
Future
Mario
von Steve Göring und Daniel Renner
Projektarbeit im Fach Angewandte Technik
stg7@gmx.de,Daniel26289@yahoo.de © 2008
Intro abbrechen

**PROJEKTARBEIT**

IM FACH

ANGEWANDTE TECHNIK

Abgabe: 02.04.2008

vorgelegt von
Steve Göring und Daniel Renner

Inhaltsverzeichnis:

1.1 Aufgabenstellung ..... 5
1.2 Einschränkungen ..... 5
1.3 Hilfsmittel ..... 6
2. Problemlösung: ..... 9
2.1 Allgemein ..... 9
2.2 Lösungsansätze ..... 10
2.2.1 Musik abspielen ..... 10
2.2.2 OpenGl „anwerfen“ ..... 12
2.2.3 Der Kollisionsalgorithmus ..... 13
2.2.4 Level Editor ..... 17
2.2.5 Anpassungsmöglichkeiten (Optionen) ..... 20
2.2.6 Uml Diagrammersteller ..... 23
2.3 UML Diagramme der verwendeten Klassen ..... 24
2.3.1 „Root“ ..... 24
2.3.2 „Objekte“ ..... 27
2.3.3 „OpenGL“ – TFOpenGLWindow in der Unit UFOpenGLWindow ..... 30
2.3.4 „Optionen“ ..... 31
2.3.5 „Tools\LevelEditor“ ..... 32
2.3.6 „USound“- TSound in der Unit Usound ..... 33
2.4 Beziehungen zwischen den Objekten /Klassen ..... 34
2.4.1 Ist Beziehungen in Future Mario ..... 34
2.4.2 Hat und Kennt Beziehungen in Future Mario ..... 36
2.5 Struktogramme ..... 37
2.5.1 Kollisionsalgorithmus ..... 37
2.5.2 Laden der Karte ..... 38
3 Quelltexte ..... 39
3.1 Future Mario ..... 39
3.1.1 UFOptionen.pas ..... 39
3.1.2 UFunktionen.pas ..... 41
3.1.3 UHaupt.pas ..... 42
3.1.4 UHilfe.pas ..... 46
3.1.5 UImageButton.pas ..... 47
3.1.6 UInfo.pas ..... 48

3.1.7 UIntro.pas ..... 49
3.1.8 UMenu.pas ..... 51
3.1.9 UMusikImSpiel.pas ..... 53
3.1.10 USpezOp.pas ..... 57
3.1.11 UTypenUndKonstanten.pas ..... 62
3.1.12 Objekte\UAnzeige.pas ..... 63
3.1.13 Objekte\UFigur.pas ..... 65
3.1.14 Objekte\UGegenstand.pas ..... 69
3.1.15 Objekte\UGegner.pas ..... 72
3.1.16 Objekte\UGLObject.pas ..... 74
3.1.17 Objekte\UListe.pas ..... 75
3.1.18 Objekte\UMap.pas ..... 76
3.1.19 Objekte\UMuenze.pas ..... 77
3.1.20 Objekte\USpieler.pas ..... 79
3.1.21 OpenGl\UOpenGlWindow.pas ..... 85
3.1.22 Optionen\UOptionen.pas ..... 87
3.1.23 Optionen\UOptionenGrafik.pas ..... 89
3.1.24 Optionen\UOptionenSound.pas ..... 90
3.1.25 USound\usound.pas ..... 92
3.2 LevelEditor ..... 96
3.2.1 Tools\Levelleditor\UHaupt.pas ..... 96
3.2.2 Tools\Levelleditor\UTileMap.pas ..... 100
3.2.3 Tools\Levelleditor\Utypen.pas ..... 100
4. Testläufe ..... 101

# 1. Problembeschreibung:

## 1.1 Aufgabenstellung

Zu einer selbstgewählten, anspruchsvollen Aufgabenstellung aus einem beliebigen Anwendungsbereich sollte im Fach angewandte Technik ein Delphi-Projekt entwickelt werden, welches Objekte selbstentwickelter Klassen über ereignisgesteuerte Benutzeroberflächen verarbeitet.

Dabei ist folgendes zu Beachten:
Weitere Delphi-Komponenten sollen selbstständig erarbeitet werden. Es sollen mindestens drei eigene Klassen entworfen werden, deren Objekte über definierte Hat-, Ist- und Kennt-Beziehungen untereinander kommunizieren, Aufträge erteilen oder Anfragen stellen.

Als Aufgabe haben wir uns für die Entwicklung eines Super Mario Clones entschieden.
Unser Super Mario Clone bekam den Namen „Future Mario“.

## 1.2 Einschränkungen

Bei der Abgabeversion gibt es keine Highscore, außerdem gibt es momentan nur ein Level und es gibt noch keine Lösung um einen Mapwechsel im Spiel durchzuführen, der Leveleditor kann aber bereits Mapwechsel in einer Map einbauen.
In „Future Mario“ gibt es keine der bekannten Mario typischen Objekte, wie Pilze, Federn, oder Sterne, es gibt nur Münzen und Gegner. Theoretisch wäre eine Erweiterung der Objekte im Spiel problemlos möglich, aber wir haben aus Zeitmangel darauf verzichtet (auch um den Umfang des Projektes nicht weiter zu vergrößern)

## 1.3 Hilfsmittel

Wir verwendeten oft die Online Hilfe von Delphi.
Das Projekt wurde zum größten Teil mit Turbo Delphi 2006 erstellt, mit dieser Delphi Version war es möglich schnell und unkompliziert das Projekt zu verwalten.

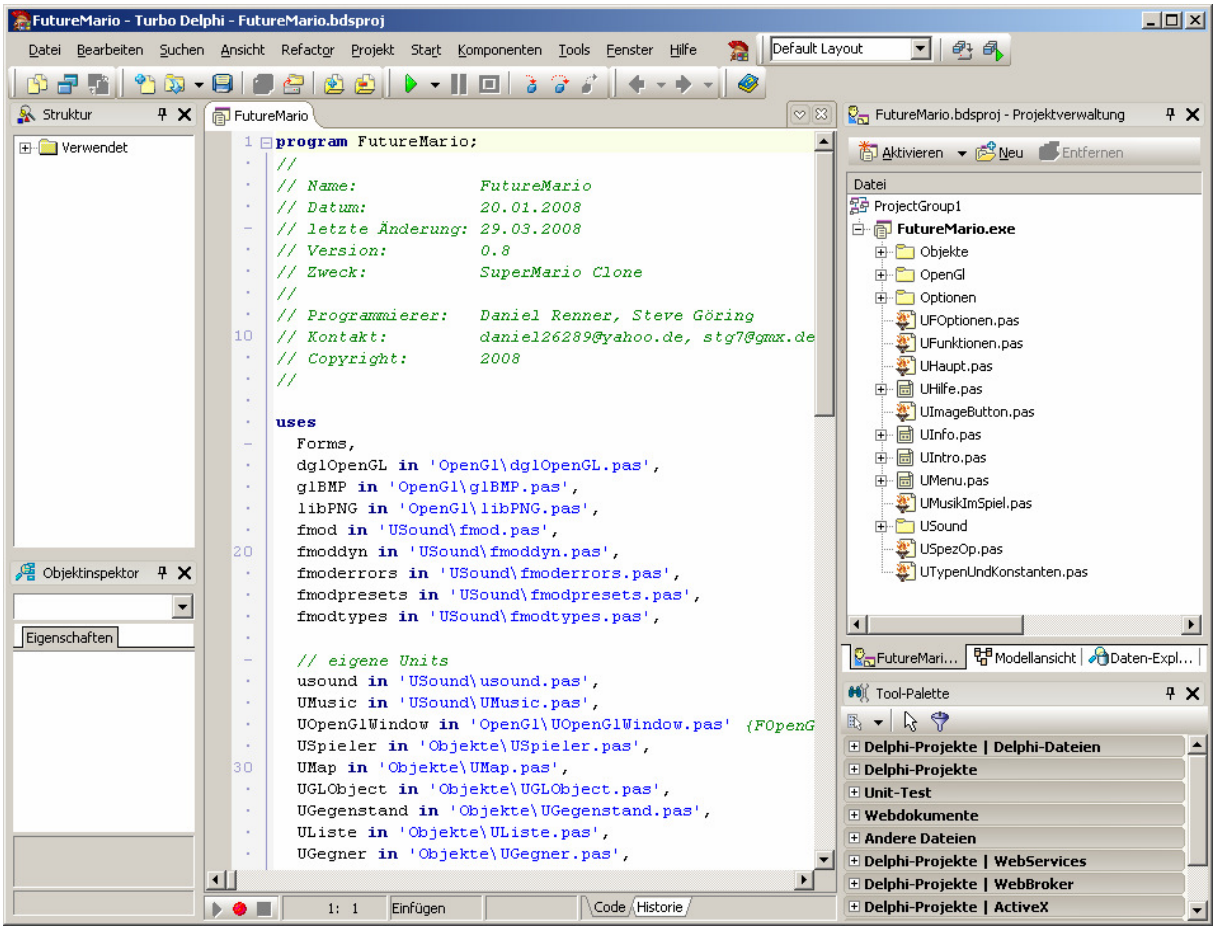



**Abbildung 1Turbo Delphi 2006**

Auf der rechten Seite sieht man das Projektverzeichnis mit allen verwendeten Units, man kann dadurch schnell erkennen welche Units verwendet werden und sie auch schnell bearbeiten. Das Programmieren wurde dadurch übersichtlicher. Es stellte sich aber auch heraus, dass das Projekt mit Delphi 7 kompatibel ist und problemlos mit dieser Version kompiliert werden sollte.

Unsere Informationsquelle war außerdem noch das Internet:
z.B.
Für OpenGl:
http://wiki.delphigl.com/index.php/Tutorial OpenGl Tutorials

Informationen zu INI Files:
http://www.delphi-treff.de/no_cache/tutorials/datenspeicherung/win32/ini-dateien/page/3/

Auf dieser Seite erhielten wir auch Informationen über „selbstdefinierte Events“ und Properties, diese Grundlagen nutzten wir dann auch in unserem Programm (z.B. http://www.delphi-treff.de/sprachen/object-pascal/funktions-und-methodenzeiger/ selbstdefinierte Events)

Um die Sound DLL FMOD nutzen zu können, beschäftigten wir uns mit der sehr guten Englischen Dokumentation

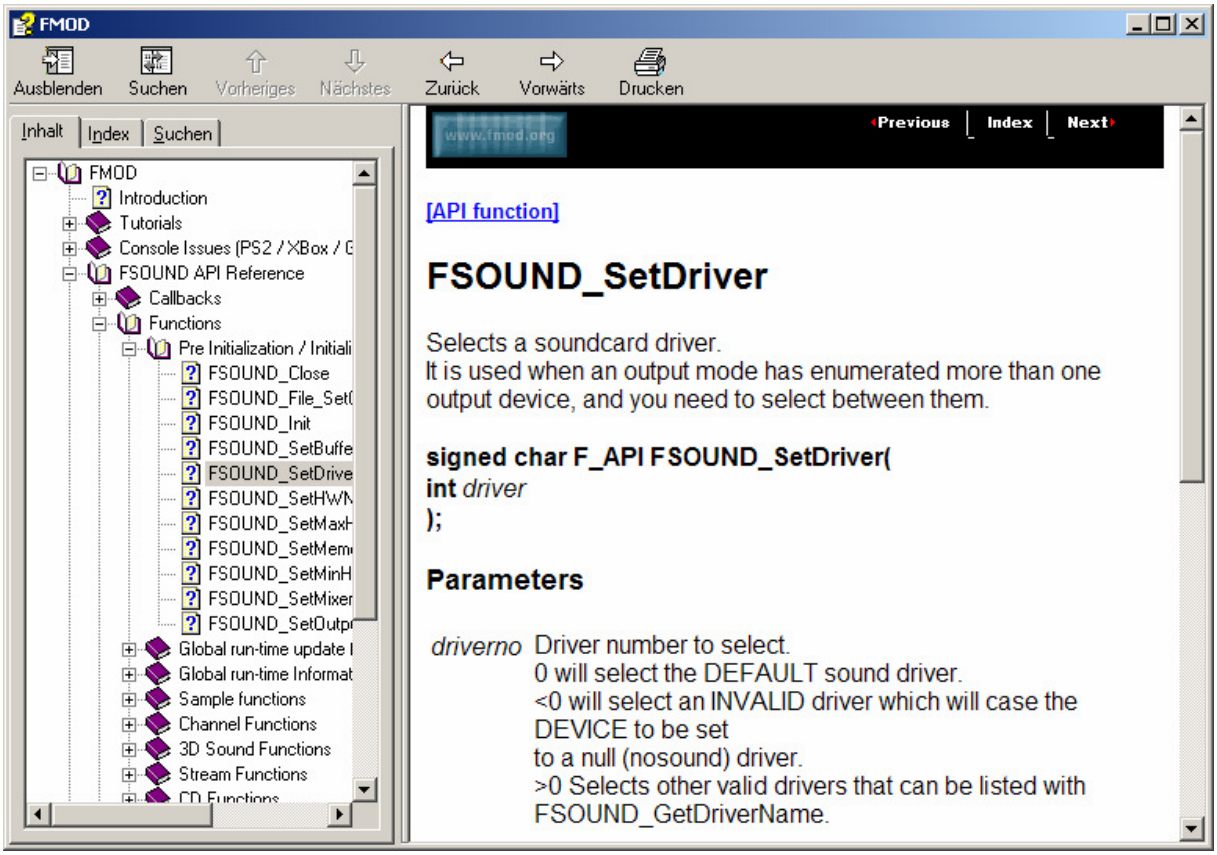



**Abbildung 2 Fmod Hilfe**

Durch die Dokumentation und einige Beispielprogramme (mit verschiedenen Programmiersprachen, darunter auch Delphi) konnten wir uns schnell in die Arbeit mit dieser Bibliothek einarbeiten.

Um Zugriffe auf spezielle Funktionen zu erlangen (OpenGl, Sound) benutzten wir Fremdunits.
Diese Units sind als solche kenntlich gemacht und stammen aus unterschiedlichen Quellen, welche bei einigen auch angegeben sind.

Fremdunits:

Aus dem DGL Paket (http://www.delphigl.com/)
- dglOpenGL
- glBMP
- libPNG

Aus dem Fmod SDK Paket
- fmod
- fmoddyn
- fmoderrors
- fmodpresets
- fmodtypes

## 2. Problemlösung:

### 2.1 Allgemein

Bereits am Anfang des Projektes stand für uns fest, dass wir ein Supermario Spiel programmieren wollen, welches mit OpenGL arbeiten soll. Außerdem sollte dieses Spiel FMOD benutzen, denn mit einer Mediakomponente ist es nicht möglich parallel mehrere Sounds abzuspielen, wir wollten eine eigene Klasse dafür entwickeln.
Unsere Wahl für die Grafik fiel auf OpenGL, obwohl es hauptsächlich für 3D Anwendungen entwickelt wurde.
Es ist ganz einfach zu erklären, wenn man auf die Canvas zeichnet hat man irgendwann bei vielen Objekten Geschwindigkeitsproblemen, oder Koordinatentransformation und Auflösungsunabhängigkeit sind hier nur schwer zu realisieren. OpenGL stellte dagegen eine gute Alternative dar, denn da es eigendlich für 3D Anwendungen ist, konnten wir uns unter 2D Anwendungen einen erheblichen Geschwindigkeitsvorteil sichern, außerdem bietet es auch Auflösungsunabhängigkeit.
An einem Computer hatten wir sogar eine Framerate von knapp 1000 Frames pro Sekunde (in der früheren Phase standen die Frames in der Titelleiste des OpenGl Fensters)

## 2.2 Lösungsansätze

### 2.2.1 Musik abspielen

Sounds abspielen war ein zentraler Gedanke bei der Problemlösung. Dabei modellierten wir das gesamte Soundsystem in einer kleinen Anwendung, und führten diese dann mit dem OpenGl Teil später zusammen. Unser Soundsystem besteht dabei aus 3 Klassen.

Hier sieht man nun das Testprogramm:

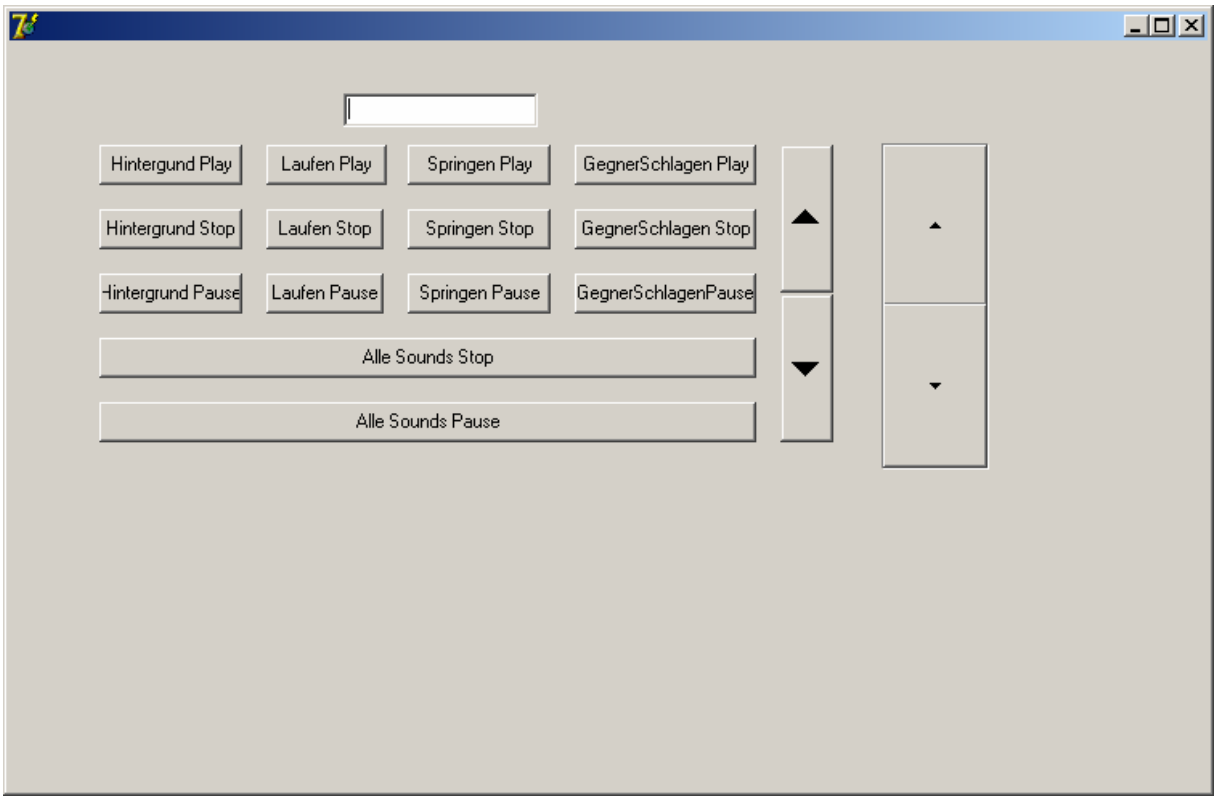



**Abbildung 3 Musikumgebung**

Um eventuelle Fehler abzufangen werden selbstdefinierte Events benutz.

In diesem Formular kann man bereits die gewünschte Funktionalität sehen.

Das Soundsystem soll verschiedene Musikdaten verwalten, sie mit Namen ansprechen können, d.h. auch parallel abspielen. Außerdem war es uns wichtig die Sounddaten variable anpassen zu können, ein schnelles wechseln der Sounddaten mit anderen Namen und Format musste möglich sein- ohne den Quelltext zu

ändern. Für diese Funktionalität setzten wir auch eine Ini Datei, die sich im Verzeichnis „cfg“ befindet und den Namen „config.ini“ trägt.

Sie hat folgenden Aufbau:

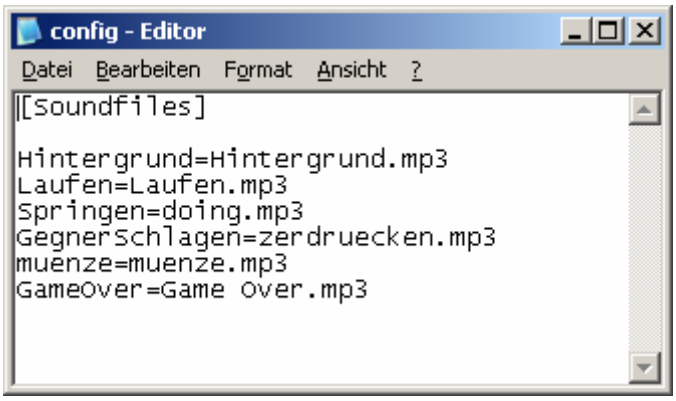



**Abbildung 4 Sound Konfigurationsdatei**

Soundname=sounddatei

Mit Hilfe des Namens kann man in einer Instanz der Klasse UMusikImSpiel genau dieses Soundfile abspielen.

### 2.2.2 OpenGl „anwerfen“

In einem anderen Miniprogramm experimentierten wir mit OpenGL .

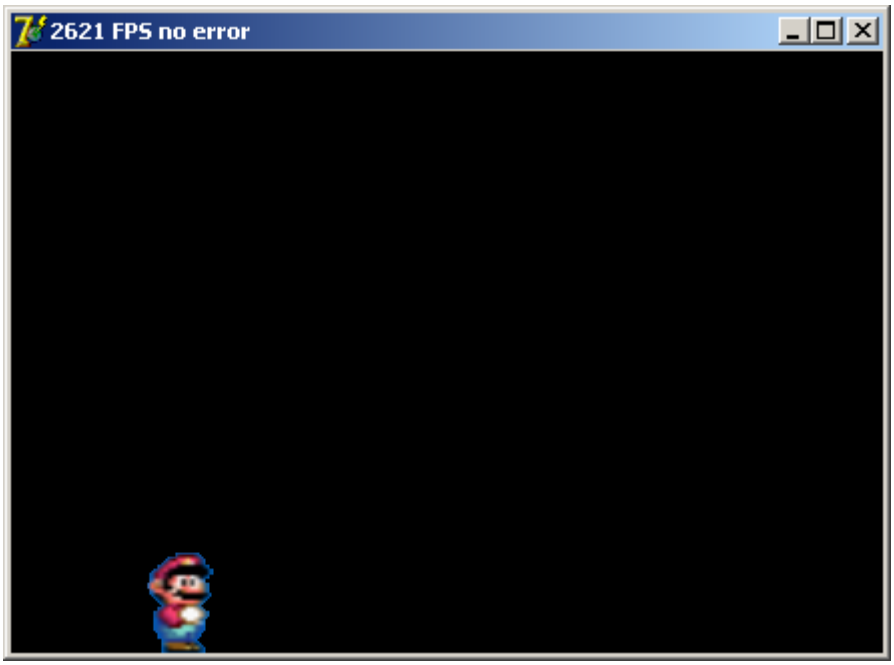



**Abbildung 5 OpenGL Entwicklung**

Mario konnte sich bereits bewegen und die Texturen wurden angepasst.
Bis Mario springen, laufen, und Gegner töten konnte verging noch ein wenig Zeit.
Um Gegner zu schlagen muss Mario feststellen können ob er mit einem Gegner zusammengestoßen ist. (auch Kollision genannt)

### 2.2.3 Der Kollisionsalgorithmus

In unserem Mario Spiel treten ständig Kollisionen auf, z.B. Mario stößt mit dem Boden zusammen, mit Gegnern, Münzen oder er stürzt ins Leere.
Aus den genannten Gründen war es wichtig einen „ordentlichen“ Kollisionsalgo zu erstellen.

Da dieser Algorithmus nicht der einfachste werden sollte und wir die Schwierigkeiten bereist erkannt hatten, starteten wir ein anderes Testprogramm um den Algorithmus auszutesten und zu entwerfen.

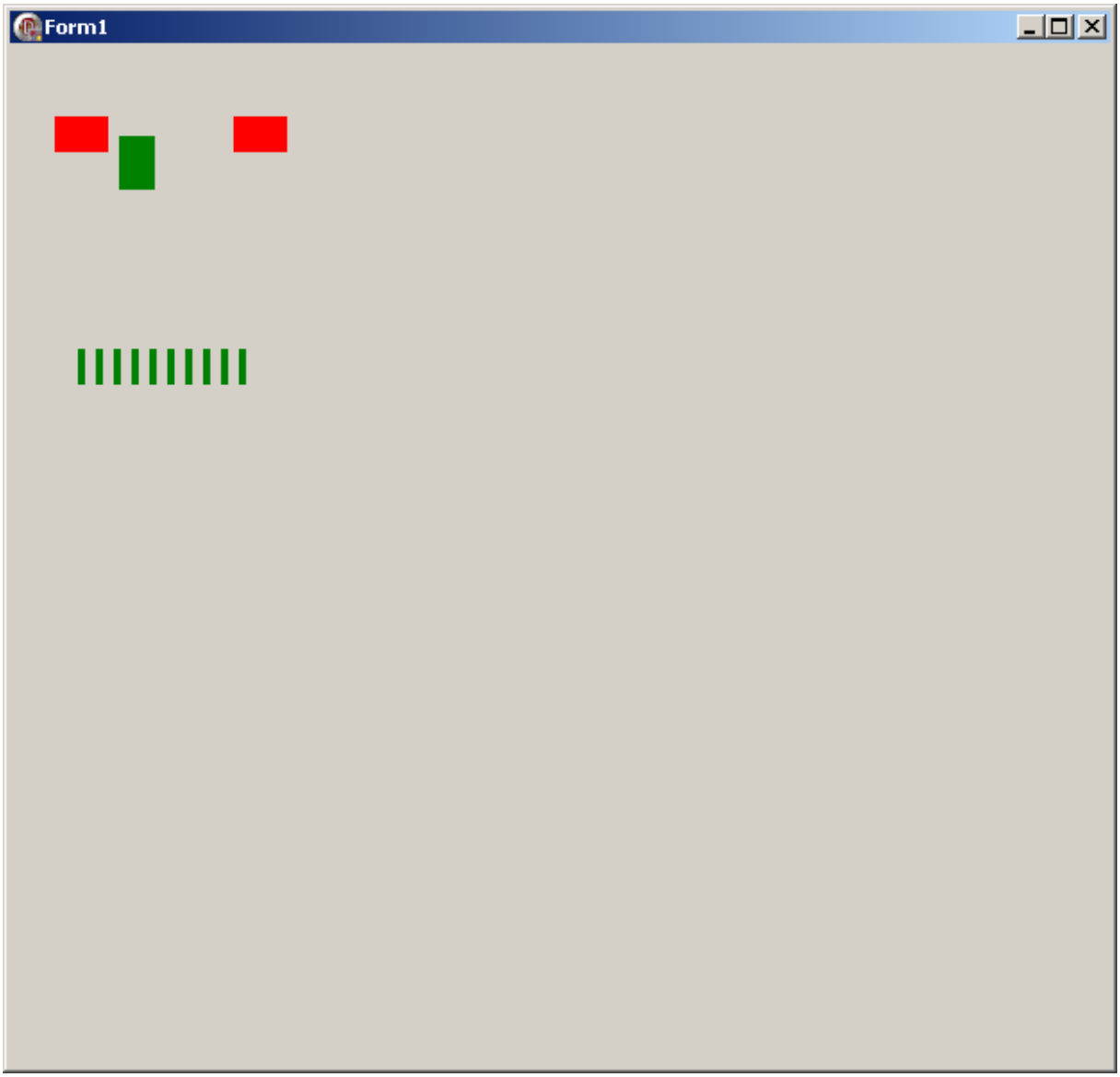



**Abbildung 6 Kollisionstestumgebung**

Das große grüne Rechteck kann man mit „W/A/S/D“ bewegen, sobald es dann auf ein Hindernis (andere Rechtecke) stößt kann man es nicht weiterbewegen.

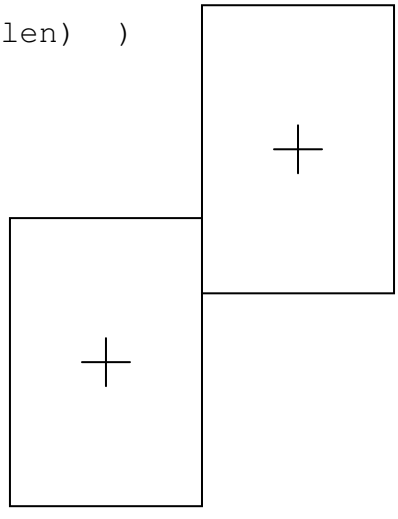
Wir probierten einige mögliche Algorithmen aus und mussten bald feststellen, dass es eben nicht so einfach war, wie wir dachten.

Die einfachste Idee war:

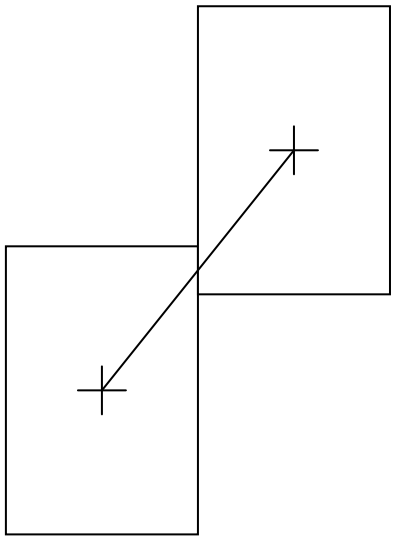
Wenn man die Rechtecke als Punktmengen auffassen würde, dann könnte man mit Hilfe der Durchschnittsmenge sofort bestimmen ob eine Kollision vorliegt. (wenn die Durchschnittsmenge ungleich der leeren Menge ist, dann erfolgte eine Kollision)

Dieser Algorithmus erwies sich aber als schlecht umsetzbar. Nach weiteren Überlegungen kamen wir auf die folgende und auch endgültige Idee.

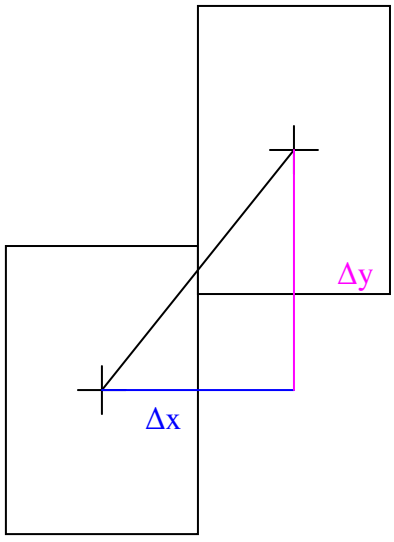
Wenn zwei Rechtecke kurz vor der Kollision stehen, also Seite an Seite:
(Hinweis, wir zeichnen die Rechtecke immer von ihrem Schwerpunkt ausgehend
(= Mittelpunkt der Diagonalen) )



Anschließend verbindet man beide Schwerpunkte:



Wenn man nun die Differenzen der X und Y Werte als Strecken einzeichnet erhält man:

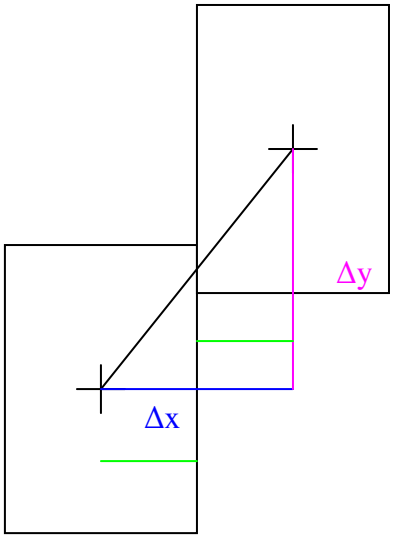



Die Δx und Δy kann man aus der Differenz der Koordinaten berechnen

Die nächste Skizze veranschaulicht den Algo komplett:

Grün eingezeichnet sind die halben Rechteckbreiten.
Man kann erkennen, wenn $\Delta x = b_1/2 + b_2/2$ , dann kommt es zu einer x Kollision
( Summer der halben Breiten <= delta x)

Gleiches gilt auch für eine y Kollision
(Summer der halben Höhen <= delta y)




Es tritt also eine Rechteck – Rechteck Kollision auf, wenn die Summe der halben Breiten kleiner oder gleich der X Koordinaten Differenz und
die Summe der halben Höhen klein der der Y –Differenz sind.

Damit wir nun Kollisionen mit mehreren Objekten haben können, organisierten wir alle Objekte in verschiedenen Listen.

### 2.2.4 Level Editor

Um mehrere mögliche Karten zu erstellen und außerdem dynamische Karten zu laden musste unser Projekt über eine Mapdatei verfügen. Da solche Dateien eher kryptisch sind und schlecht manuell erzeugt werden können, musste ein MapEditor erstellt werden.

#### 2.2.4.1 Beschreibung des Map Formates

Eine Map besteht aus mehreren Zeilen in denen jeweils die Koordinaten und der Typ eines Objektes gespeichert sind.
(Aufbau einer Zeile :

x,y , typ

Dabei werden 3 Objekttypen unterschieden
Gegenstandsobjekte (allgemeine Hindernisse)
„1,10,11 6“ dies ist ein Gegenstandsobjekt mit den Texturkoordinaten 11,6 und den Mapkoordinaten 1,10
Münzobjekte:
„11,7,münze“ beschreibt eine Münze
Gegnerobjekte:
„28,8,gegner“ Gegner befindet sich in Spalte 28 , Zeile 8

Es werden also keine Leerfelder gespeichert, da dies auch Speicherplatzverschwendung wäre.
Unser Levelformat hat die Dateiendung „.mp“ für Map

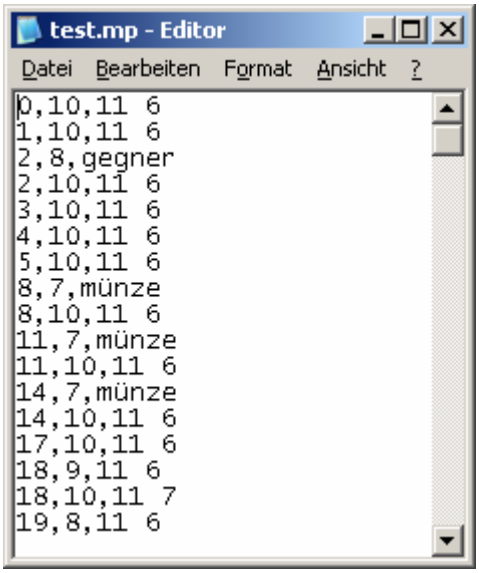



**Abbildung 7 Beispielmap**

### 2.2.4.2 Funktionalität

Der entwickelte Levelleditor besitzt keinerlei große grafische Ausgabe. Die möglichen Feldarten bzw. Objektarten werden durch verschiedene Farben kenntlich gemacht
Zur Darstellung benutzen wir eine DrawGrind.

Man kann bereits erstellte Level öffnen, oder auch ein momentanes Speichern.
Wichtig war es, das man für die Hindernisse die Texturkoordinaten aus der Tilemap erstellen kann (gelöst mit einem Unterformular).

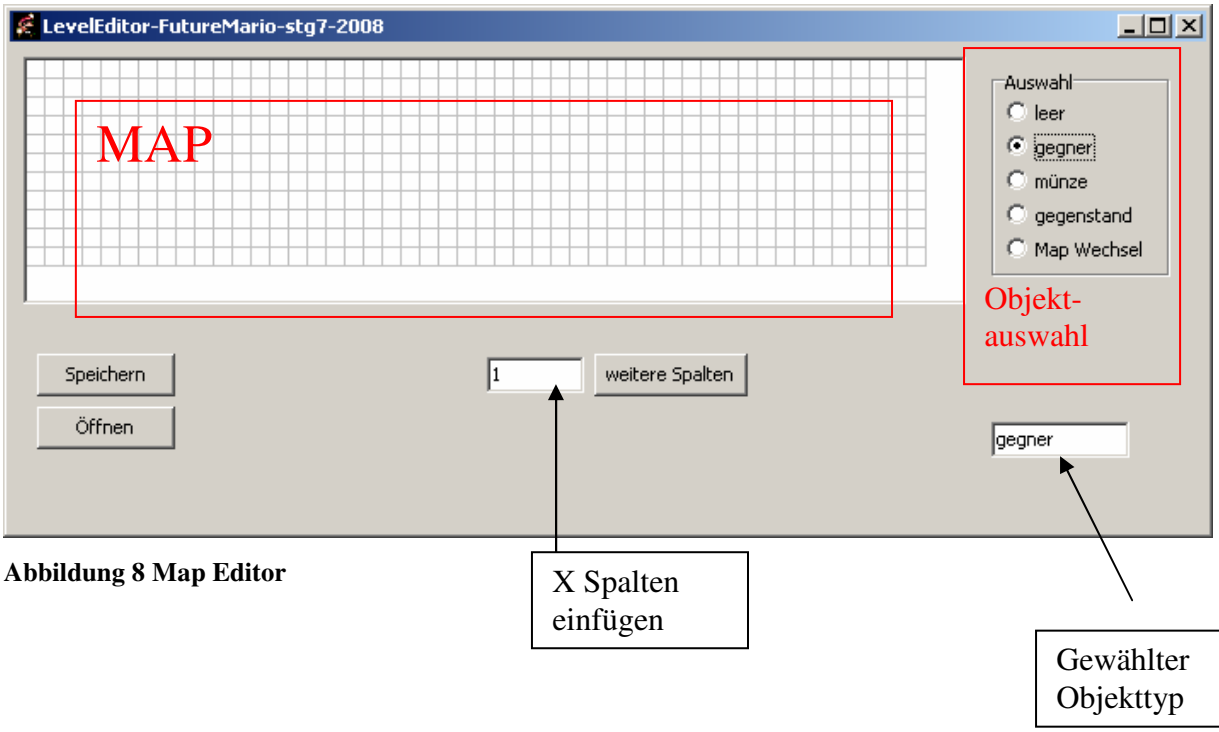



**Abbildung 8 Map Editor**

Eine geladene Map sieht man auf dem folgenden Bild

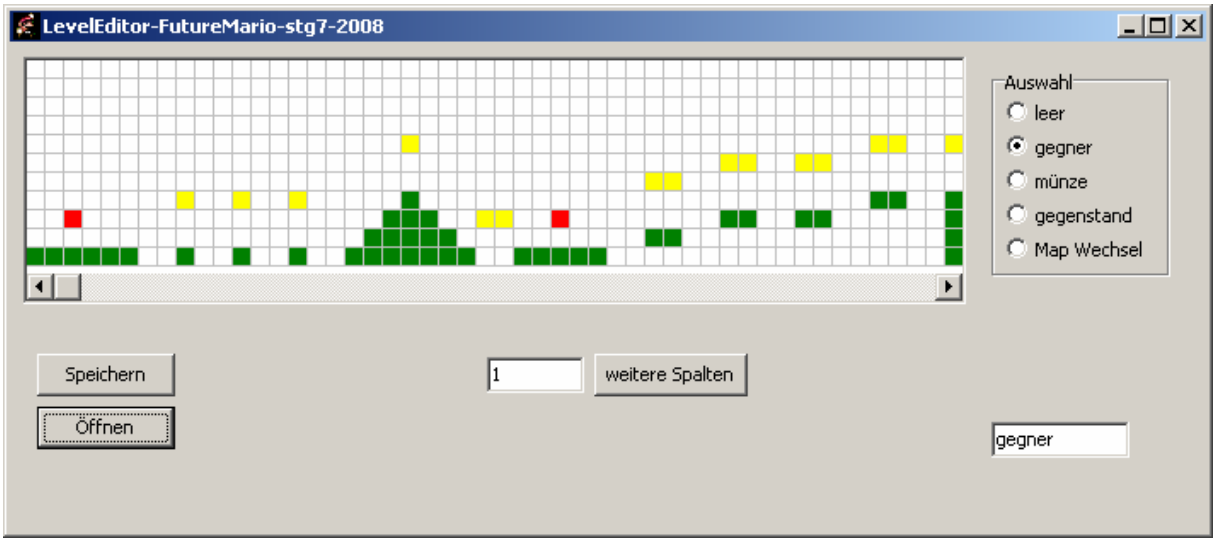



**Abbildung 9 geladene Map**

Rot = Gegner , Grün = Hindernis , Gelb= Münze

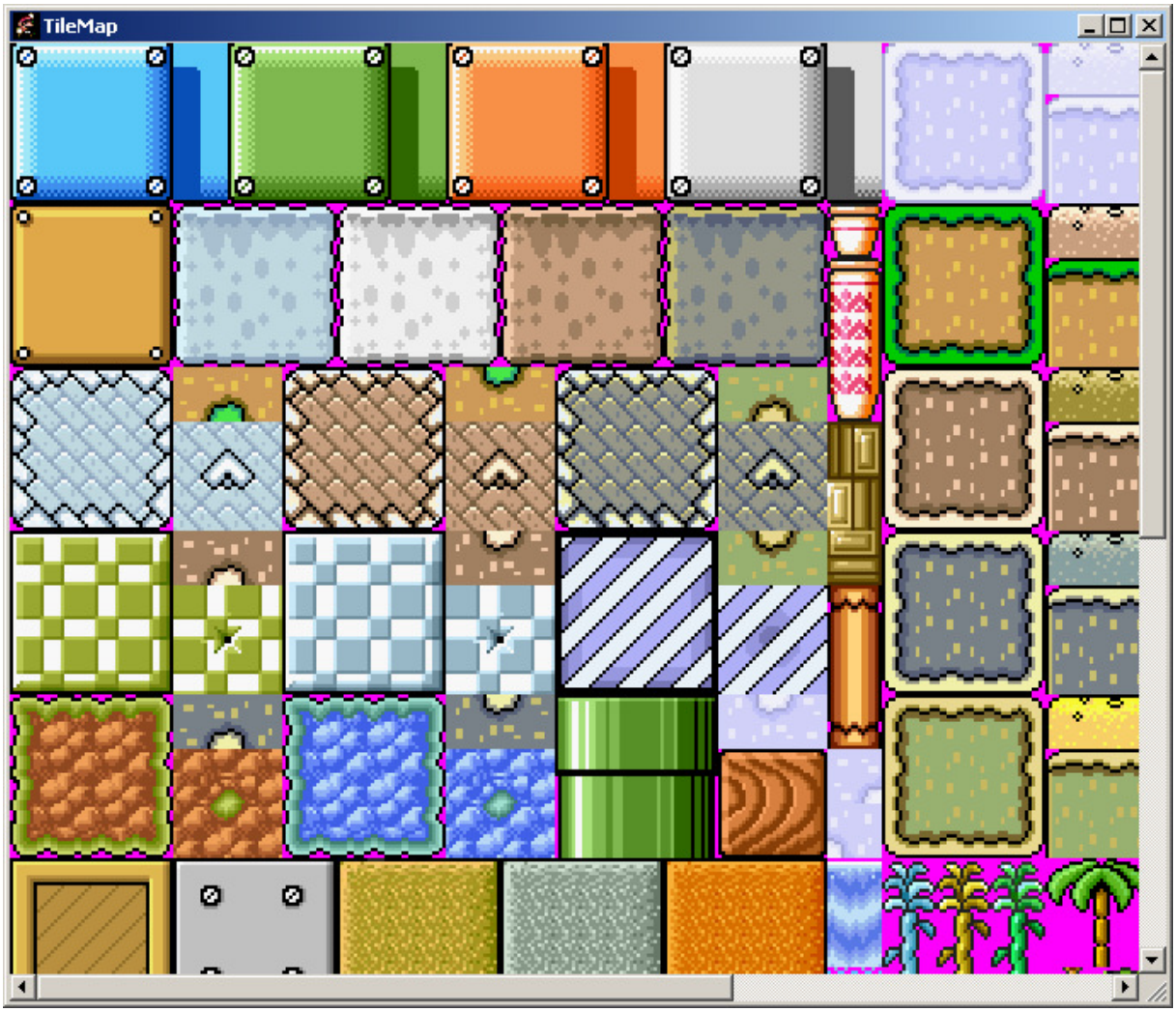



**Abbildung 10 Tielbild auswahl bei Gegenstand**

## 2.2.5 Anpassungsmöglichkeiten (Optionen)

Das Spiel sollte auch über einen Optionsdialog verfügen, dabei können unterschiedliche Einstellungen getroffen werden, die in einer Ini Datei abgelegt werden.

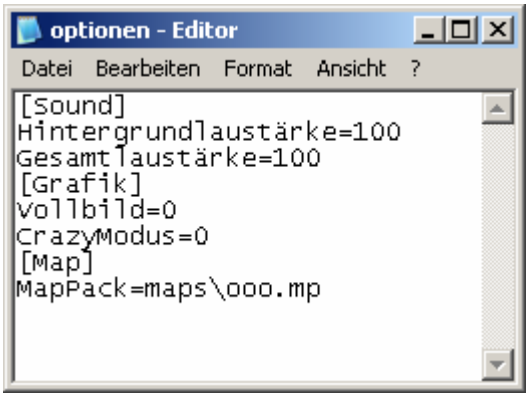
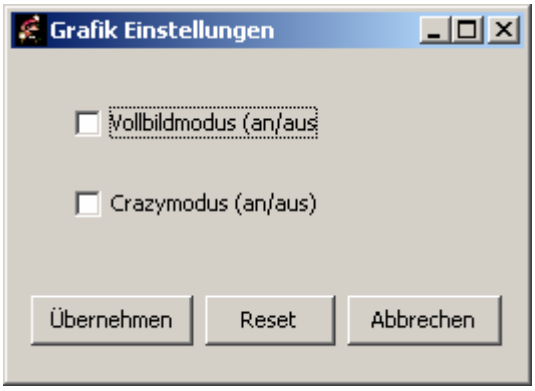



**Abbildung 11 Optionen.ini im Ordner "cfg"**

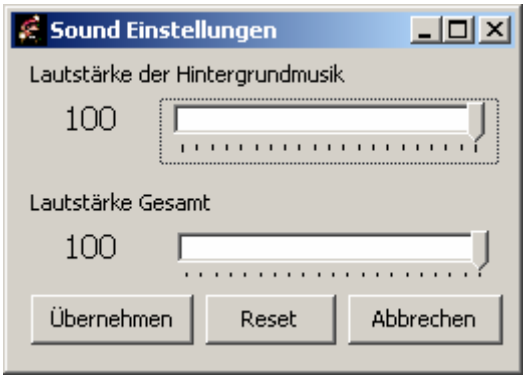
Im Spiel gibt es die Möglichkeit 2 mögliche Einstellungen vorzunehmen, einmal Grafik und dann noch Soundeinstellungen

### 2.2.5.1 Crazy Modus / Vollbild (Grafik)




Die Option Vollbild ist selbsterklärend. Doch was versteht man unter dem Crazymodus.
Dieser spezielle Modus erzeugt ein verrückteres Spielgefühl, denn alle Steine sind bunt eingefärbt und der Hintergrund ist schwarz.

### 2.2.5.2 Soundeinstellungen




Hintergrundmusik und die Gesamtlautstärke können verändert werden.

#### 2.2.5.3 Mappackwahl

In der „Optionen.ini“ befindet sich auch eine Festlegung für das Mappacket, dies stellt die Startmap dar, und kann bei bedarf in der Ini Datei verändert werden.

### 2.2.6 Uml Diagrammersteller

Um all die ganzen UML Diagramme zu erzeugen, programmierten wir ein kleines Tool, welches aus einer Delphie Klassendeklaration ein UML Diagramm in Form eines Formulares erzeugt:

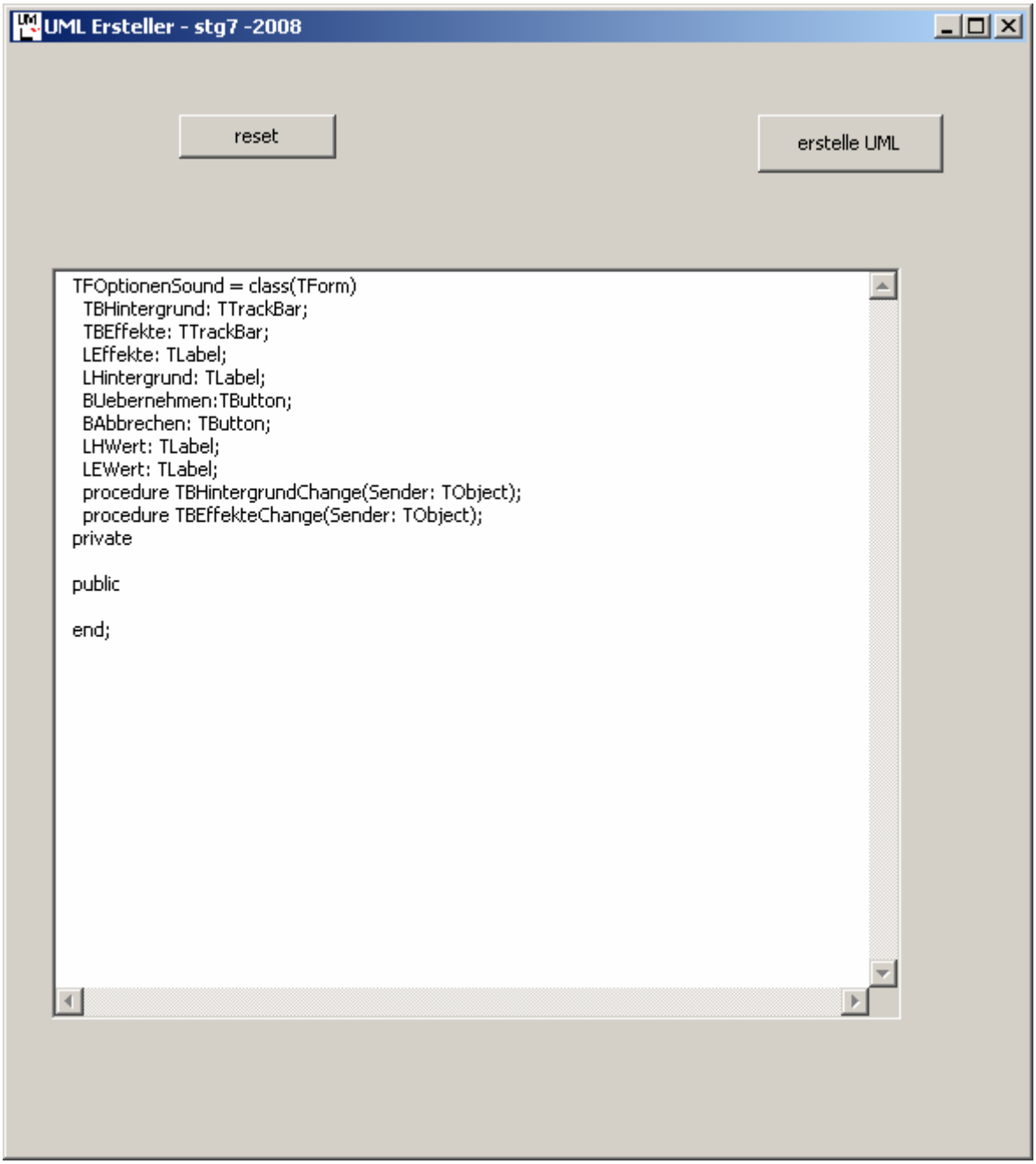


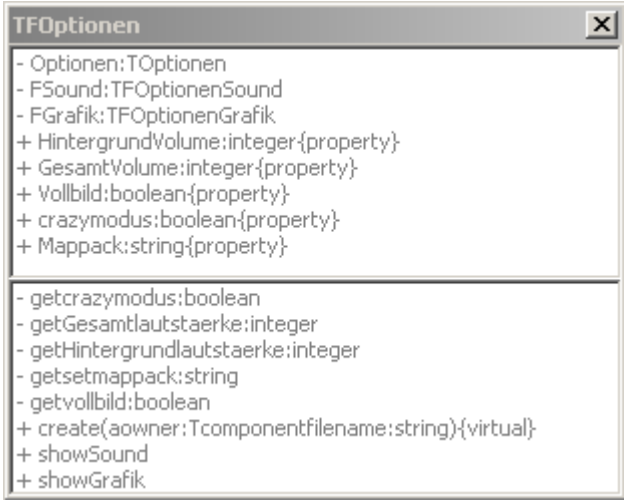

**Abbildung 12 UML Tool im Ordner Tools**

## 2.3 UML Diagramme der verwendeten Klassen

Die UML Diagramme werden nach der Ordnerstruktur des Projektes aufgelistet


### 2.3.1 „Root"

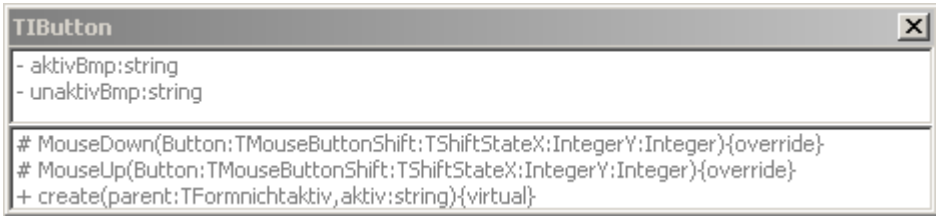
#### 2.3.1.1 TFOptionen in der Unit UFOptionen



### 2.3.1.2 TFSuperM in der Unit UHaupt

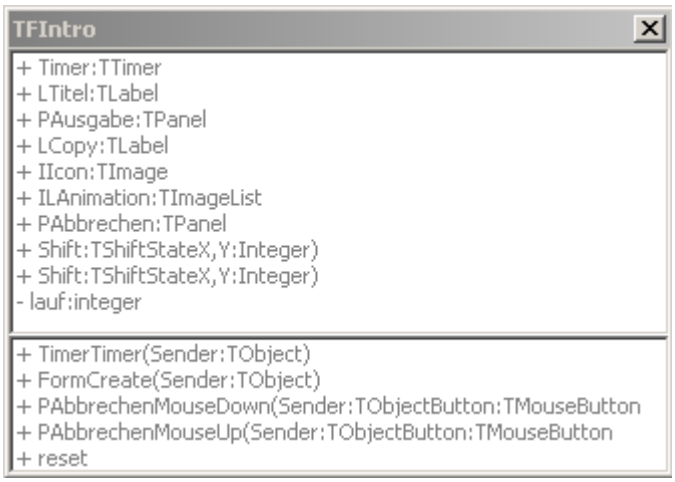

### 2.3.1.3 TFHilfe in der Unit UHilfe


### 2.3.1.4 TIButton in der Unit UImageButton

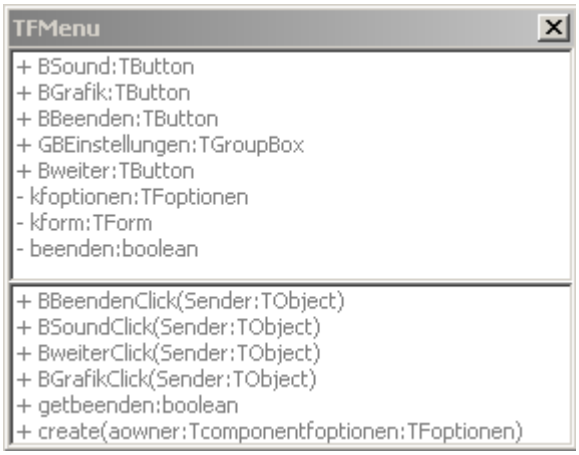


### 2.3.1.5 TFInfo in der Unit UInfo


### 2.3.1.6 TFIntro in der Unit UIntro

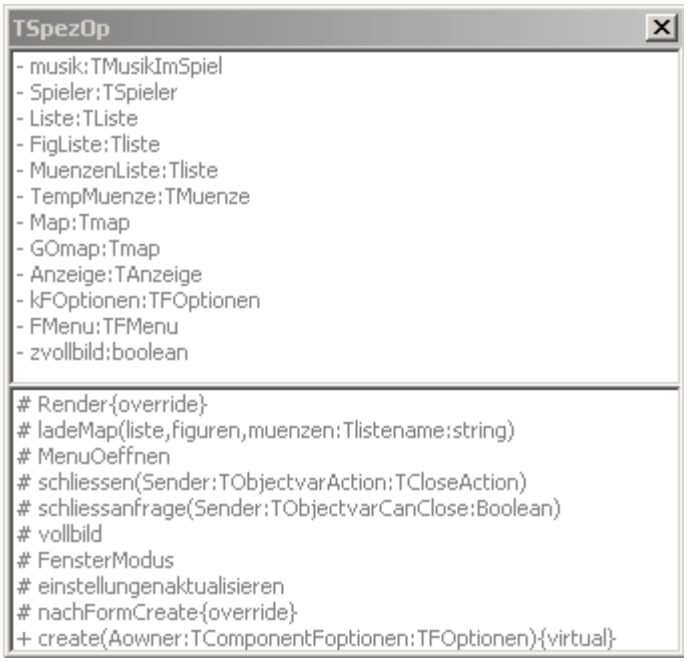



### 2.3.1.7 TFMenu in der Unit UMenu



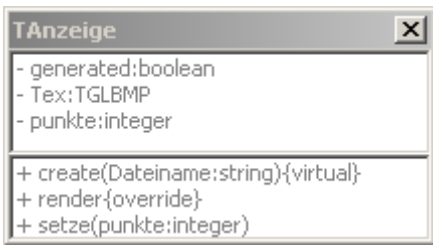
### 2.3.1.8 TFMusikImSpiel in der Unit UMusikImSpiel


### 2.3.1.10 TSpezOp in der Unit USpezOp



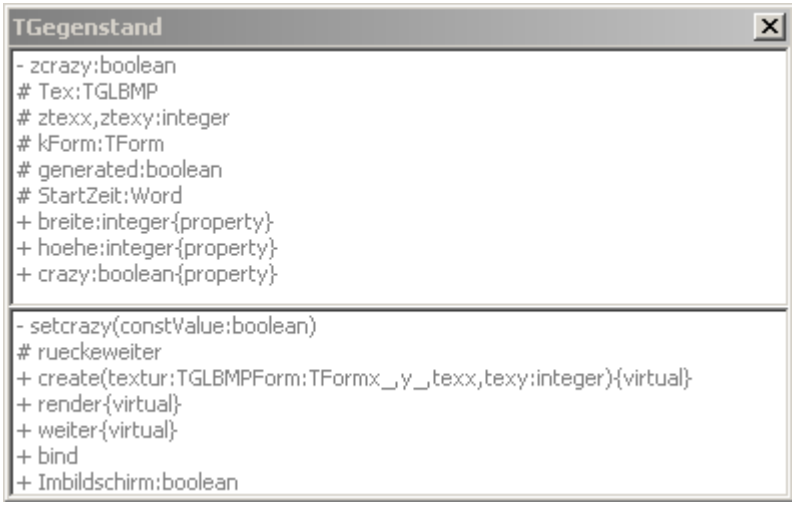

## 2.3.2 „Objekte"

### 2.3.2.1 TAnzeige in der Unit UAnzeige

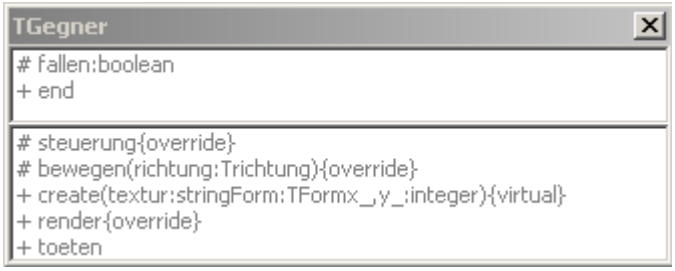


### 2.3.2.2 TFigur in der Unit UFigur


### 2.3.2.3 TGegenstand in der Unit UGegenstand

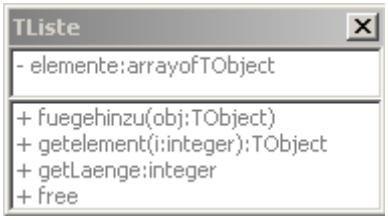


### 2.3.2.4 TGegner in der Unit UGegner

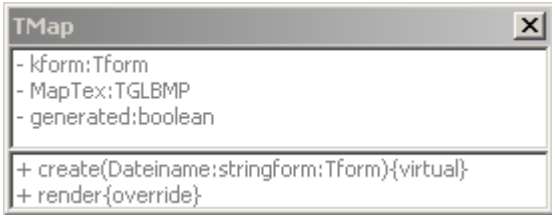



### 2.3.2.5 TGLObject in der Unit UGLObject

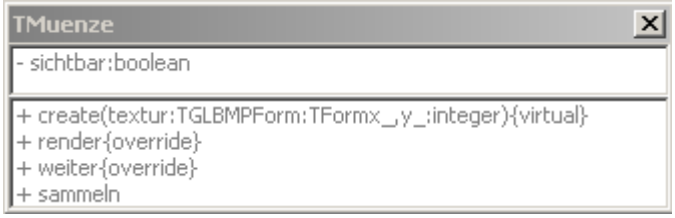

### 2.3.2.6 TListe in der Unit UListe



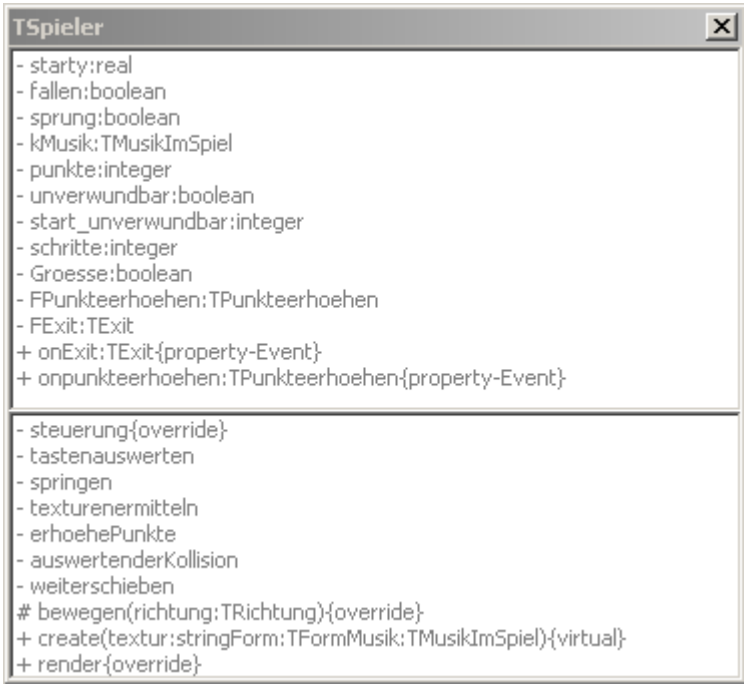

### 2.3.2.7 TMap in der Unit UMap



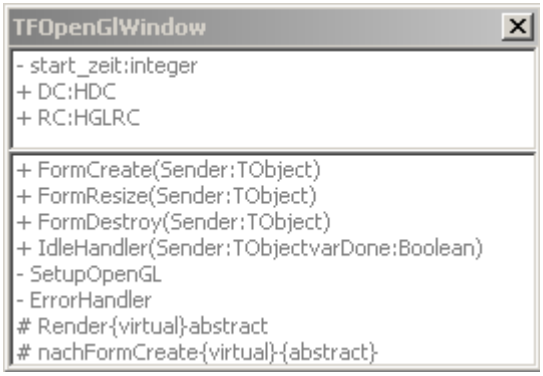

### 2.3.2.8 TMuenze in der Unit UMuenze



### 2.3.2.9 TSpieler in der Unit USpieler



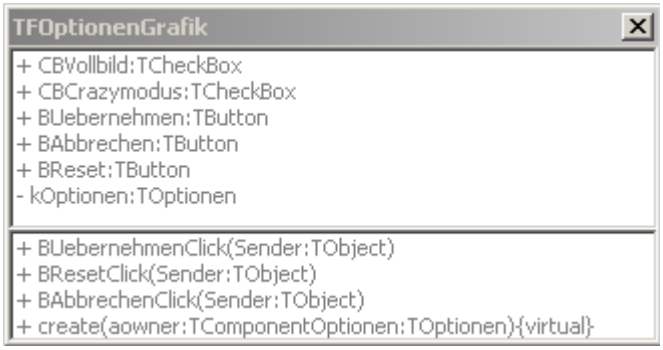

### 2.3.3 „OpenGL" – TFOpenGLWindow in der Unit UFOpenGLWindow



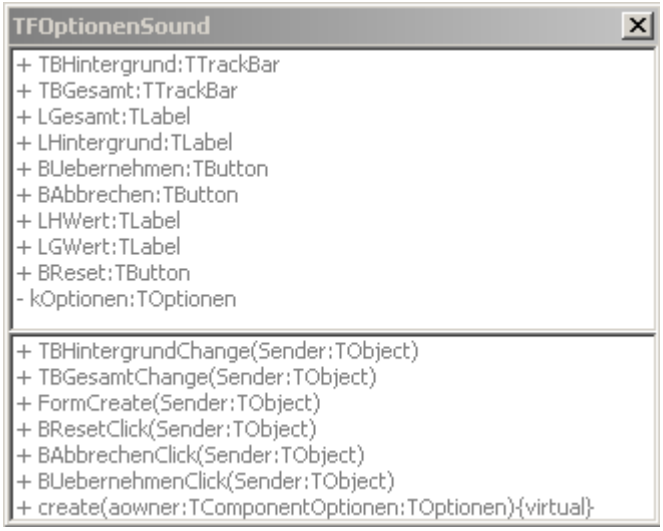
## 2.3.4 „Optionen“

### 2.3.4.1 TOptionen in der Unit UOptionen


### 2.3.4.2 TFOptionenGrafik in der Unit UOptionenGrafik

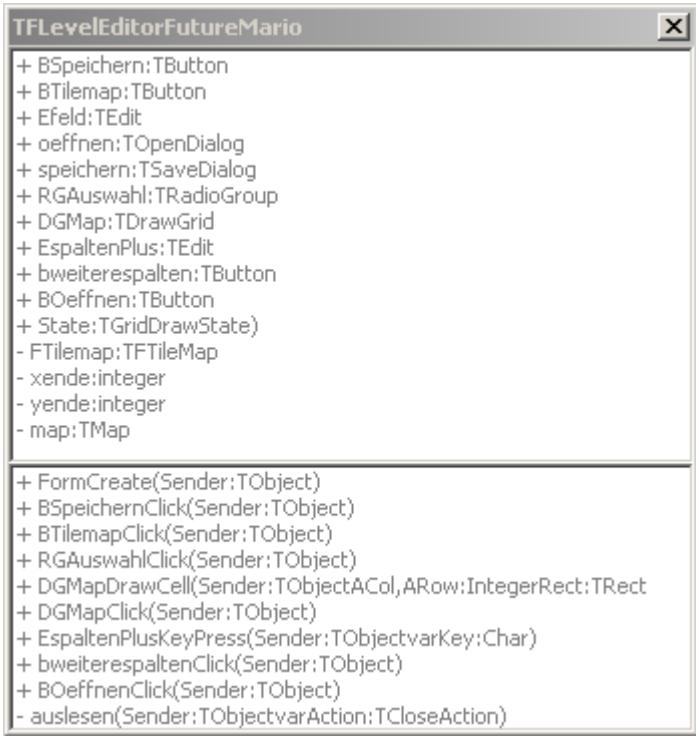


### 2.3.4.3 TFOptionenSound in der Unit UOptionenSound

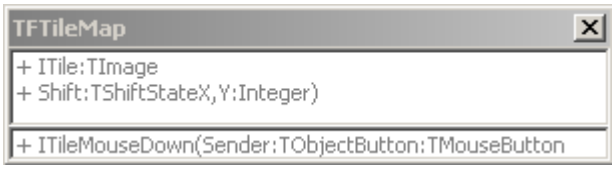



### 2.3.5 „Tools\LevelEditor"

### 2.3.5.1 TFLevelEditroFutureMario in der Unit UHaupt




### 2.3.5.2 TFTileMap in der Unit UTileMap

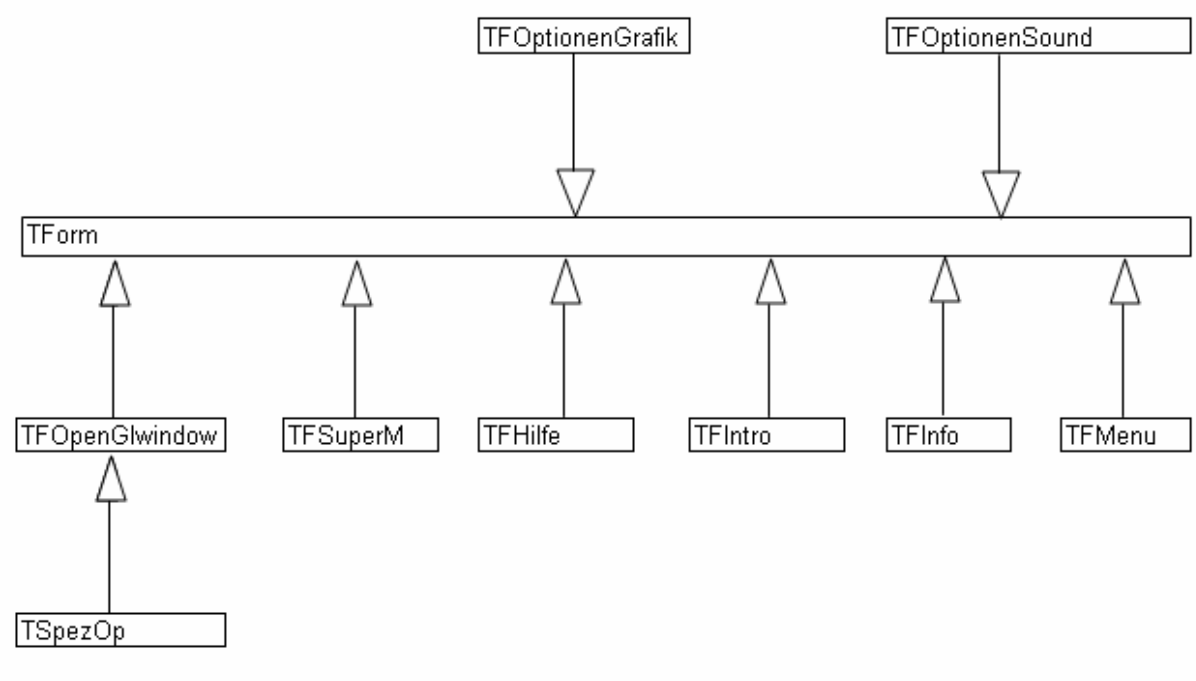


## 2.3.6 „USound"- TSound in der Unit Usound

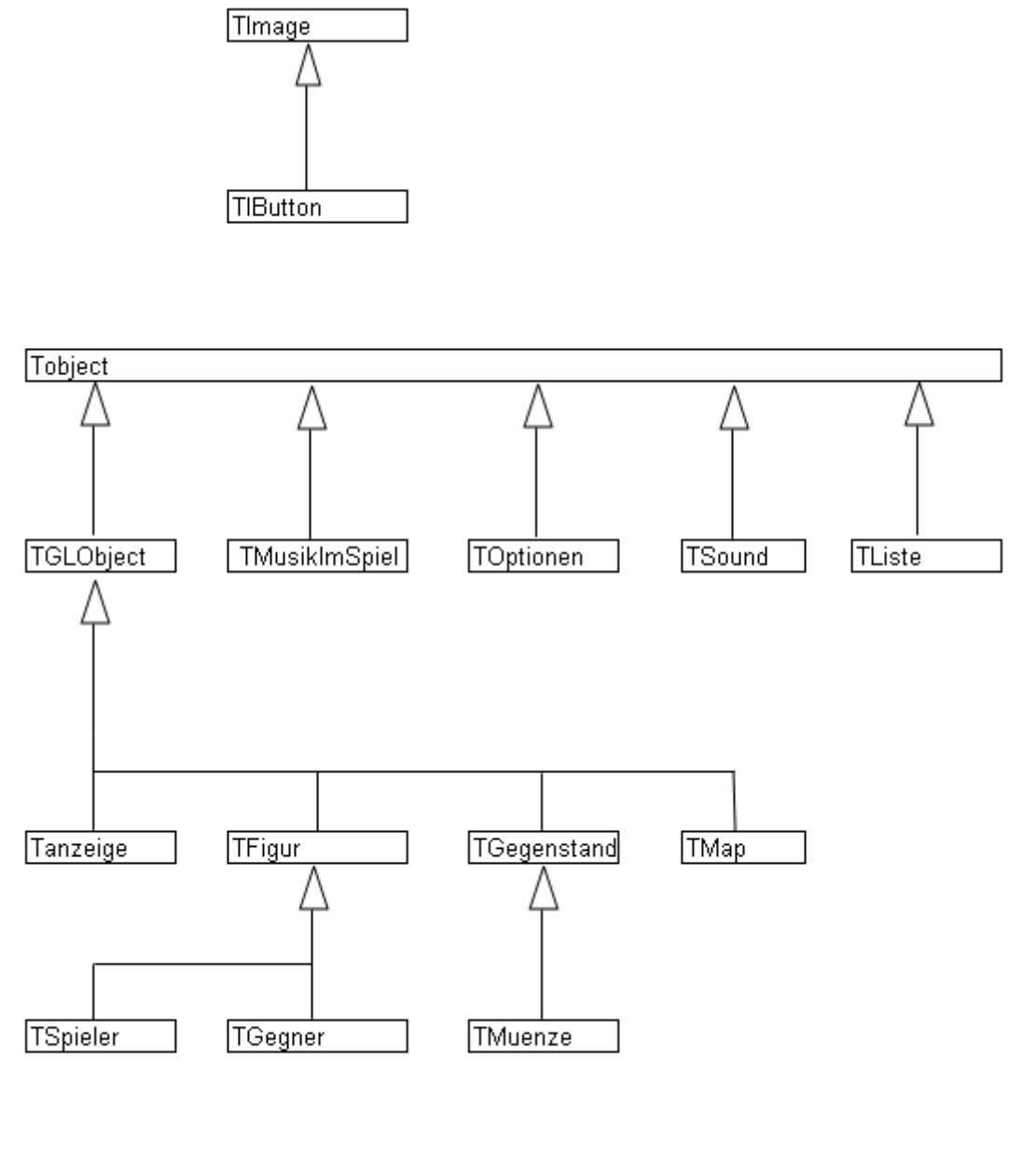
## 2.4 Beziehungen zwischen den Objekten /Klassen

### 2.4.1 Ist Beziehungen in Future Mario

Formulare

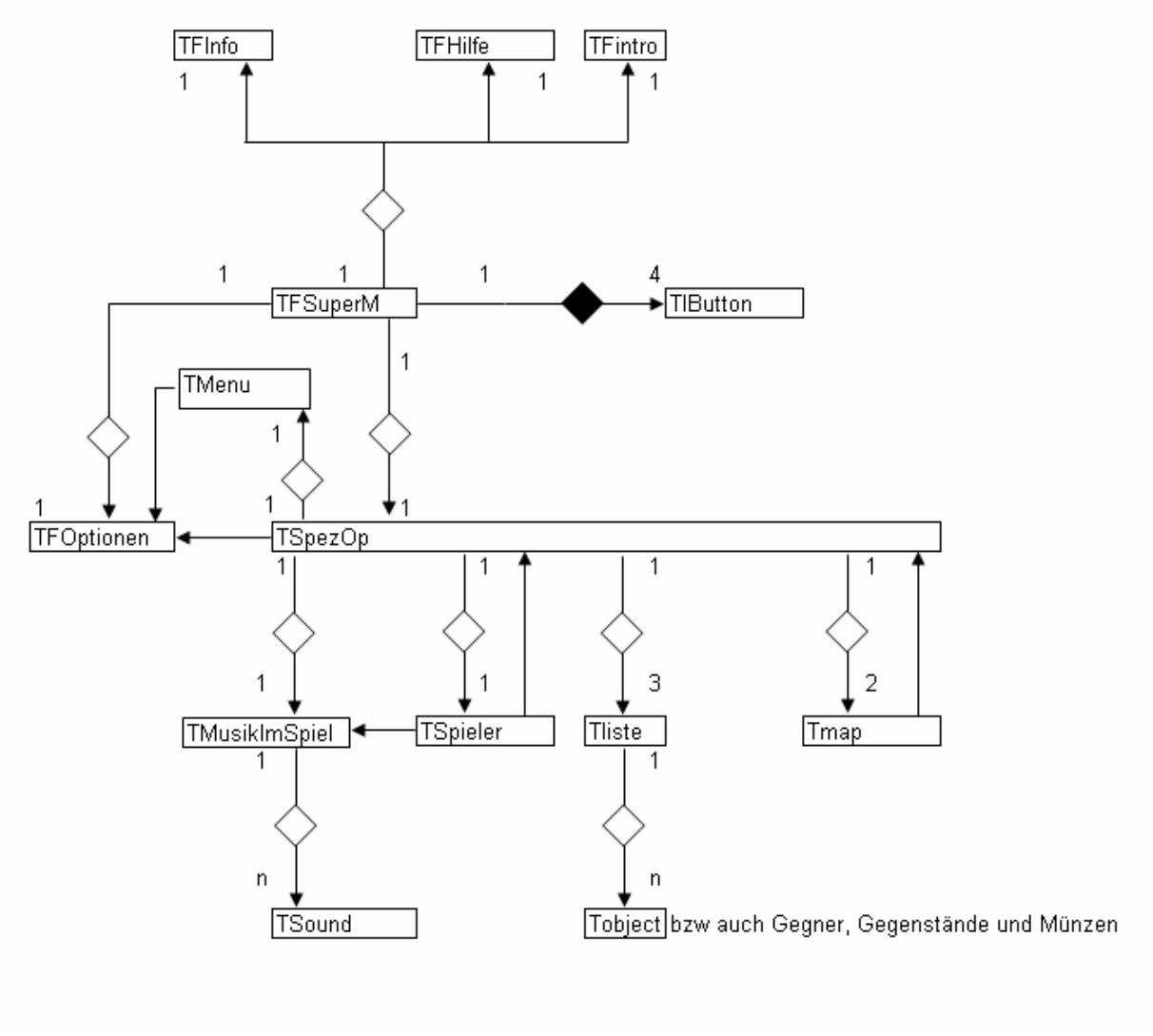


Alles was TObject ist und die ImageButtons:



## 2.4.2 Hat und Kennt Beziehungen in Future Mario

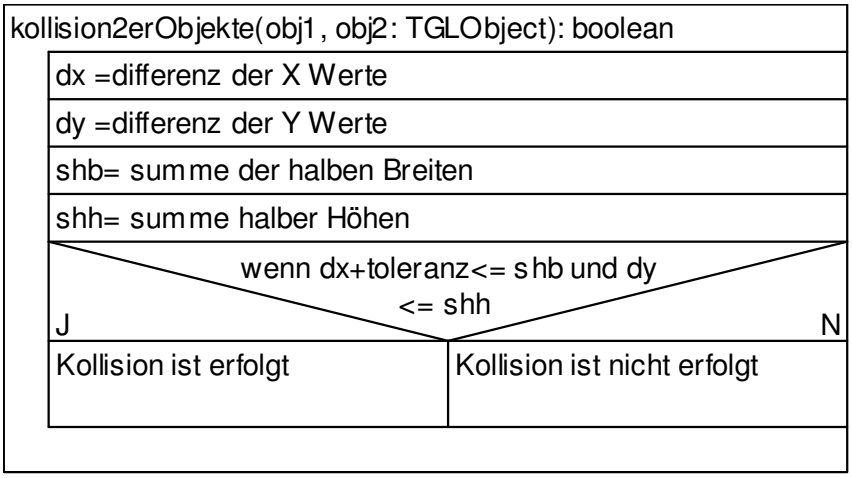
(Bei diesen Beziehungen werden nur die mit eigenen Klassen oder besonders wichtige angegeben, um alle Hat Beziehungen aufzuführen müsste man auch jeden Button, jedes Image usw. darstellen, dadurch könnte man keinen Überblick mehr erhalten)



## 2.5 Struktogramme

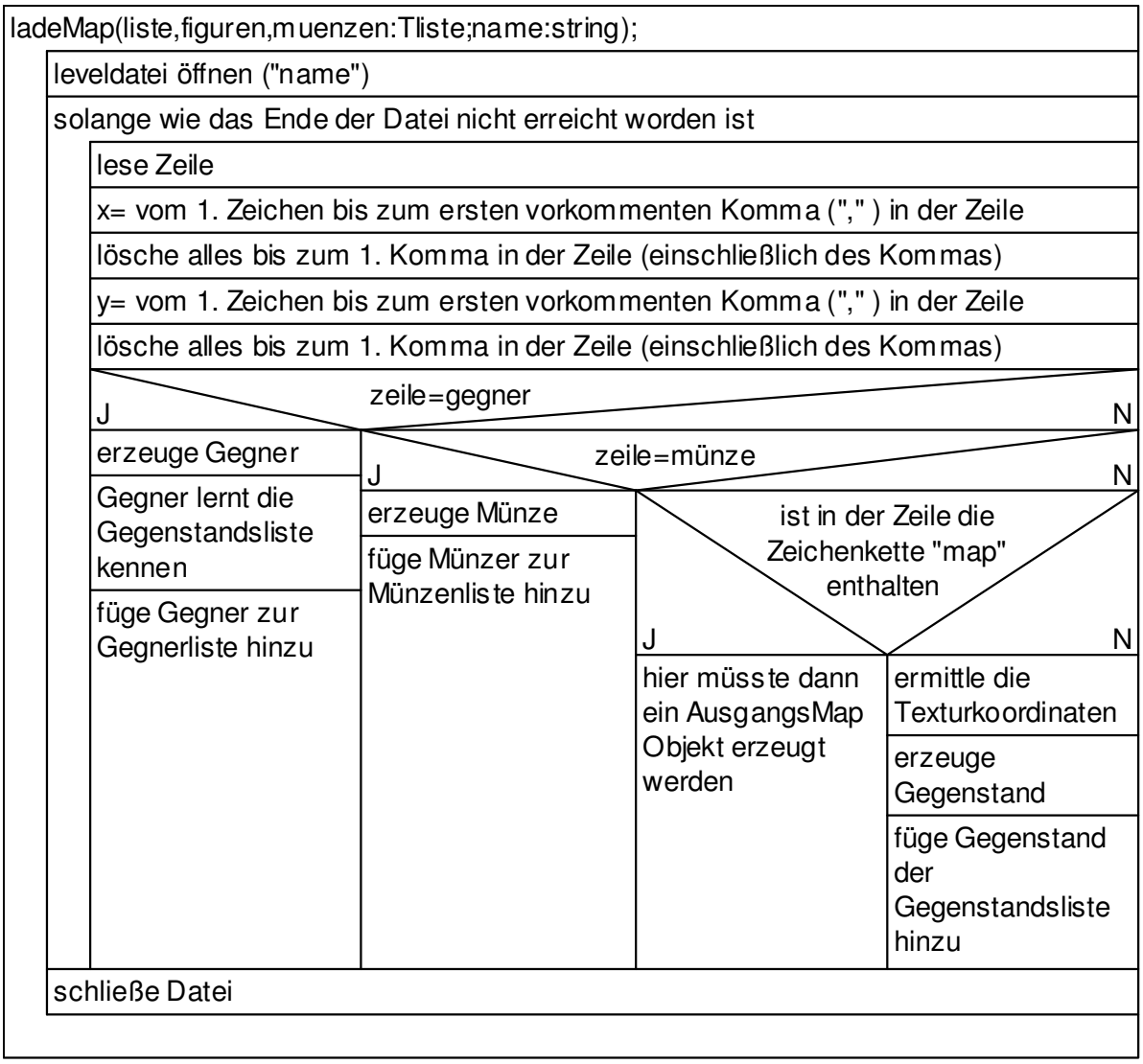
### 2.5.1 Kollisionsalgorithmus

Dieser Algo befindet sich in der Klasse TFigur



### 2.5.2 Laden der Karte

In der Klasse TSpezOp



# 3 Quelltexte

## 3.1 Future Mario

### 3.1.1 UFOptionen.pas

```
//
// Name:        UOptionen
// Datum:       20.03.2008
// letzte Änderung: 26.03.2008
// Version:       0.8
// Zweck:         OptionsFormular
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//
unit UFOptionen;

interface

uses Classes,dialogs,

    UOptionen,
    UOptionenSound,
    UOptionenGrafik;

type TFOptionen=class(TObject)

      private
       Optionen:TOptionen;
       FSound:TFOptionenSound;
       FGrafik:TFOptionenGrafik;

       function getcrazymodus: boolean;
       function getGesamtlautstaerke: integer;
       function getHintergrundlautstaerke: integer;
       function getsetmappack: string;
       function getvollbild: boolean;

      public
       constructor create(aowner:Tcomponent;filename:string);virtual;
       procedure showSound;
       procedure showGrafik;

       property HintergrundVolume:integer read getHintergrundlautstaerke;
       property GesamtVolume:integer read getGesamtlautstaerke;
       property Vollbild:boolean read getvollbild;
       property crazymodus:boolean read getcrazymodus;
       property Mappack:string read getsetmappack;
     end;
```

```
implementation

{ TFOptionen }

constructor TFOptionen.create;
begin
 inherited create;
 Optionen:=TOptionen.create(filename);

 FSound:=TFOptionenSound.create(aowner,Optionen);
 FGrafik:=TFOptionenGrafik.create(aowner,Optionen);

end;

function TFOptionen.getcrazymodus: boolean;
begin
 result:=optionen.crazymodus;
end;

function TFOptionen.getGesamtlautstaerke: integer;
begin
 result:=optionen.GesamtVolume;
end;

function TFOptionen.getHintergrundlautstaerke: integer;
begin
 result:=optionen.HintergrundVolume;
end;

function TFOptionen.getsetmappack: string;
begin
 result:=optionen.Mappack;
end;

function TFOptionen.getvollbild: boolean;
begin
 result:=optionen.Vollbild;
end;

procedure TFOptionen.showGrafik;
begin
 FGrafik.showmodal;
end;

procedure TFOptionen.showSound;
begin
 Fsound.showmodal;
end;

end.
```

### 3.1.2 UFunktionen.pas

```
//
// Name:        UFunktionen
// Datum:       20.03.2008
// letzte Änderung: 26.03.2008
// Version:       0.8
// Zweck:         eine Hilfsfunktionen in einer Unit
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//

unit UFunktionen;

interface
 uses Windows,Forms;

 function key_pressed(k1,k2:integer):boolean;overload;
 function key_pressed(k1:integer):boolean;overload;
 procedure zentriereFormular(Form:TForm);
implementation

function key_pressed(k1: integer): boolean;
begin
  result:=(GetKeyState(k1)<0) ;
end;

function key_pressed(k1, k2: integer): boolean;
begin
 result:=key_pressed(k1) or key_pressed(k2);
end;

procedure zentriereFormular(Form:TForm);
begin
 form.left:=(screen.Width - form.width) DIV 2;
 form.top:=(screen.height - form.height) DIV 2;
end;



end.
```

### 3.1.3 UHaupt.pas

```
//
// Name:         UHaupt
// Datum:        01.02.2008
// letzte Änderung: 26.01.2008
// Version:        0.8
// Zweck:          Hauptfenster
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//


unit UHaupt;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs,StdCtrls, ImgList, ExtCtrls, Buttons, Menus,

  USpezOp,
  UHilfe,
  UInfo,
  UIntro,
  UImageButton,
  UFunktionen,
  UTypenUndKonstanten,
  UFOptionen;

type
  TFSuperM = class(TForm)
    PStart: TPanel; // dient zur Ausrichtung des Menüs
    IHintergrund: TImage;
    IMario: TImage;
    Hauptmenue: TMainMenu;
    MSpiel: TMenuItem;
    MOptionen: TMenuItem;
    MHilfe: TMenuItem;
    MInfos: TMenuItem;
    MStarten: TMenuItem;
    MBeenden: TMenuItem;
    MSound: TMenuItem;
    MGrafik: TMenuItem;
    MIntro: TMenuItem;

    procedure FormCreate(Sender: TObject);
    procedure StartClick(Sender: TObject);
    procedure MBeendenClick(Sender: TObject);
    procedure MHilfeClick(Sender: TObject);
    procedure InfoClick(Sender: TObject);
    procedure MSoundClick(Sender: TObject);
    procedure MIntroClick(Sender: TObject);
    procedure MGrafikClick(Sender: TObject);

  private
    { Private-Deklarationen }
    x,y:integer;
```

```
    IStart:TIButton;
    IInfo:TIButton;
    IHilfe:TIButton;
    IBeenden:TIButton;

    OpenglWindow:TSpezOp;
    FInfo :TFInfo;
    FHilfe:TFHilfe;
    FIntro:TFintro;

    FOptionen:TFOptionen;

    procedure verstecken;
    procedure zeigen;
    procedure introende(Sender: TObject; var Action: TCloseAction);


  public
    { Public-Deklarationen }
  end;

var
  FSuperM: TFSuperM;

implementation

{$R *.dfm}

procedure TFSuperM.MBeendenClick(Sender: TObject);
begin
 Close;
end;

procedure TFSuperM.MGrafikClick(Sender: TObject);
begin
 FOptionen.showGrafik;
end;

procedure TFSuperM.MHilfeClick(Sender: TObject);
begin
 if not assigned(FHilfe)
  then FHilfe:=TFHilfe.Create(self);
 FHilfe.Show;
end;

procedure TFSuperM.InfoClick(Sender: TObject);
begin
 if not assigned(FInfo)
  then FInfo:=TFInfo.Create(self);
 FInfo.Show;
end;

procedure TFSuperM.MIntroClick(Sender: TObject);
begin
 FIntro.reset;
 FIntro.Show;
end;

procedure TFSuperM.introende(Sender: TObject; var Action: TCloseAction);
begin
 zeigen;
end;
```

```
procedure TFSuperM.MSoundClick(Sender: TObject);
begin
 FOptionen.showSound;
end;

procedure TFSuperM.StartClick(Sender: TObject);
begin
 if assigned(OpenGlWindow)
  then OpenGlWindow.free;
 OpenGlWindow:=TSpezOp.create(self,FOptionen);
 OpenGlWindow.show;
end;

procedure TFSuperM.verstecken;
begin
 x:=left;
 y:=top;
 hide;
 left:=-width;
 top:=-height;
end;

procedure TFSuperM.zeigen;
begin
 show;
 left:=x;
 top:=x;

end;

procedure TFSuperM.FormCreate(Sender: TObject);
begin

 with ihintergrund do
 begin
  Left:=0;
  top:=0;
  Picture.LoadFromFile(Kmenu+'\Hintergrund.bmp');
  Width:=Picture.Width;
  height:=Picture.Height;
  self.Width:=width;
  self.height:=height;
 end;

 IStart:=TIButton.create(self,Kmenu+'\start.bmp',Kmenu+'\start_aktiv.bmp');
 with IStart do
 begin
  OnClick:=StartClick;
  Left:=PStart.Left;
  Top:=PStart.Top;
 end;

 IInfo:=TIButton.create(self,Kmenu+'\info.bmp',Kmenu+'\info_aktiv.bmp');;
 with IInfo do
 begin
  OnClick:=InfoClick;
  Left:=Istart.left-100;
  Top:=Istart.Top+100;
 end;

 with imario do
 begin
  Left:=IInfo.left+IInfo.Width+30;
```

```
  top:=IInfo.top-20;
  Picture.LoadFromFile(Kmenu+'\mario.bmp');
  picture.Bitmap.TransparentColor:=rgb(255,255,255);
  Width:=Picture.Width;
  height:=Picture.Height;
 end;


IHilfe:=TIButton.create(self,Kmenu+'\hilfe.bmp',Kmenu+'\hilfe_aktiv.bmp');;
 with IHilfe do
 begin
  OnClick:=MHilfeClick;
  Left:=Istart.left-100;
  Top:=Istart.Top+200;
 end;


IBeenden:=TIButton.create(self,Kmenu+'\beenden.bmp',Kmenu+'\beenden_aktiv.b
mp');
 with IBeenden do
 begin
  OnClick:=MBeendenClick;
  Left:=Istart.left;
  Top:=Istart.Top+300;
 end;
 zentriereFormular(self);
 verstecken;

 FIntro:=TFIntro.create(self);

 Fintro.OnClose:=introende;
 FIntro.Show;

 FOptionen:=TFOptionen.create(self,koptionen);

end;

end.
```

### 3.1.4 UHilfe.pas

```
//
// Name:        UHilfe
// Datum:       20.03.2008
// letzte Änderung: 20.03.2008
// Version:       0.8
// Zweck:         Hilfe-fenster
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//

unit UHilfe;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,

  UTypenUndKonstanten,
  UFunktionen;
type
  TFHilfe = class(TForm)
    LBHilfe: TListBox;
    procedure FormCreate(Sender: TObject);
  end;

implementation

{$R *.dfm}

procedure TFHilfe.FormCreate(Sender: TObject);
begin
 zentriereFormular(self);
 LBHilfe.Items.LoadFromFile(Hilfe);
end;

end.
```

### 3.1.5 UImageButton.pas

```
//
// Name:        UImageButton
// Datum:       22.03.2008
// letzte Änderung: 22.03.2008
// Version:          0.8
// Zweck:           BildTasten für Startbildschirm
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//
unit UImageButton;

interface

uses ExtCtrls,forms,Controls,Classes;

type TIButton=class(TImage)
      private
       aktivBmp,unaktivBmp:string;
      protected
       procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X:
Integer; Y: Integer);override;
       procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X:
Integer; Y: Integer);override;
      public
       constructor create(parent: TForm;  nichtaktiv,aktiv:
string);virtual;
     end;

implementation

{ TIButton }

constructor TIButton.create(parent: TForm;  nichtaktiv,aktiv: string);
begin
 inherited create(parent);
 self.parent:=parent;
 aktivBmp:=aktiv;
 unaktivBmp:=nichtaktiv;
 picture.LoadFromFile(unaktivBmp);
 width:=picture.Width;
 height:=picture.Height;
end;

procedure TIButton.MouseDown(Button: TMouseButton; Shift: TShiftState; X,
  Y: Integer);
begin
 picture.LoadFromFile(aktivBmp);
end;

procedure TIButton.MouseUp(Button: TMouseButton; Shift: TShiftState; X,
  Y: Integer);
begin
 picture.LoadFromFile(unaktivBmp);
end;

end.
```

### 3.1.6 UInfo.pas

```
//
// Name:        UInfo
// Datum:       19.03.2008
// letzte Änderung: 19.03.2008
// Version:          0.8
// Zweck:           Informationsfenster
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//


unit UInfo;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,

  UTypenUndKonstanten,
  UFunktionen ;
type
  TFInfo = class(TForm)
    LBInfo: TListBox;
    procedure FormCreate(Sender: TObject);
  end;


implementation

{$R *.dfm}

procedure TFInfo.FormCreate(Sender: TObject);
begin
 zentriereFormular(self);
 LBInfo.Items.LoadFromFile(Info);
end;

end.
```

## 3.1.7 UIntro.pas

```
//
// Name:        UIntro
// Datum:       22.03.2008
// letzte Änderung: 25.03.2008
// Version:          0.8
// Zweck:           SpielIntro
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//


unit UIntro;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ExtCtrls, StdCtrls, ImgList,

  UFunktionen,
  UTypenUndKonstanten;

type
  TFIntro = class(TForm)
    Timer: TTimer;
    LTitel: TLabel;
    PAusgabe: TPanel;
    LCopy: TLabel;
    IIcon: TImage;
    ILAnimation: TImageList;
    PAbbrechen: TPanel;
    procedure TimerTimer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure PAbbrechenMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure PAbbrechenMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
   lauf:integer;
  public
   procedure reset;
  end;



implementation

{$R *.dfm}

procedure TFIntro.FormCreate(Sender: TObject);
begin
 zentriereFormular(self);
 LTitel.Caption:=Ktitel;
 LCopy.caption:=KCopy;
 reset;
```

```
end;

procedure TFIntro.PAbbrechenMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 (sender as Tpanel).Bevelouter:=bvlowered;
 (sender as Tpanel).color:=clMenuHighlight;
end;

procedure TFIntro.PAbbrechenMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 (sender as Tpanel).Bevelouter:=bvraised;
 (sender as Tpanel).color:=clGradientActiveCaption;
 timer.Enabled:=false;
 close;
end;

procedure TFIntro.reset;
begin
 lauf:=0;
 timer.Enabled:=true;
end;

procedure TFIntro.TimerTimer(Sender: TObject);
var Icon:TIcon;
begin
  Icon:=TIcon.Create();
  ILAnimation.GetIcon(lauf,Icon);
  IIcon.Picture:=TPicture(Icon);
  Icon.free;
  inc(lauf);
  if lauf=ILAnimation.Count
   then
    begin
      close;
      timer.Enabled:=false;
    end;
end;

end.
```

### 3.1.8 UMenu.pas

```
//
// Name:        UMenu
// Datum:       26.03.2008
// letzte Änderung: 26.03.2008
// Version:       0.1
// Zweck:         Spielmenü
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//


unit UMenu;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,

  UFOptionen;

type
  TFMenu = class(TForm)
    BSound: TButton;
    BGrafik: TButton;
    BBeenden: TButton;
    GBEinstellungen: TGroupBox;
    Bweiter: TButton;
    procedure BBeendenClick(Sender: TObject);
    procedure BSoundClick(Sender: TObject);
    procedure BweiterClick(Sender: TObject);
    procedure BGrafikClick(Sender: TObject);
   private
    kfoptionen:TFoptionen;
    kform:TForm;
    beenden:boolean;
   public
    function getbeenden:boolean;
    constructor create(aowner:Tcomponent;foptionen:TFoptionen);
  end;


implementation

{$R *.dfm}

{ TFMenu }

procedure TFMenu.BBeendenClick(Sender: TObject);
begin
 beenden:=true;
 kform.close;
 close;
end;
```

```
procedure TFMenu.BGrafikClick(Sender: TObject);
begin
 kfoptionen.showGrafik;
end;

procedure TFMenu.BSoundClick(Sender: TObject);
begin
 kfoptionen.showSound;
end;

procedure TFMenu.BweiterClick(Sender: TObject);
begin
 close;
end;

constructor TFMenu.create(aowner: Tcomponent; foptionen: TFoptionen);
begin
 inherited create(aowner);
 kform:=TForm(aowner);
 kfoptionen:=foptionen;
end;

function TFMenu.getbeenden: boolean;
begin
 result:=beenden;
end;

end.
```

## 3.1.9 UMusikImSpiel.pas

```
//
// Name:        UMusikImSpiel
// Datum:       20.01.2008
// letzte Änderung: 26.01.2008
// Version:          0.8
// Zweck:           Zum abspielen der Musik in diversen Situationen
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//



unit UMusikImSpiel;
   // Die unit dient später zum einbinden in das Spiel da das Formular nur
zu testzwecken da ist
interface
 uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls,IniFiles,

  USound;



type
   // Datenstruktur zum aufnehmen der Sounddateien und deren Namen
  TMusikdateien = Array OF record
                            Sound     : TSound;
                            Soundname : string;
                           end;

  //Klasse mit  Methoden zum abspielen der Sounds
 TMusikImSpiel = class(TObject)

  private
   // Musikdaten aufnehmen
   Musikdateien : TMusikdateien;
   // Lautstärke für alle Sound Daten festlegen
   Lautstaerke  : byte;
   // KonfigurationsIni Datei , Pfad und Name
   CFG:String;

   // initialisierung
   procedure MusikLaden;
  public
   // lädt alle musikdateien in das  array musikdateien
   // @param cfg: Pfad+Name Cfg Datei für Sound
   Constructor create(cfg:String);virtual;

   // spielt die sounds abspielen einzeln ab
   procedure MusikAbspielen(Name : string);
   // stopt die einzlnen Sounds
   Procedure MusikStop(Name : string);
   //pausiert die einzelnen Sounds
```

```
    Procedure MusikPause(Name : string);
    //stoppen aller Sound
    Procedure AlleStopp;
    //pausiert alle Sounds
    Procedure AllePause;
    //ermitteln der Lautstärke
    function getLautstaerke:byte;
    // lautstärke für alle lieder festlegen
    Procedure LautstaerkeRegeln(Volume:byte);overload;
    // dauerPlay eines Musikstückes
    procedure dauerhaftAbspielen(name:string);
    // lautstärke eines einzelnen Liedes ändern
    procedure lautstaerkeregeln(name:string;Volume:byte);overload;
    // lautstärke aller
    procedure lautstaerkeMaster(vol:byte);


end;

implementation

{ TMusikImSpiel }

procedure TMusikImSpiel.AllePause;
Var i : integer;
begin
 // durhsucht das array und paussiert dann die lieder nacheinander einzeln,
 //hier muss lenght-1 da die Letzte "Zelle" nicht belegt ist und der pause
befehl darauf ein fehler verusachen würde
 For i:=0 To length(Musikdateien)-1 DO
  Musikdateien[i].sound.pause;
end;

procedure TMusikImSpiel.AlleStopp;
VAR i : integer;
begin
 // durhsucht das array und stoppt dann die lieder nacheinander einzeln,
 //hier muss lenght-1 da die Letzte "Zelle" nicht belegt ist und der stopp
befehl darauf ein fehler verusachen würde
 For i:=0 To length(Musikdateien)-1 DO
  Musikdateien[i].sound.stop;
end;


constructor TMusikImSpiel.create;
begin
 inherited Create();    //Construktor der SuperKlasse

 lautstaerke:=255;    // aus ner Ini Lesen!
 self.CFG:=Cfg;

 MusikLaden;

end;

function TMusikImSpiel.getLautstaerke: byte;
begin
 result:=Lautstaerke;
end;

procedure TMusikImSpiel.LautstaerkeRegeln(Volume:byte);
VAR i : integer;
begin
```

```
 Lautstaerke:=volume;

 For i:=0 To length(Musikdateien)-1 DO
  Musikdateien[i].sound.setVolume(volume);

end;
procedure TMusikImSpiel.lautstaerkeregeln(name:string;Volume:byte);
var i : integer;
begin
 For i:=0 To length(Musikdateien)-1 DO
  if Musikdateien[i].Soundname=name
   then Musikdateien[i].sound.setVolume(Volume);
end;


procedure TMusikImSpiel.MusikAbspielen(Name: string);
 var i:integer;
begin
// Hier wird unsere Ini datei durchgegangen und sobald er den Soundnamen
findet spielt er ihn ab.
 for i:=0 to length(Musikdateien)-1 do
  if Musikdateien[i].Soundname=name
   then Musikdateien[i].sound.play;
end;

procedure TMusikImSpiel.MusikLaden;

var i : integer;
    Ini: TIniFile;
    Musikliste:TStringList;
    Dateiname:string;
begin
  Musikliste:=TStringlist.create();
  ini:=nil;
  try
    Ini:=TIniFile.Create(CFG);
    Ini.ReadSection('Soundfiles',Musikliste);  // Hier wird eine Sektion in
der Ini datei ausgelesen und in der Liste aabgelegt

    SetLength(Musikdateien,Musikliste.count); //Hier legen wir die Länge
des Array fest , da unser dynamisches Array am Anfang die Länge 0 hatte


    for i:=0 to Musikliste.Count-1 do
    begin         // Diese Schleife kreiiert die einzelnen Instanzen des
array's
     Musikdateien[i].sound := TSound.Create;
     Musikdateien[i].Soundname:=Musikliste[i];  //Nun wird jeder "Zelle"
unseres Array's ein Name zugeweisen (im record sound name werden die
Soundnamen zugewiesen)

     Dateiname:=Ini.ReadString('Soundfiles',Musikliste[i],'');//Nun setzten
wir den Dateiname indem wir die Musikliste auslesen  damit später die Sound
über den Dateinamen gelsen werden können
     Musikdateien[i].sound.load('Soundfiles\'+Dateiname); //Letztendlich
müssen wir noch die eigentlichen sounds in unseren Array mit dem record
aufnhemen
    end;


  finally    // im Finally part werden anweisungen ausgeführt die bei jeder
try anweisung ausgeführt werden, egal was die try anweisung ergibt
    Ini.Free;    // freigeben des Speichers des Zeigers auf die Ini - Datei
```

```
    Musikliste.Free;  // Nun muss noch der Speiher der Musikliste
freigegebenem werden
  end;

end;

procedure TMusikImSpiel.MusikPause(Name: string);
VAR i : integer;
begin
 //hier durchsuchen wir nun unsere Array nach dem übergebenen namen um dann
auch die musik anhalten zu können um sie später weiterlaufen zu lassen
 For i:=0 To length(Musikdateien)-1 Do
  if Musikdateien[i].Soundname=name
   then Musikdateien[i].sound.pause;
end;

procedure TMusikImSpiel.MusikStop(Name: string);
Var i : integer;
begin
 // Druchgehe des Array zum finden des Dateinamen und stoppen des Sounds
 For i:=0 To length(Musikdateien) Do
  if Musikdateien[i].Soundname=name
   then Musikdateien[i].sound.stop;
end;


procedure TmusikImSpiel.dauerhaftAbspielen;
VAR i : integer;
begin
 //hier durchsuchen wir nun unsere Array nach dem übergebenen namen um dann
auch die musik anhalten zu können um sie später weiterlaufen zu lassen
 For i:=0 To length(Musikdateien)-1 Do
  if Musikdateien[i].Soundname=name
   then Musikdateien[i].sound.dauerplay;

end;
procedure TmusikImSpiel.lautstaerkeMaster(vol:byte);
begin
 TSound.SetMasterVol(vol);
end;


end.
```

## 3.1.10 USpezOp.pas

```
//
// Name:        USpezOp
// Datum:       01.02.2008
// letzte Änderung: 27.03.2008
// Version:          0.9
// Zweck:           Spezialisiertes OpenGl Fenster zur Ausgabe des Spieles
//
// Programmierer:   Steve Göring, Daniel Renner
// Kontakt:         stg7@gmx.de, daniel26289@yahoo.de
//
// Copyright:       2008
//

unit USpezOp;


interface
uses Classes,DGLOpenGL,GLBMP,SysUtils,dialogs,Forms,UFigur,

     UFunktionen,
     UGegner ,
     UopenGlWindow,
     USpieler,
     Umap,
     UGegenstand,
     UListe,
     UMusikImSpiel,
     UMuenze,
     UFOptionen,
     UMenu,
     UAnzeige;

type TSpezOp=class(TFOpenGlwindow)
      private
       musik:TMusikImSpiel;
       Spieler:TSpieler;
       Liste:TListe;
       FigListe:Tliste;
       MuenzenListe:Tliste;
       TempMuenze : TMuenze;
       Map:Tmap;
       GOmap:Tmap;
       Anzeige:TAnzeige;
       kFOptionen:TFOptionen;
       FMenu:TFMenu;
       zvollbild:boolean;
      protected
       procedure Render;override;
       procedure ladeMap(liste,figuren,muenzen:Tliste;name:string);
       procedure MenuOeffnen;
       procedure schliessen(Sender: TObject; var Action: TCloseAction);
       procedure schliessanfrage(Sender: TObject; var CanClose: Boolean);
       procedure vollbild;
       procedure FensterModus;
       procedure einstellungenaktualisieren;
       procedure nachFormCreate;override;
      public
       constructor create(Aowner:TComponent;Foptionen:TFOptionen);virtual;
```

```
    end;

implementation

{ TSpezOp }



constructor TSpezOp.create(Aowner: TComponent; Foptionen: TFoptionen);

begin
  inherited create(aowner);

  Anzeige:=TAnzeige.create('pics\zahlen.png');

  onclose:=schliessen;
  onCloseQuery:=schliessanfrage;

  musik:=TMusikImSpiel.create('cfg\config.ini');
  musik.dauerhaftAbspielen('Hintergrund');
  musik.MusikAbspielen('Hintergrund');

  Spieler:=TSpieler.create( 'pics\sm2.png',self,Musik);

  Spieler.onpunkteerhoehen:=Anzeige.setze;
  spieler.onExit:=MenuOeffnen;

  Map:=TMap.create('pics\Background.png',self);


  Liste:=Tliste.create;
  FigListe:=TListe.Create;

  MuenzenListe:=Tliste.Create;



  // Laden der Map
  ladeMap(liste,FigListe,MuenzenListe,FOptionen.Mappack);

  Spieler.kennt(liste);
  //Gegner.kennt(liste);
  Spieler.kenneMuenzen(MuenzenListe);

  Spieler.kenntFiguren(FigListe);

  kFOptionen:=FOptionen;

  zentriereFormular(self);

  FMenu:=TFMenu.create(self,FOptionen);

  zentriereFormular(Fmenu);
end;

procedure TSpezOp.einstellungenaktualisieren;
begin
 musik.lautstaerkeMaster(round(kFoptionen.GesamtVolume*2.55));

musik.lautstaerkeregeln('Hintergrund',round(kfoptionen.HintergrundVolume*2.
55));
```

```
 if kFoptionen.Vollbild
  then vollbild
  else
   if zvollbild
    then FensterModus;
end;

procedure TSpezOp.FensterModus;
begin
 width:=651;
 height:=400;
 borderstyle:=bsSizeable;
 zentriereFormular(self);

end;

procedure TSpezOp.ladeMap(liste,figuren,muenzen:Tliste;name:string);
var Gegenstand:TGegenstand;
    Muenze:TMuenze;
    Gegner:TGegner;
    f:textfile;
    zeile:string;
    x,y,texx,texy:integer;
    textur,texturMuenzen:TGLBMP;
begin

 textur:=TGLBMP.Create('pics\tileset2.png');
 textur.ColorKey(255,0,255,0);
 texturMuenzen:=TGLBMP.Create('pics\Coin.png');


 assignFile(f,name);
 reset(f);

 while not eof(f) do
 begin
  readln(f,zeile);
  if not(zeile='')
   then
    begin
     x:=strtoint(copy(zeile,1,POS(',',zeile)-1));
     DELETE(zeile,1,POS(',',zeile));
     y:=strtoint(copy(zeile,1,POS(',',zeile)-1));
     DELETE(zeile,1,POS(',',zeile));

     if zeile='gegner'
      then
       begin
         Gegner:=TGegner.create('pics\BusterBeetle.png',self,x*40,(11-
y)*40-20);
         gegner.kennt(liste);
         figuren.fuegehinzu(Gegner);
       end
      else
       if zeile='münze'
        then
         begin
          Muenze:=TMuenze.create(texturMuenzen,self,x*40,(11-y)*40-20);
          muenzen.fuegehinzu(muenze);
         end
         else
          if POS('map',zeile) > 0
           then   // Objekt für Mapwechsel.. noch nicht vorhanden
```

```
           else
            begin
             texx:=strtoint(copy(zeile,1,POS(' ',zeile)-1));
             DELETE(zeile,1,POS(' ',zeile));
             texy:=strtoint(zeile);
             Gegenstand:=TGegenstand.create(textur,self,x*40,(11-y)*40-
20,texx,texy);
             Liste.fuegehinzu(Gegenstand);
            end;
    end;
 end;
 closefile(f);

end;

procedure TSpezOp.render;
var Gegenstand:TGegenstand;
    i:integer;
    Figur:TFigur;
begin

 if spieler.nichtlebend
  then
   begin

    if not assigned(GoMap)
     then
      begin
       GoMap:=TMap.create('pics\gameover.png',self);
       musik.allestopp;
       musik.dauerhaftAbspielen('GameOver');
       musik.MusikAbspielen('GameOver');
      end;
    GoMap.render;
    if key_pressed(ord(27))
     then close;
   end
  else
   begin

    Anzeige.render;
    Spieler.render;
    TempMuenze:=TMuenze(MuenzenListe.getelement(0));
    if assigned(tempmuenze)
     then Tempmuenze.bind;

    for i:=0  to MuenzenListe.getLaenge-1 do
    begin
     TempMuenze:=TMuenze(MuenzenListe.getelement(i));
     if assigned(TempMuenze)
      then Tempmuenze.render;
    end;


    for i:= 0 to Figliste.getLaenge-1 do
    begin
     Figur:=TFigur(FigListe.getelement(i));
     if assigned(figur)
      then Figur.render;
    end;

    Gegenstand:=TGegenstand(Liste.getelement(0));
    if assigned(gegenstand)
```

```
     then gegenstand.bind;

    for i:= 0 to liste.getLaenge-1 do
    begin
     Gegenstand:=TGegenstand(Liste.getelement(i));
     if assigned(gegenstand) and gegenstand.imbildschirm
      then
       begin
        gegenstand.crazy:=kfoptionen.crazymodus;
        Gegenstand.render;
       end;
    end;

    if not kfoptionen.crazymodus
     then map.render;

   end;
end;


procedure TSpezOp.schliessanfrage(Sender: TObject; var CanClose: Boolean);
begin
 if not Fmenu.Visible
  then Fmenu.ShowModal;
 canclose:=Fmenu.getbeenden;
end;

procedure TSpezOp.schliessen(Sender: TObject; var Action: TCloseAction);
begin
 musik.AlleStopp;
end;

procedure TSpezOp.vollbild;
begin
 borderstyle:=bsnone;
 left:=0;
 top:=0;
 width:=screen.Width;
 height:=screen.Height;
 zvollbild:=true;
end;

procedure TSpezOp.MenuOeffnen;
begin
 if not Fmenu.Visible
  then Fmenu.ShowModal;
 einstellungenAktualisieren;
end;

procedure TSpezOp.nachFormCreate;
begin
 einstellungenaktualisieren;
end;

end.
```

### 3.1.11 UTypenUndKonstanten.pas

```
//
// Name:        UTypenUndKonstanten
// Datum:       20.03.2008
// letzte Änderung: 27.03.2008
// Version:          0.2
// Zweck:           Sammlung alle Typen und Konstanten in einer separaten
Unit
//               zur zentralen Verwaltung
//
// Programmierer:   Steve Göring, Daniel Renner
// Kontakt:         stg7@gmx.de, daniel26289@yahoo.de
//
// Copyright:       2008
//

unit UTypenUndKonstanten;

interface

const xToleranz=10;
      yToleranz=6;
      sprunghoehe=100;
      NearClipping = 1;
      FarClipping  = 1000;

Const cfg='cfg';
      Info =cfg+'\Info.txt';
      Hilfe =cfg+'\Hilfe.txt';
      KOptionen=cfg+'\optionen.ini';

      KTitel='Future'+chr(13)+'    Mario';
      KCopy='von Steve Göring und Daniel Renner'+chr(13)+chr(13)+'
Projektarbeit im Fach Angewandte Technik '+chr(13)+chr(13)+'
stg7@gmx.de,Daniel26289@yahoo.de '+chr(169)+' 2008 ';
      KPics='pics' ;
      Kmenu=Kpics+'\menu';
      anziehung=2;

Type TPos=record
            x:integer;
            y:integer;
           end;
     TRichtung=(Rstop,Rrechts,Roben,Rlinks,Runten);

implementation

end.
```

## 3.1.12 Objekte\UAnzeige.pas

```
unit UAnzeige;

interface

uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,

     UGLObject;

type TAnzeige=class(TGLObject)
      private
       generated:boolean;
       Tex:TGLBMP;
       punkte:integer;

      public
       constructor create(Dateiname:string);virtual;
       procedure render;override;
       procedure setze(punkte:integer);
     end;
implementation

{ TMap }

constructor TAnzeige.create;
begin
  tex:= TGLBMP.Create(Dateiname);
  tex.ColorKey(0,0,0,0);
  hoehe:=40;
  breite:=40;
  x:=640 div 2;
  y:=480 - hoehe div 2 ;
end;

procedure TAnzeige.render;
var links:real;
    ziffer:integer;
begin

  glMatrixMode(GL_MODELVIEW);

  if not(generated) then
   begin
    tex.GenTexture();
    generated:=true;
   end;

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity;

  glTranslatef(x,y,10);
  glcolor4ub(255,255,255,255);

  glEnable(GL_TEXTURE_2D);

  tex.bind();

  ziffer:=punkte div 10;
  links:=ziffer*1/10;
```

```
  glBegin(GL_QUADS);
   glTexCoord2f( links      ,1  );

     glVertex2f(-zbreite/2 , zhoehe / 2);

   glTexCoord2f(links       ,0);

     glVertex2f(-zbreite/2  , -zhoehe / 2);

   glTexCoord2f(links + 1/10,0  );

     glVertex2f(0 , -zhoehe / 2);

   glTexCoord2f(links + 1/10,1   );

     glVertex2f(0 ,zhoehe / 2);

  glEnd;

  ziffer:=punkte mod 10;
  links:=ziffer*1/10;

  glBegin(GL_QUADS);
   glTexCoord2f( links      ,1  );

     glVertex2f(0 , zhoehe / 2);

   glTexCoord2f(links       ,0);

     glVertex2f(0  , -zhoehe / 2);

   glTexCoord2f(links + 1/12,0  );

     glVertex2f(zbreite/2 , -zhoehe / 2);

   glTexCoord2f(links + 1/12,1   );

     glVertex2f(zbreite/2 ,zhoehe / 2);

  glEnd;



  glDisable(GL_TEXTURE_2D);


end;
procedure TAnzeige.setze(punkte: integer);
begin
 self.punkte:=punkte;
end;

end.
```

## 3.1.13 Objekte\UFigur.pas

```
//
// Name:        UFigur
// Datum:       24.02.2008
// letzte Änderung: 21.03.2008
// Version:          0.1
// Zweck:           Objektmodel Supermario Figur
//
// Programmieren:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//

unit UFigur;

interface
uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,math,forms,

     UGLObject,
     uliste,
     UGegenstand,
     UMuenze,
     UTypenUndKonstanten;

type
     TFigur=class(TGLObject)
      protected
       Tod     : boolean;

       dy:real;


       MyTex : TGLBMP;
       texturnr:TPos;
       richtung:Trichtung;
       v:integer;


       generated:boolean;
       kForm:TForm;

       kgegenstaende,kFiguren,KMuenzen:TListe;
       kollobj:TGLObject;
       obj:TGlObject;

       procedure setTod(const Value: boolean);

       function kollision2erObjekte(obj1,obj2:TGLObject):boolean;
       function
kollision_mit_Objekten(x,y:integer;Liste:TListe):boolean;overload;
       function kollision_mit_Objekten(x,y:integer):boolean;overload;

       procedure steuerung;virtual;abstract;
       procedure nachrechts(n:integer); //n schritte nach
rechts,links,oben,unten
       procedure nachlinks(n:integer);
       procedure nachoben(n:integer);
       procedure nachunten(n:integer);
       procedure bewegen(richtung:Trichtung);virtual;abstract;
```

```
      public
       constructor create(textur:string;Form:TForm);virtual;

       procedure kennt(Liste:Tliste);
       procedure kenntFiguren(Liste:Tliste);
       procedure kenneMuenzen(Muenzen:TListe);
       property nichtlebend:boolean read Tod write setTod;
       destructor destroy;
     end;



implementation

{ TSpieler }


constructor TFigur.create;
begin

 zbreite:=60;
 zhoehe:=90;

 mytex:= TGLBMP.Create(textur);
 mytex.ColorKey(0,0,255, 0);

 kForm:=Form;

 zx:=50;
 zy:=100;
 dy:=0;

 obj:=TGLObject.Create;
 obj.breite:=zbreite;
 obj.hoehe:=zhoehe;


 texturnr.x:=6;
 texturnr.y:=6;
 richtung:=Rstop;
 V:=5;



end;




procedure TFigur.setTod(const Value: boolean);
begin
 tod:=value;

end;

destructor TFigur.destroy;
begin
 obj.Free;
end;
```

```
procedure TFigur.kenneMuenzen(Muenzen: TListe);
begin
 kMuenzen:=Muenzen;
end;

procedure TFigur.kennt(liste:Tliste);
begin
 kgegenstaende:=liste;
end;
procedure TFigur.kenntFiguren(Liste: Tliste);
begin
 kFiguren:=liste;
end;




function TFigur.kollision_mit_Objekten(x, y: integer; Liste: TListe):
boolean;
var i:integer;
    kollision:boolean;
begin
 kollision:=false;
 obj.x:=x;
 obj.y:=y;
 obj.hoehe:=hoehe;
 obj.breite:=breite;

 if assigned(liste)
  then
   begin
     i:=0;
     while (i<liste.getlaenge)and not kollision  do
     begin
      kollobj:=liste.getelement(i) as TGLObject;
      kollision:=kollision or kollision2erObjekte(obj,kollobj);
      inc(i);
     end;
   end;

 if not kollision
  then kollobj:=nil;

 result:=kollision;
end;

function TFigur.kollision_mit_Objekten(x, y: integer): boolean;
begin
 // zuerst wird die Kollision mit meunzen getestet
 // anschließend die mit gegnern und dann die mit gegenständen
 // da bei einer or verknüpfung bei einem true direkt abgebrochen wird
 // und der rest des ausdruckes nicht ausgeführt wird
 result:= kollision_mit_Objekten(x, y,kmuenzen) or
          kollision_mit_Objekten(x, y,kfiguren) or
          kollision_mit_Objekten(x, y,kgegenstaende)  and
          not (kollobj is TMuenze);

end;

procedure TFigur.nachlinks;
```

```
var startx:real;
begin
 startx:=zx;
 while not (kollision_mit_Objekten(round(zx-1), round(zy))) and
       (abs(zx-startx)<n) do
  zx:=zx-1;
end;

procedure TFigur.nachoben;
var starty:real;
begin
 starty:=zy;
 while not (kollision_mit_Objekten(round(zx), round(zy+1))) and
       (abs(zy-starty)<n) do
  zy:=zy+1;
end;

procedure TFigur.nachrechts;
var startx:real;
begin
 startx:=zx;
 while not (kollision_mit_Objekten(round(zx+1), round(zy))) and
       (abs(zx-startx)<n) do
  zx:=zx+1;
end;

procedure TFigur.nachunten;
var starty:real;
begin
 starty:=zy;
 while not (kollision_mit_Objekten(round(zx), round(zy-1))) and
       (abs(zy-starty)<n) do
  zy:=zy-1;
end;

function TFigur.kollision2erObjekte(obj1, obj2: TGLObject): boolean;
var dx,dy:real;
    Shb,Shh:real;
begin
 dx:=abs( obj1.x - obj2.x);
 dy:=abs(obj1.y - obj2.y);
 Shb:=(obj1.breite + obj2.breite) / 2; // Summe halber breiten;
 Shh:=(obj1.hoehe + obj2.hoehe) / 2; // Summe halber hoehen;
 result:= (dx+xtoleranz<= (shb) ) and (dy <=(shh));
end;

end.
```

### 3.1.14 Objekte\UGegenstand.pas

```
//
// Name:        UGegenstand
// Datum:       19.03.2008
// letzte Änderung: 19.03.2008
// Version:       0.1
// Zweck:         ein Gegenstand im Spiel
//
// Programmieren:   Daniel Renner ,Steve Göring
// Kontakt:         daniel26289@yahoo.de,stg7@gmx.de
// Copyright:       2008
//
//
unit UGegenstand;

interface

uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,math,forms,

     UGLObject;

type TGegenstand=class(TGLObject)
      private
       zcrazy:boolean;
       procedure setcrazy(const Value: boolean);
      protected
       Tex : TGLBMP;
       ztexx,ztexy:integer;
       kForm:TForm;
       generated:boolean;
       StartZeit:Word;
       procedure rueckeweiter;
      public
       property breite:integer read zbreite;
       property hoehe:integer read zhoehe;
       property crazy:boolean read zcrazy write setcrazy;
       constructor
create(textur:TGLBMP;Form:TForm;x_,y_,texx,texy:integer);virtual;
       procedure render;virtual;
       procedure weiter;virtual;
       procedure bind;
       function Imbildschirm: boolean;
     end;

implementation

{ TGegenstand }



procedure TGegenstand.bind;
begin
  if not(generated) then
   begin
```

```
    tex.GenTexture();
    generated:=true;
   end;
 tex.Bind;
end;

constructor TGegenstand.create;
begin
 inherited create;

 tex:= textur;
 tex.ColorKey(255,0,255,0);
 kForm:=Form;
 generated:=false;
 zx:=x_;
 zy:=y_;
 ztexx:=texx;
 ztexy:=(31-texy);
 zbreite:=40;
 zhoehe:=40;
 StartZeit:=MilliSecondOf(Time);

end;

function TGegenstand.Imbildschirm: boolean;
begin
 result:= (x>=- zbreite / 2) and  ( x<= 640+ zbreite /2 );
end;

procedure TGegenstand.render;
Var links,oben:real;
begin
  If zx>-zbreite div 2
   Then
    Begin

     if kform.active
      then rueckeweiter;

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity;

     glTranslatef(zx,zy,10);

     if zcrazy
      then glcolor4ub(random(255),random(255),random(255),random(255))
      else glcolor4ub(255,255,255,255);

     glenable(GL_TEXTURE_2D);


     links:=ztexx * 1/32;
     oben:=ztexy*1/32 ;

     glBegin(GL_QUADS);
      glTexCoord2f( links      ,oben+ 1/32  );
      glVertex2f(-zbreite / 2 ,  zhoehe / 2);
      glTexCoord2f(links       ,oben);
      glVertex2f(-zbreite / 2 , -zhoehe / 2);
      glTexCoord2f(links + 1/32,oben  );
      glVertex2f( zbreite / 2  ,-zhoehe / 2);
      glTexCoord2f(links + 1/32,oben+1/32   );
      glVertex2f( zbreite / 2  , zhoehe / 2);
```

```
     glEnd;
     glDisable(GL_TEXTURE_2D);
    End;
end;

procedure TGegenstand.rueckeweiter;
begin
 if (abs(startzeit-millisecondof(time) )>200)
  Then weiter;
end;
procedure TGegenstand.setcrazy(const Value: boolean);
begin
 zcrazy:=value;
end;

procedure TGegenstand.weiter;
begin
 zx:=zx-1;
 Startzeit:=MillisecondOf(time);
end;


end.
```

## 3.1.15 Objekte\UGegner.pas

```
unit UGegner;

interface
Uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,math,forms,

     UFigur,
     UFunktionen,
     UTypenUndKonstanten;

 Type
  TGegner=class(TFigur)
   protected
    fallen:boolean;
    procedure steuerung;override;
    procedure bewegen(richtung:Trichtung);override;
   public
    constructor create(textur:string;Form: TForm;x_,y_:integer);virtual;
    procedure render;override;
    procedure toeten;

 end;
implementation

{ TGegner }

procedure TGegner.bewegen(richtung: Trichtung);
begin
 if richtung=Rrechts
  then nachrechts(v)
  else nachlinks(v+1);
end;

constructor TGegner.create(textur: string; Form: TForm;x_,y_:integer);
begin
  inherited create(Textur,Form);

  y:=y_;
  x:=x_;
  zbreite:=40;
  zhoehe:=40;
  richtung:=Rlinks;
  v:=1;
  fallen:=true;

end;

procedure TGegner.render;
var links:real;
begin

 if not(tod) then
 begin

  // eventuelles Generieren der Textur, dies kann nur
  // geschehen wenn bereits OpengL Initialisiert wurde
  if not(generated) then
   begin
    mytex.GenTexture();
```

```
    generated:=true;
   end;

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity;

  glTranslatef(x,y,10);
  glcolor4ub(255,255,255,255);
  glEnable(GL_TEXTURE_2D);

  mytex.bind();

  links:= texturnr.x*1/6;

  glBegin(GL_QUADS);
   glTexCoord2f(links,1 );

     glVertex2f(-zbreite / 2, zhoehe / 2);

   glTexCoord2f(links,0);

     glVertex2f(-zbreite / 2, -zhoehe / 2);

   glTexCoord2f(links+1/6,0);

     glVertex2f(zbreite / 2, -zhoehe / 2);

   glTexCoord2f(links+1/6,1);

     glVertex2f(zbreite / 2,zhoehe / 2);

  glEnd;
  glDisable(GL_TEXTURE_2D);

  if kform.Active
  then steuerung;
 end;

end;

procedure TGegner.steuerung;
begin
 nachunten(anziehung*2);

  bewegen(richtung);

 if not(kollobj=nil)
  then
   begin
    If Richtung=Rrechts
     Then Richtung:=Rlinks
     Else Richtung:=Rrechts;
    bewegen(Richtung);
   end;

 tod:=y<0;
 if tod
  then y:=-40;

 texturnr.y:=1;
 if richtung=Rlinks
  then texturnr.x:=3
  else texturnr.x:=5;
```

```
 If x<0
  Then Richtung:=Rrechts;

End;
procedure TGegner.toeten;
begin
 tod:=true;
 y:=-hoehe;
 x:=-breite;
end;
end.
```

### 3.1.16 Objekte\UGLObject.pas

```
unit UGLObject;

interface

type TGLObject=class(TObject)
     private

       procedure setx(const Value: real);
       procedure sety(const Value: real);
       procedure setbreite(const Value: integer);
       procedure sethoehe(const Value: integer);
      protected
       zx:real;
       zy:real;
       zbreite,zhoehe:integer;
      public
       procedure render;virtual;abstract;
       property x:real read zx write setx;
       property y:real read zy write sety;
       property breite:integer read zbreite write setbreite;
       property hoehe:integer read zhoehe write sethoehe;
     end;
implementation

{ TGLObject }

procedure TGLObject.setbreite(const Value: integer);
begin
 zbreite:=value;
end;

procedure TGLObject.sethoehe(const Value: integer);
begin
 zhoehe:=value;
end;
procedure TGLObject.setx(const Value: real);
begin
 zx:=value;
end;

procedure TGLObject.sety(const Value: real);
begin
 zy:=value;
end;

end.
```

### 3.1.17 Objekte\UListe.pas

```
unit UListe;

interface

type TListe=class(TObject)
      private
       elemente:array of TObject;
      public
       procedure fuegehinzu(obj:TObject);
       function getelement(i:integer):TObject;
       function getLaenge : integer;
       procedure free;
      end;

implementation
procedure TListe.free;
var
  i: Integer;
begin
  for i := 0 to length(elemente) - 1 do
   elemente[i].Free;
  inherited;
end;

procedure TListe.fuegehinzu;
var neueLaenge:integer;
begin
 neueLaenge:=length(elemente)+1;
 setlength(elemente,neuelaenge);
 elemente[neueLaenge-1]:=obj;
End;

function tliste.getelement(i:integer):TObject;
begin
 if length(elemente)>0
  then result:=elemente[i]
  else result:=nil;
end;

function tliste.getLaenge : integer;
begin
 result:=length(elemente);
end;


end.
```

## 3.1.18 Objekte\UMap.pas

```
unit UMap;

interface
uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,math,forms,

     UGLObject;

type TMap=class(TGLObject)
      private
       kform:Tform;
       MapTex:TGLBMP;
       generated:boolean;

      public
       constructor create(Dateiname:string;form:Tform);virtual;
       procedure render;override;
     end;

implementation

{ TMap }

constructor TMap.create(Dateiname:string;form:Tform);
begin
 kform:=form;
 maptex:= TGLBMP.Create(Dateiname);
end;

procedure TMap.render;
begin

  glMatrixMode(GL_MODELVIEW);

  if not(generated) then
   begin
    maptex.GenTexture();
    generated:=true;
   end;

     glLoadIdentity;
     glEnable(GL_TEXTURE_2D);
     glTranslatef(0,0,0);
     maptex.Bind;

     glBegin(GL_QUADS);
      glTexCoord2f( 0,1  );
       glVertex2f(0, 480);
      glTexCoord2f(0       ,0);
       glVertex2f(0 , 0);
      glTexCoord2f(1,0  );
       glVertex2f(640, 0);
      glTexCoord2f(1,1);
       glVertex2f(640,480);
     glEnd;

 glDisable(GL_TEXTURE_2D);
end;
end.
```

## 3.1.19 Objekte\UMuenze.pas

```
unit UMuenze;

interface

uses GLBMP,Forms,DGLOpenGL,

     Ugegenstand;

type TMuenze= class(Tgegenstand)

      private
       sichtbar:boolean;
      public
       constructor create(textur:TGLBMP;Form:TForm;x_,y_:integer);virtual;
       procedure render;override;
       procedure weiter;override;
       procedure sammeln;
     end;


implementation

{ TMuenze }



constructor TMuenze.create(textur: TGLBMP; Form: TForm; x_, y_: integer);
begin
  inherited create(textur,Form,x_,y_,0,0);
  tex.ColorKey(0,255,0,0);
  sichtbar:=true;
end;

procedure TMuenze.render;
Var links : real;
begin

  If sichtbar and ( zx>-zbreite div 2)
   Then
    Begin

     rueckeweiter;

     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity;

     glTranslatef(zx,zy,10);
     glcolor4ub(255,255,255,255);

     glenable(GL_TEXTURE_2D);

     links:=ztexx * 1/4;

     glBegin(GL_QUADS);
      glTexCoord2f( links      ,1  );
      glVertex2f(-zbreite / 2 ,  zhoehe / 2);
      glTexCoord2f(links       ,0);
      glVertex2f(-zbreite / 2 , -zhoehe / 2);
      glTexCoord2f(links + 1/4,0  );
```

```
       glVertex2f( zbreite / 2  ,-zhoehe / 2);
       glTexCoord2f(links + 1/4,1   );
       glVertex2f( zbreite / 2  , zhoehe / 2);
      glEnd;
      glDisable(GL_TEXTURE_2D);

     End;

end;

procedure TMuenze.sammeln;
begin
 sichtbar:=false;
 zx:=-90;
 zy:=-90;
end;

procedure TMuenze.weiter;
begin
  inherited weiter;
  ztexx:=Ztexx+1;
  if ztexx>3
   then ztexx:=0;

end;

end.
```

## 3.1.20 Objekte\USpieler.pas

```
//
// Name:        USpieler
// Datum:       19.02.2008
// letzte Änderung: 19.02.2008
// Version:          0.1
// Zweck:           Objektmodel Supermario Figur
//
// Programmierer:   Steve Göring,Daniel Renner
// Kontakt:         stg7@gmx.de
// Copyright:       2008
// Änderungen:      ...
// todo:        Eigenschaften des Spielers hinzufügen (Vorarbeit
//              von Daniel in der Klasse UMario!)
//
//

unit USpieler;

interface

uses DGLOpenGL,glBMP,SysUtils,DateUtils,Classes,windows,dialogs,math,forms,

     UMuenze,
     UFunktionen,
     UListe,
     Ugegenstand,
     UMusikImSpiel,
     UFigur,
     UGegner,
     UTypenUndKonstanten;

type
     TPunkteerhoehen=procedure (Punkte:integer) of object;
     TExit=procedure of object;


     TSpieler=class(TFigur)
      private
       starty:real;
       fallen:boolean;
       sprung:boolean;

       kMusik:TMusikImSpiel;
       punkte:integer;

       unverwundbar:boolean;
       start_unverwundbar:integer;
       schritte:integer;
       Groesse : boolean;   //Tja Mario kann klein sein oder Normal oder
groß bzw. könnte bei Groß so ne art special rauskommen oder so :-P Sonst
könnten wir es auch weglassen


       FPunkteerhoehen: TPunkteerhoehen;
       FExit: TExit;


       procedure steuerung;override;
```

```
       procedure tastenauswerten;
       procedure springen;
       procedure texturenermitteln;
       procedure erhoehePunkte;

       procedure auswertenderKollision;
       procedure weiterschieben;

      protected
       procedure bewegen(richtung:TRichtung);override;
      public
       property onExit:TExit read FExit write FExit;
       property onpunkteerhoehen:TPunkteerhoehen read FPunkteerhoehen write
Fpunkteerhoehen;
       constructor create(textur:string;Form:TForm; Musik :
TMusikImSpiel);virtual;
       procedure render;override;

      end;

implementation

{ TSpieler }



procedure TSpieler.auswertenderKollision;
begin
// kollobj is .. Testet ob das Kollisionsobjet eine Instanz der angegebenen
Klasse ist

 if (kollobj is TGegner) and not unverwundbar
  then
   if fallen
    then Begin
          KMusik.MusikAbspielen('GegnerSchlagen');
          (kollobj as Tgegner).toeten;
          bewegen(richtung);
         End
    else begin
          if not groesse
           then tod:=true;
          groesse:=false;
          unverwundbar:=true;
         end;

  if kollobj is TMuenze
   then
    begin
     KMusik.MusikAbspielen('muenze');
     (Kollobj as TMuenze).sammeln;
     erhoehePunkte;
     bewegen(richtung);
    end;

  if kollobj is TGegenstand
   then
    begin
     if (richtung=Rlinks)
      then bewegen( Rrechts );
     if richtung=Rrechts
      then bewegen( Rlinks );
```

```
    end;

end;

procedure TSpieler.bewegen(richtung: TRichtung);
begin
 if richtung=Rrechts
  then nachrechts(v);
 if richtung=Rlinks
  then nachlinks(v);
 if richtung=Roben
  then nachoben(v);
 if richtung=Runten
  then nachunten(v);

end;

constructor TSpieler.create(textur: string; Form: TForm; Musik :
TMusikImSpiel);
begin
 inherited create(Textur,Form);
 kMusik:=Musik;
 groesse:=true;
 schritte:=0;
end;


procedure TSpieler.erhoehePunkte;
begin
 punkte:=punkte+1;
 if assigned(FPunkteerhoehen)
  then FPunkteerhoehen(Punkte);
end;

procedure TSpieler.render;
var links,oben:real;
begin
  // eventuelles Generieren der Textur, dies kann nur
  // geschehen wenn bereits OpengL Initialisiert wurde
  if not(generated) then
   begin
    mytex.GenTexture();
    generated:=true;
   end;

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity;

  glTranslatef(x,y,10);
  if unverwundbar
   then
glcolor4ub(random(55)+200,random(55)+200,random(55)+200,random(255))
   else glcolor4ub(255,255,255,255);


  glEnable(GL_TEXTURE_2D);

  mytex.bind();

  links:=texturnr.x * 1/12;
  oben:=texturnr.y*1/8 ;

  glBegin(GL_QUADS);
```

```
    glTexCoord2f( links      ,oben+ 1/8  );
      glVertex2f(-zbreite / 2, zhoehe / 2);
    glTexCoord2f(links       ,oben);
      glVertex2f(-zbreite / 2 , -zhoehe / 2);
    glTexCoord2f(links + 1/12,oben  );
      glVertex2f(zbreite / 2, -zhoehe / 2);
    glTexCoord2f(links + 1/12,oben+1/8   );
      glVertex2f(zbreite / 2,zhoehe / 2);

  glEnd;

  glDisable(GL_TEXTURE_2D);


  steuerung;
end;

procedure TSpieler.springen;
begin
 if (key_pressed(ord(' '))) and (sprung=false)and (fallen=false)
  then
   begin
    kMusik.MusikAbspielen('Springen');
    starty:=y;
    sprung:=true;
   end;

 if ( abs(y-starty)>=sprunghoehe)
  then
   begin
    fallen:=true;
    sprung:=false;
   end
  else
   if sprung
    then nachoben(10);

 if not(kollobj is Tgegenstand)
  then
   Begin
    If (not sprung or fallen )
     then nachunten(10) ;
   end
  Else fallen:=false;
end;

procedure TSpieler.steuerung;
var
  i,k: Integer;
  tmp:TGegenstand;
begin
 if kform.Active
  then
   begin

    texturnr.y:=6;
    V:=5;
    richtung:=Rstop;

    nachunten(anziehung);

    tastenauswerten;
```

```
    springen;


    if (y<0) or (x<0)
     then Tod:=true;

    bewegen(Richtung);


    auswertenderKollision ;

    weiterschieben;

    nachlinks(1);

    if unverwundbar
     then
      begin
       start_unverwundbar:=start_unverwundbar+1;
       if start_unverwundbar =80
        then
         begin
           start_unverwundbar:=0;
           unverwundbar:=false;
         end;
      end;

     texturenermitteln;

   end;
end;

procedure TSpieler.tastenauswerten;
var Canclose:boolean;
begin
 if key_pressed(VK_LEFT,ord('A'))
  then richtung:=Rlinks;

 if key_pressed(VK_RIGHT,ord('D'))
  then richtung:=Rrechts;

 if key_pressed(VK_DOWN,ord('S'))
  then texturnr.y:=5;

 if key_pressed(VK_UP,ord('W'))
  then texturnr.y:=7;

 if key_pressed(VK_CONTROL)
  then V:=v*2;

 if key_pressed(27)// ESC
  then if assigned(onexit)
        then onexit; // Exit Event auslösen
end;

procedure TSpieler.texturenermitteln;
begin
 if richtung=Rlinks
  then texturnr.y:=4;
 if richtung=Rrechts
  then texturnr.y:=6;
 if richtung=Rstop
  then texturnr.x:=7;
```

```
 If not(groesse)
   Then Texturnr.x:=1
   else Texturnr.x:=7;

 inc(schritte);
 if schritte mod 4 = 0
  then
   begin
    if ord(richtung)mod 2=1
     then texturnr.x:=texturnr.x+1;
    schritte:=1;
    if groesse
     then
      begin
       if texturnr.x>=8
        then texturnr.x:=6
      end
     else
      begin
       if texturnr.x>=3
       then texturnr.x:=1;
      end;
  end;

 if sprung
  then
   begin
    Texturnr.y:=8;
    If (Richtung=Rlinks)
     then
      Begin
       Texturnr.y:=8;
       Texturnr.x:=Texturnr.x + 3;
      End;
   end;
end;
procedure TSpieler.weiterschieben;
var i:integer;
    tmp:Tgegenstand;
begin
 for i := 0 to kgegenstaende.getLaenge- 1 do
    begin
      tmp:=Tgegenstand(kgegenstaende.getelement(i));
      tmp.weiter;
      kollision_mit_Objekten(round(x),round(y),kgegenstaende);
      if kollobj is TGegenstand
       then
        begin
         if (richtung=Rlinks)
          then bewegen( Rrechts );
         if (richtung=Rrechts)  or (richtung=Rstop)
          then bewegen( Rlinks );
        end;
     end;
   for i := 0 to kMuenzen.getLaenge- 1 do
    begin
      tmp:=TMuenze(kMuenzen.getelement(i));
      tmp.weiter;
      auswertenderKollision ;
    end;
end;
end.
```

## 3.1.21 OpenGl\UOpenGlWindow.pas

```
unit UOpenGlWindow;
// Stg7

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs,DateUtils, DGLOpenGL,

  UTypenUndKonstanten;

type
  TFOpenGlWindow = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure IdleHandler(Sender: TObject; var Done: Boolean)  ;
  private
    start_zeit:integer;
    procedure SetupOpenGL;

    procedure ErrorHandler;
  protected

    procedure Render;virtual;abstract;
    procedure nachFormCreate;virtual;abstract;
  public
    DC                                  : HDC;  //Handle auf Zeichenfläche
    RC                                  : HGLRC;//Rendering Context
  end;

var
  FOpenGlWindow: TFOpenGlWindow;

implementation

{$R *.dfm}

procedure TFOpenGlWindow.FormCreate(Sender: TObject);
begin
  DC:= GetDC(Handle);
  if not InitOpenGL then Application.Terminate;
  RC:= CreateRenderingContext( DC,
                               [opDoubleBuffered],
                               32,
                               24,
                               0,0,0,
                               0);
  ActivateRenderingContext(DC, RC);
  SetupOpenGL;
  Application.OnIdle := IdleHandler;

  start_zeit:=millisecondof(time);
  nachFormCreate;
end;

procedure TFOpenGlWindow.SetupOpenGL;
begin
```

```
  glEnable(GL_DEPTH_TEST);          //Tiefentest aktivieren
  glEnable(GL_CULL_FACE);           //Backface Culling aktivieren
  glEnable(GL_TEXTURE_2D); // Texturen aktiviren!


  glEnable(GL_ALPHA_TEST);
  glAlphaFunc(GL_GREATER, 0.1);


  glClearColor( 0.0, 0.0, 0.0, 0.0 );

end;

procedure TFOpenGlWindow.FormResize(Sender: TObject);
var tmpBool : Boolean;
begin
  glViewport(0, 0, ClientWidth, ClientHeight);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity;

  glOrtho(0,640,0,480,-128,128);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity;
  IdleHandler(Sender, tmpBool);
end;

procedure TFOpenGlWindow.FormDestroy(Sender: TObject);
begin
  DeactivateRenderingContext;
  DestroyRenderingContext(RC);
  ReleaseDC(Handle, DC);
end;

procedure TFOpenGlWindow.ErrorHandler;
begin
 self.Caption := self.Caption+' '+gluErrorString(glGetError);

end;
procedure TFOpenGlWindow.IdleHandler(Sender: TObject; var Done: Boolean);
begin

  if (abs(start_zeit-millisecondof(time) )>=25) then
   begin
    start_zeit:=millisecondof(time);

    glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);

    glMatrixMode(GL_PROJECTION);
    glLoadIdentity;
    glViewport(0,0,ClientWidth,ClientHeight);
    glOrtho(0,640,0,480,-128,128); // orthogonale Projektion

    Render;
    SwapBuffers(DC);

  end;
  Done:= false;
end;

end.
```

## 3.1.22 Optionen\UOptionen.pas

```
unit UOptionen;

interface

uses IniFiles,SysUtils ;
// Informationen zu INI Files: http://www.delphi-
treff.de/no_cache/tutorials/datenspeicherung/win32/ini-dateien/page/3/


type TOptionen=class(TObject)
      private
       zHintergrundlautstaerke:integer;
       zGesamtlautstaerke:integer;
       zVollbild:boolean;
       zcrazymodus:boolean;
       zMappack:string;
       filename:string;
       procedure setHintergrundlautstaerke(const Value: integer);
       procedure setGesamtlautstaerke(const Value: integer);
       procedure setvollbild(const Value: boolean);
       procedure setcrazymodus(const Value: boolean);
       procedure setmappack(const Value: string);
      public
       property HintergrundVolume:integer read zHintergrundlautstaerke
write setHintergrundlautstaerke;
       property GesamtVolume:integer read zGesamtlautstaerke write
setGesamtlautstaerke;
       property Vollbild:boolean read zVollbild write setvollbild;
       property crazymodus:boolean read zcrazymodus write setcrazymodus;
       property Mappack:string read zmappack write setmappack;
       procedure loadfromfile(filename:string);
       procedure savetofile(filename:string);
       procedure save;
       procedure load;
       constructor create(filename:string);virtual;
     end;


implementation

{ TOptionen }

constructor TOptionen.create(filename: string);
begin
 self.filename:=filename;
 load;
end;

procedure TOptionen.load;
begin
 loadfromfile(filename);
end;

procedure TOptionen.loadfromfile(filename: string);
var ini: TIniFile;
begin
  ini:=TIniFile.create(ExtractFilePath(ParamStr(0))+filename);
```

```
zHintergrundlautstaerke:=ini.ReadInteger('Sound','Hintergrundlaustärke',0);
  zGesamtlautstaerke:=ini.ReadInteger('Sound','Gesamtlaustärke',0);
  zVollbild:=ini.ReadBool('Grafik','Vollbild',false);
  zCrazyModus:=ini.ReadBool('Grafik','CrazyModus',false);
  zMapPack:=ini.ReadString('Map','MapPack','');
  ini.free;
end;

procedure TOptionen.save;
begin
 savetofile(filename);
end;

procedure TOptionen.savetofile(filename: string);
var ini: TIniFile;
begin
  ini:=TIniFile.create(ExtractFilePath(ParamStr(0))+filename);

  ini.WriteInteger('Sound','Hintergrundlaustärke',zHintergrundlautstaerke);
  ini.WriteInteger('Sound','Gesamtlaustärke',zGesamtlautstaerke);
  ini.WriteBool('Grafik','Vollbild',zVollbild);
  ini.WriteBool('Grafik','CrazyModus',zCrazyModus);
  ini.WriteString('Map','MapPack',zMappack);
  ini.Free;
end;

procedure TOptionen.setcrazymodus(const Value: boolean);
begin
 zcrazymodus:=value;
end;

procedure TOptionen.setGesamtlautstaerke(const Value: integer);
begin
 zGesamtlautstaerke:=value;
end;

procedure TOptionen.setHintergrundlautstaerke(const Value: integer);
begin
 zHintergrundlautstaerke:=value;
end;

procedure TOptionen.setmappack(const Value: string);
begin
 zmappack:=value;
end;

procedure TOptionen.setvollbild(const Value: boolean);
begin
 zvollbild:=value;
end;

end.
```

## 3.1.23 Optionen\UOptionenGrafik.pas

```
unit UOptionenGrafik;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs,StdCtrls,

  UFunktionen,
  UOptionen ;

type
  TFOptionenGrafik = class(TForm)
    CBVollbild: TCheckBox;
    CBCrazymodus: TCheckBox;
    BUebernehmen: TButton;
    BAbbrechen: TButton;
    BReset: TButton;
    procedure BUebernehmenClick(Sender: TObject);
    procedure BResetClick(Sender: TObject);
    procedure BAbbrechenClick(Sender: TObject);
   private
    kOptionen:TOptionen;
   public
    constructor create(aowner:TComponent;Optionen:TOptionen);virtual;
  end;
implementation
{$R *.dfm}
{ TFOptionenGrafik }
procedure TFOptionenGrafik.BAbbrechenClick(Sender: TObject);
begin
 close;
end;
procedure TFOptionenGrafik.BResetClick(Sender: TObject);
begin
 CBcrazymodus.Checked:=koptionen.crazymodus;
 CBVollbild.Checked:=koptionen.vollbild;
end;
procedure TFOptionenGrafik.BUebernehmenClick(Sender: TObject);
begin
 koptionen.crazymodus:=CBcrazymodus.Checked;
 koptionen.vollbild:=CBVollbild.Checked;
 koptionen.save;
 close;
end;
constructor TFOptionenGrafik.create(aowner: TComponent; Optionen:
TOptionen);
begin
 inherited create(aowner);
 koptionen:=optionen;
 zentriereFormular(self);
 CBcrazymodus.Checked:=koptionen.crazymodus;
 CBVollbild.Checked:=koptionen.vollbild;

end;

end.
```

### 3.1.24 Optionen\UOptionenSound.pas

```
unit UOptionenSound;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, ComCtrls,

  UFunktionen,
  UOptionen;

type
  TFOptionenSound = class(TForm)
    TBHintergrund: TTrackBar;
    TBGesamt: TTrackBar;
    LGesamt: TLabel;
    LHintergrund: TLabel;
    BUebernehmen: TButton;
    BAbbrechen: TButton;
    LHWert: TLabel;
    LGWert: TLabel;
    BReset: TButton;
    procedure TBHintergrundChange(Sender: TObject);
    procedure TBGesamtChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BResetClick(Sender: TObject);
    procedure BAbbrechenClick(Sender: TObject);
    procedure BUebernehmenClick(Sender: TObject);
   private
    kOptionen:TOptionen;
   public
    constructor create(aowner:TComponent;Optionen:TOptionen);virtual;
  end;


implementation

{$R *.dfm}

procedure TFOptionenSound.BAbbrechenClick(Sender: TObject);
begin
 close;
end;

procedure TFOptionenSound.BResetClick(Sender: TObject);
begin

 TBGesamt.Position:=koptionen.GesamtVolume;
 TBHintergrund.Position:=koptionen.HintergrundVolume;
 TBGesamtChange(nil);
 TBHintergrundChange(nil);
end;

procedure TFOptionenSound.BUebernehmenClick(Sender: TObject);
begin
 koptionen.GesamtVolume:=TBGesamt.Position;
```

```
 koptionen.HintergrundVolume:=TBHintergrund.Position;
 koptionen.save;
 close;
end;

constructor TFOptionenSound.create(aowner: TComponent; Optionen:
TOptionen);
begin
 inherited create(aowner);
 kOptionen:=Optionen;
 TBGesamt.Position:=koptionen.GesamtVolume;
 TBHintergrund.Position:=koptionen.HintergrundVolume;

end;

procedure TFOptionenSound.FormCreate(Sender: TObject);
begin
 zentriereFormular(self);
end;


procedure TFOptionenSound.TBGesamtChange(Sender: TObject);
begin
 LGwert.caption:= InttoStr( TBGesamt.Position );
end;

procedure TFOptionenSound.TBHintergrundChange(Sender: TObject);
begin
 LHwert.caption:= InttoStr( TBHintergrund.Position );
end;

end.
```

### 3.1.25 USound\usound.pas

```
//
// Name:               USound
// Datum:                      13.01.2008
// letzte Änderung: 20.01.2008
// Version:             0.2
// Zweck:            Soundausgabe von Mp3 Files & Co  , für Supermario
//
// Programmieren:   Steve Göring
// Kontakt:         stg7@gmx.de
// Copyright:       2008
// Änderungen:
//              Kommentierung des Interfaces
//
//

unit USound;
// Sound Ausgabe mittels FMOD!
interface

uses
  fmod,
  fmodtypes,
  fmoderrors,
  Windows,SysUtils;

type TError = procedure of object; // Fehler ereignis
     Tchangestatus=procedure of object;// Statuswechselereignis

     // Klasse zur Soundausgabe
     // unterstüzt werden dabei mp3, ogg, und andere
     // für midi Ähnliche Formate bitte UMusic benutzen
     TSound = class(TObject)
      private
       // Dateiname der abzuspielenden Sound Datei (mp3,ogg ...)
       filename:string;
       // Fehlermeldung
       errorstr:string;
       // Statusmeldung
       status:string;
       // Stream der Sound Datei
       stream: PFSoundStream;
       // Kanal der Sound Datei (siehe FMOD)
       channel:integer;
       // Fehlerereignis
       doerror:TError;
       // Statusänderungsereignis
       changestatus:Tchangestatus;
       // True wenn die Initialierung bereits erfolgt ist
       initialized:boolean;
       // lautstärke
       volume:integer;
       // initialisieren von der Sound Ausgabe
       procedure initialize;
       // Statusmeldung hinzufügen
       // @param st: Status der hinzugefügt werden soll
       procedure addstatus(st:string);
       // Fehlermeldung erzeugen
       // @param e : Fehler
```

```
        procedure error(e:string);
      public
        // OnError Event
        property OnError :TError write doerror;
        // OnChangestatus
        property Onchangestatus :Tchangestatus write changestatus;
        // ermittelt den Status
        function getstatus:string;
        // Kontruktor
        constructor Create;
        // Abspielen
        procedure play;
        // dauerhaftes Abspielen der Musik
        procedure dauerplay;
        // Soundausgabe stoppen
        procedure stop;
        // Soundausgabe pausieren
        procedure pause;
        // Laustärke setzen
        // @param vol: Lautstärke
        procedure setVolume(vol:byte);
        // Datei Laden
        // @param Filename: Dateiname der Soun Datei
        procedure load(Filename:String);
        // Sound ausgabe beenden und Datei schließen
        procedure close;
        // CPU Auslastung ermitteln
        function getcpuusage:string;
        // momentane Position in s
        function getposition:string;
        // Länge der Sounddatei in s
        function getlength:string;
        // Errorstring zurückgeben
        function geterror:string;

        class procedure SetMasterVol(vol:byte);


      end;


implementation



{ TSound }

procedure TSound.close;
begin
 FSOUND_Stream_Close(stream);
 FSOUND_Close();
end;

constructor TSound.Create;
begin
 errorstr:='';
 status:='';
 initialized:=false;
end;

procedure TSound.dauerplay;
begin
```

```
  stream := FSOUND_Stream_Open(PChar(Filename), FSOUND_LOOP_NORMAL, 0, 0);
  if not assigned(stream)
   then error('Fehler beim Laden der Song Datei')
   else addstatus('load '+Filename+' ok' );
end;

function TSound.getcpuusage: string;
begin
  result:=FloatTostr(FSOUND_GetCPUUsage());
end;

function TSound.geterror: string;
begin
   result:=errorstr;
end;

function TSound.getposition: string;
begin
 result:=inttostr(FSOUND_Stream_GetTime(stream) div 1000);
end;

procedure TSound.initialize;
begin
  FSOUND_SetOutput(FSOUND_OUTPUT_WINMM);
  FSOUND_SetDriver(0);  // Standart Soundkarte als Ausgabegerät

  if not FSOUND_Init(44100, 16, 0)
   then error('Fehler bei der Initialisierung')
   else
    begin
     initialized:=true;
     addstatus('initialized') ;
    end;
end;

procedure TSound.load(Filename: String);
begin
  self.filename:=filename;
  if not initialized
   then initialize;

  stream := FSOUND_Stream_Open(PChar(Filename), FSOUND_LOOP_OFF, 0, 0);
  if not assigned(stream)
   then error('Fehler beim Laden der Song Datei')
   else addstatus('load '+Filename+' ok' );

end;

procedure TSound.pause;
begin
 FSOUND_SetPaused(channel, not FSOUND_GetPaused(channel));
 if FSOUND_GetPaused(channel)
  then addstatus('pause')
  else addstatus('start');

end;

procedure TSound.play;

begin

  channel := FSOUND_Stream_Play(FSOUND_FREE, stream);
```

```
  if channel < 0
   then error('Fehler beim Abspielen der Song Datei')
   else addstatus('play') ;
end;

procedure TSound.addstatus(st: string);
begin
 if not(status='')
  then status:=status+','+st
  else status:=st;
 if assigned(changestatus)
  then changestatus;
end;

procedure TSound.stop;
begin
 if FSOUND_Stream_Stop(stream)
  then addstatus('stop');
end;

function TSound.getstatus: string;
begin
 result:=status;
end;

procedure TSound.error(e: string);
begin
 errorstr:=e+' , '+FMOD_ErrorString(FSOUND_GetError()) ;
 FSOUND_Close();
 if Assigned(doError)
  then doError;
end;

procedure TSound.setVolume(vol: byte);
begin
 FSOUND_SetVolume(channel,vol);// FSOUND_SetSFXMasterVolume(vol);
end;

function TSound.getlength: string;
begin
 result:=inttostr(FSOUND_Stream_GetLengthMs(stream)DIV 1000)
end;

class procedure TSound.SetMasterVol(vol:byte);
begin
 FSOUND_SetSFXMasterVolume(vol);
end;


end.
```

## 3.2 LevelEditor

### 3.2.1 Tools\Levelleditor\UHaupt.pas

```
unit UHaupt;
//
// Name:        UHaupt
// Datum:        20.03.2008
// letzte Änderung: 29.03.2008
// Version:          0.8
// Zweck:           Hauptfenster des Editors
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//


interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ExtCtrls, StdCtrls,UTileMap,UTypen, Grids;



type
  TFLevelEditorFutureMario = class(TForm)
    BSpeichern: TButton;
    BTilemap: TButton;
    Efeld: TEdit;
    oeffnen: TOpenDialog;
    speichern: TSaveDialog;
    RGAuswahl: TRadioGroup;
    DGMap: TDrawGrid;
    EspaltenPlus: TEdit;
    bweiterespalten: TButton;
    BOeffnen: TButton;
    procedure FormCreate(Sender: TObject);
    procedure BSpeichernClick(Sender: TObject);
    procedure BTilemapClick(Sender: TObject);
    procedure RGAuswahlClick(Sender: TObject);
    procedure DGMapDrawCell(Sender: TObject; ACol, ARow: Integer; Rect:
TRect;
      State: TGridDrawState);
    procedure DGMapClick(Sender: TObject);
    procedure EspaltenPlusKeyPress(Sender: TObject; var Key: Char);
    procedure bweiterespaltenClick(Sender: TObject);
    procedure BOeffnenClick(Sender: TObject);
  private
    FTilemap:TFTileMap;
    xende:integer;
    yende:integer;
    map:TMap;

    procedure auslesen(Sender: TObject; var Action: TCloseAction);

  public
    { Public-Deklarationen }
  end;
```

```
var
  FLevelEditorFutureMario: TFLevelEditorFutureMario;

implementation


{$R *.dfm}

procedure TFLevelEditorFutureMario.FormCreate(Sender: TObject);
begin
 xende:=16*3;
 yende:=11;

 setlength(map,xende+1,yende+1);
 DGMap.rowcount:=yende;
 DGMap.colcount:=xende;

 FTilemap:=TFTilemap.create(self);
 FTilemap.onclose:=auslesen;

end;
procedure TFLevelEditorFutureMario.RGAuswahlClick(Sender: TObject);
begin
 BTilemap.Visible:=RGAuswahl.ItemIndex=3;
 case RGAuswahl.ItemIndex of
  0:Efeld.Text:='';
  1:Efeld.Text:='gegner';
  2:Efeld.Text:='münze';
  4:Efeld.Text:='map :'+inputbox('Map Wechsel','name der Map ','test.mp');
  3:BTilemap.Click;
 end;

end;







procedure TFLevelEditorFutureMario.auslesen(Sender: TObject; var Action:
TCloseAction);
begin
 Efeld.Text:=FTilemap.Caption;
end;

procedure TFLevelEditorFutureMario.BOeffnenClick(Sender: TObject);
var f:textfile;
    zeile,inhalt:string;
    x,y:integer;

begin
 oeffnen.Execute;
 if oeffnen.FileName<>'' then
 begin
  assignfile(f,oeffnen.FileName);
  reset(f);
  while not eof(f) do
  begin
    readln(f,zeile);
    if not(zeile='')
    then
```

```
     begin
      x:=strtoint(copy(zeile,1,POS(',',zeile)-1));
      DELETE(zeile,1,POS(',',zeile));
      y:=strtoint(copy(zeile,1,POS(',',zeile)-1));
      DELETE(zeile,1,POS(',',zeile));
      inhalt:=zeile;
      if (x>xende)
       then
        begin
         xende:=x;
         setlength(map,x+1,yende+1);
         DGMap.ColCount:=x;
        end;
      map[x,y]:=inhalt;
    end;
  end;

 end;
 DGMap.Repaint;
end;

procedure TFLevelEditorFutureMario.BSpeichernClick(Sender: TObject);
var f:textfile;
    z,i:integer;
begin
 speichern.Execute;
 if speichern.FileName<>'' then
  begin
   if pos('.mp',speichern.FileName)>0
    then assignfile(f,speichern.FileName)
    else assignfile(f,speichern.FileName+'.mp');
   rewrite(f);
   for i:=0 to xende do
    for z:=0 to yende do
     if not(map[i,z]='')
      then writeln(f,inttostr(i)+','+inttostr(z)+','+map[i,z]);
   closeFile(f);
  end;
end;




procedure TFLevelEditorFutureMario.bweiterespaltenClick(Sender: TObject);
begin
 xende:=xende+strtoint(Espaltenplus.Text );
 SetLength(map,xende+1,yende+1 );

 DGMap.Colcount:=xende;
 caption:='spalten: '+inttostr(xende);

end;

procedure TFLevelEditorFutureMario.BTilemapClick(Sender: TObject);
begin
 FTilemap.show;
end;


procedure TFLevelEditorFutureMario.DGMapClick(Sender: TObject);
begin
 map[DGMap.col,DGMap.Row]:=Efeld.Text;
```

```
 DGMap.Repaint;

end;

procedure TFLevelEditorFutureMario.DGMapDrawCell(Sender: TObject; ACol,
ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
 if acol in [0..xende] then

 if map[acol,arow] =''
  then DGMap.Canvas.Brush.Color := clwhite
  else
 if map[acol,arow] ='gegner'
  then DGMap.Canvas.Brush.Color := clRed
  else
 if map[acol,arow] ='münze'
  then DGMap.Canvas.Brush.Color := clyellow
  else
 if POS('map',map[acol,arow]) > 0
  then DGMap.Canvas.Brush.Color := clblack
  else DGMap.Canvas.Brush.Color := clgreen;

 DGMap.Canvas.FillRect(Rect);

 DGMap.Canvas.Brush.Color := clwhite;

end;

procedure TFLevelEditorFutureMario.EspaltenPlusKeyPress(Sender: TObject;
var Key: Char);
begin
 if not (key in ['0'..'9',#8])
  then
   if key=#13
    then Bweiterespalten.click
    else key:=#0;
end;

end.
```

## 3.2.2 Tools\Leveleditor\UTileMap.pas

```
unit UTileMap;
//
// Name:        UTileMap
// Datum:        20.03.2008
// letzte Änderung: 29.03.2008
// Version:          0.8
// Zweck:           Fenster zur Auswahl eines Tiles..
//
// Programmierer:   Daniel Renner, Steve Göring
// Kontakt:         daniel26289@yahoo.de, stg7@gmx.de
// Copyright:       2008
//
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ExtCtrls;

type
  TFTileMap = class(TForm)
    ITile: TImage;
    procedure ITileMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;


implementation

{$R *.dfm}

procedure TFTileMap.ITileMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 caption:=inttostr(x div 32)+' '+inttostr(y div 32);
 close;
end;

end.
```

## 3.2.3 Tools\Leveleditor\Utypen.pas

```
unit Utypen;

interface

type Tmap=array of array of string;
implementation

end.
```

## 4. Testläufe

Das Spiel und auch der Levelleditor wurden bereits in der Entwicklung oft getestet und immer weiter angepasst. Ein paar bekannte Probleme wurden erkannt, aber bisher noch nicht verhindert, so kann es z.B. kommen das man einen Gegner nicht Töten kann, denn dies geht nur wenn man sich im Sprung befindet- auch wenn man sich eventuell wundert, wenn man vom Gegner geschlagen wird und Mario anschließend kleiner wird kann man für kruze Zeit keinen Gegner schlagen, man ist quasi unverwundbar.

Bisher wurden keine weiteren Fehler gefunden.

Die Steuerung funktioniert, Sound wird ausgegeben, Grafik arbeitet auch.

Hinweis: auf älteren Rechnern mit seltenen oder langsamen Grafikkarten kann es durchaus möglich sein, dass Future Mario nicht spielbar ist- da wir OpenGL verwenden benötigt man mindestens eine 3D Grafikkarte mit aktuellem Treiber.