierig Fachbegriffe zu umschreiben oder anders auszudrücken. Somit war es auch kompliziert nicht allzu schwer verständliche Quellen zu finden und um dies sinnvoll darzustellen.

Ich bedanke mich herzlich bei Oliver, meinem Bruder, Herr Bintz, und Alix, meiner Mutter und Mme Kohll für ihre Unterstützung



4.Quellenverzeichnis:

<http://books.google.lu/books?id=tVEmTLh4II8C&pg=PA23&lpg=PA23&dq=visual+basic+lernen&source=bl&ots=9CDVpfYV6w&sig=cH32u5JtFPnLYs60A9J44PLV3Ww&hl=de&sa=X&ei=nmJrU4CVAZTN7AaArYHQAQ&ved=0CGMQ6AEwCA#v=onepage&q&f=true>

<http://www.vb-fun.de/vb/index.htm> <http://www.vb-fun.de/vb/index.htm>

http://de.wikibooks.org/wiki/Visual_Basic_6:_Schleifen

