



Komponente Hexgrid für Delphi 5.0



INHALTSVERZEICHNIS

Komponente Hexgrid	4
Private - Funktion	5
Protected - Funktion.....	6
Neue interne Methoden.....	6
Public - Funktion	7
Neue - STREAM - Routinen	7
Allgemeine Routinen die neu gesteuert werden	7
Published - Funktion	8
Ereignisse	10
Komponenten - Fehler.....	11
Beispiel - Anwendung.....	11
Beispiel - Quellcode.....	12



Microsoft, . Windows, Windows NT, Windows 2000, Windows XP sind Marken oder eingetragene Marken der Microsoft Corporation.

Delphi 5.0 sind Produkte der Inprise Corporation.

Die Nennung weitere Namen erfolgt in diesem Handbuch in der Regel ohne Erwähnung bestehender Patente, Gebrauchsmuster oder Warenzeichen. Das Fehlen eines entsprechenden Vermerks begründet nicht die Annahme, die Namen seien frei nutzbar. Alle Warenzeichen werden anerkannt.



Komponente Hexgrid

Die Komponente Hexgrid ist von mir auf den Delphi 5.0 umgesetzt worden. Es gibt zwar hierzu mehrere Varianten, leider konnte keine auch noch zu Editieren. Hexgrid ermöglicht es Hexzahlen in einer Tabelleform darzustellen. Der ANSI - Code wird hier wenn möglich auch dargestellt. Weiter wurde eine bewegliche Hint - Funktion eingebaut, damit man weiß, wo sich die dargestellte Speicherposition befindet. Da es hierbei um eine Ableitung von einer Standard - Grid handelt, sollte diese ohne Problem bei Ihnen funktionieren. Es sind aber weitere Zusatzfunktionen hinzu gekommen die in dem Handbuch beschreiben werden. Alles andere wird dabei von mir ausgeklammert, da dieses in der Hilfe von Delphi ja vorhanden ist.

Die meisten Funktionen sind in der Protected Struktur versteckt und sollten auch nicht sichtbar gemacht werden durch eine Ableitung. Man muss hier eigentlich nichts ändern.

Die Aufschlüsselung der Komponente wird in den nachfolgenden Seiten erklärt.



Private - Funktion

Wie das Wort Private schon ausdrückt, sind hier alle Variablen hinterlegt, die man tunlichst nicht direkt ändern sollte.

FAbout	: THexgridAboutInfo;	About Version
FXShowHint	: boolean;	Soll der normale Hint sichtbar sein ?
FHexHint	: string;	Der Text vom neuen Hint
FShowHexHint	: boolean;	Soll der Hexhint sichtbar sein ?
fScrollHint	: THintWindow;	Die HINT - Struktur
Fstream	: TmemoryStream;	Ist ein Stream, das seine Daten im dynamischen Speicher ablegt.
fStreamOffset	: longint;	Offsetposition innerhalb des Streams
fWorkFillStream,		Hilfsvariable für Fillstream
fWorkMoveStream,		Hilfsvariable für Movestream
fWorkCopyStream	: boolean;	Hilfsvariable für Copystream
Buffer	: tBuffer;	Eigene Ableitung für einen Puffer
FFont	: TFont;	Kapselt eine logische Windows-Schriftart.

Wenn die PROPERTY Variable STREAM gesetzt werden soll, so wird die Zuweisung hier für Hexgrid gemacht

```
procedure SetStream(const Value: TmemoryStream);
```

Scrollfunktion geändert, für die eigenen Zecke.

```
procedure WMVScroll(var Msg: TWMVScroll); message WM_VSCROLL;
```



Protected - Funktion

Die Methode DrawCell stellt den abstrakten Prototyp für eine Methode bereit, die den Inhalt einer Zelle in das Gitter zeichnet.

```
procedure DrawCell(ACol, ARow: Longint; ARect: TRect; AState: TGridDrawState); override;
```

Die Methode CanEditModify zeigt an, ob die aktuelle Zelle des Gitters bearbeitet werden kann.

```
function CanEditModify: Boolean; override;
```

Mit der Methode CreateEditor können Sie den Zelleditor erstellen.

```
function CreateEditor: TInplaceEdit; override;
```

Neue interne Methoden

Den Font ändern, der jetzt neu festlegt wurde

```
procedure FontChange(Sender: TObject);
```

Den neuen ausgewählten Font festlegen

```
procedure SetFFont( Value : TFont );
```

Den Text für den Zusatzhint festlegen

```
procedure SetFHexhint( Value : String );
```

Festlegen, ob der neue Hint sichtbar gemacht werden soll.

```
procedure SetFShowHexHint( Value : boolean );
```



Public - Funktion

Hier sind alle Funktionen und Variablen die für die Öffentlichkeit zugänglich sind.

Die Methode erzeugt und initialisiert ein THexgrid-Objekt.

constructor create(aOwner :tComponent); override;

Die Methode zerstört ein THexgrid-Objekt.

destructor destroy; override;

Neue - STREAM - Routinen

Eigenschaft Stream ermöglicht es, direkt Streamdaten einzubinden.

property Stream : tmemoryStream read FStream write SetStream;

Eigenschaft Soffset ermöglicht es, die Aktuelle Position festzulegen

property Soffset : longint read FStreamOffset write FStreamOffset default 0;

Den Gesamten Stream löschen

function ClearStream : boolean;

Den Stream von einer Start- bis Endposition mit einem Byte füllen

function FillStream (startpos, endpos : longint; Value : Byte): boolean;

Innerhalb des Streams Daten bewegen

function MoveStream (startpos, endpos, destpos : longint): boolean;

Innerhalb des Streams Daten kopieren

function CopyStream (startpos, endpos, destpos : longint): boolean;

Innerhalb des Streams ein bestimmtes Byte zurück geben lassen.

function Getstreambyte(startpos : longint) : byte;

Innerhalb des Streams ein bestimmtes Byte übergeben.

procedure Setstreambyte (startpos : longint; AValue : byte);

Direkte Kontrolle über den Streamdaten erhalten.

property streambyte[Value : longint]: byte read Getstreambyte write Setstreambyte;

Allgemeine Routinen die neu gesteuert werden

Die Methode CanEditAcceptKey gibt an, ob der Inhalt des Parameters Key in die aktuelle Zelle eingegeben werden kann.

function CanEditAcceptKey (Key : Char): boolean; override;

Die Methode GetEditText gibt den für die Bearbeitung formatierten Wert der angegebenen Zelle zurück.

function GetEditText(ACol, ARow: Longint): String; override;

Die Methode GetEditMask stellt eine Schnittstelle für eine Methode bereit, die eine Eingabemaske für die angegebene Zelle zurückgibt.

function GetEditMask(ACol, ARow: Longint): String; override;

Die Methode SetEditText stellt eine Schnittstelle für eine Methode bereit, die den mit einer Zelle verbundenen Text aktualisiert.

procedure SetEditText(ACol, ARow: Longint; const Value: string); override;



Public - Funktion

Hier sind weitere Funktionen und Variablen die für die Öffentlichkeit zugänglich sind.

Die Methode SetGotostream stellt eine Schnittstelle für die Methode Gotostream bereit, die das Festlegen einer Adressposition ermöglicht. Und es auch im Gitter dann anzeigt.

```
procedure SetGotostream( avalue : longint );
```

Eigenschaft Gotostream ermöglicht es, eine bestimmte Adressposition festzulegen

```
property Gotostream : longint read fGotoStream write SetGotostream;
```

Published - Funktion

Endlich kommt der Objektinspektor zum Zug. Alles was sichtbar sein soll, in der Komponente wird hier angezeigt.

Version von der Komponente

```
property About : THexgridAboutInfo read FAbout write FAbout;
```

Die Eigenschaft Align bestimmt, wie das Steuerelement innerhalb eines Containers (übergeordneten Komponente) ausgerichtet wird.

```
property Align;
```

Die Eigenschaft Anchors legt fest, wie das Steuerelement in seinem übergeordneten Objekt verankert wird.

```
property Anchors;
```

Die Eigenschaft Cursor bestimmt, welche Grafik für den Mauszeiger innerhalb des Steuerelements angezeigt wird.

```
property Cursor;
```

Die Eigenschaft Color gibt die Hintergrundfarbe des Steuerelements an.

```
property Color;
```

Die Eigenschaft DragCursor gibt an welche Grafik beim Ziehen des Steuerelements für den Mauszeiger verwendet wird.

```
property DragCursor;
```

Die Eigenschaft DragKind gibt an, ob das Steuerelement auf die übliche Weise gezogen oder zum Andocken verschoben wird.

```
property DragKind;
```

Die Eigenschaft DragMode bestimmt, wie das Steuerelement Drag&Drop- oder Drag&Dock-Operationen beginnt.

```
property DragMode;
```

Die Eigenschaft Enabled steuert, ob das Steuerelement auf Maus-, Tastatur- und Timer-Ereignisse reagiert.

```
property Enabled;
```

Die Eigenschaft FixedColor gibt die Hintergrundfarbe der fixierten Zeilen und Spalten des Gitters an.

```
property FixedColor;
```

Font festlegen für die Ausgabe von Hexgrid

```
property Font read FFont write SetFFont stored True;
```




Die Eigenschaft `GridLineWidth` gibt die Breite der Linien in Pixel an, die die Zellen des Gitters trennen.

property `GridLineWidth`;

Die Eigenschaft `Hexhint` enthält den String, der angezeigt wird, wenn sich der Mauszeiger über dem Scroll-Steuerelement befindet.

property `Hexhint` : string read `FHexhint` write `SetFHexhint`;

Die Eigenschaft `Hint` enthält den String, der angezeigt wird, wenn sich der Mauszeiger über dem Steuerelement befindet.

property `Hint`;

Die Eigenschaft `InplaceEditor` bezeichnet das Steuerelement, mit dem der Inhalt der ausgewählten Zelle bearbeitet werden kann.

property `InplaceEditor`;

Die Eigenschaft `Name` gibt den Namen an, mit dem im Programmcode auf das Steuerelement verwiesen wird.

property `Name`;

Mit der Eigenschaft `Options` können Sie die verschiedenen Eigenschaften zur Anzeige und zum Verhalten des Gitters festlegen.

property `Options`;

Die Eigenschaft `ParentFont` legt fest, woher ein Steuerelement seine Schriftinformationen bezieht.
Wird von mir nicht benutzt.

property `ParentFont`;

Die Eigenschaft `PopupMenu` gibt das Popup-Menü des Steuerelements an.

property `PopupMenu`;

Mit der Eigenschaft `ScrollBars` können Sie festlegen, ob das Gitter eine horizontale und eine vertikale Bildlaufleiste enthält.

property `ScrollBars`;

Die Eigenschaft `ShowHint` bestimmt, ob das Steuerelement einen Kurzhinweis anzeigt.

property `ShowHint`;

Die Eigenschaft `ShowHexHint` bestimmt, ob das Scroll-Steuerelement einen Kurzhinweis anzeigt.

property `ShowHexHint` : boolean read `FShowHexHint` write `SetFShowHexHint` default false;

Die Eigenschaft `TabOrder` gibt die Position eines Steuerelements in der Tabulatorreihenfolge seines übergeordneten Fensters an.

property `TabOrder`;

Die Eigenschaft `TabStop` bestimmt, ob der Benutzer mit der Tabulatortaste zu einem Steuerelement wechseln kann.

property `TabStop`;

Die Eigenschaft `Tag` speichert eine Ganzzahl als Teil der Komponente.

property `Tag`;

Die Eigenschaft `Visible` bestimmt, ob die Komponente auf dem Bildschirm angezeigt wird.

property `Visible`;



Ereignisse

Das Ereignis `OnClick` wird ausgelöst, wenn der Benutzer auf das Steuerelement klickt.

property `OnClick`;

Das Ereignis `OnEnter` wird ausgelöst, sobald ein Steuerelement den Eingabefokus erhält.

property `OnEnter`;

Das Ereignis `OnExit` wird ausgelöst, sobald der Fokus von einem Steuerelement zu einem anderen wechselt.

property `OnExit`;

Das Ereignis `OnKeyDown` wird ausgelöst, sobald der Benutzer eine Taste drückt.

property `OnKeyDown`;

Das Ereignis `OnKeyPress` wird ausgelöst, sobald der Benutzer eine einzelne Zeichentaste drückt.

property `OnKeyPress`;

Das Ereignis `OnKeyUp` wird ausgelöst, sobald der Benutzer eine gedrückte Taste los lässt.

property `OnKeyUp`;

Das Ereignis `OnMouseDown` wird ausgelöst, wenn der Benutzer über einem Steuerelement eine Maustaste drückt.

property `OnMouseDown`;

Das Ereignis `OnMouseMove` tritt auf, wenn der Benutzer den Mauszeiger verschiebt, während sich dieser über einem Steuerelement befindet.

property `OnMouseMove`;

Das Ereignis `OnMouseUp` tritt ein, wenn der Benutzer eine Maustaste los lässt, die er gedrückt hat, während sich der Mauszeiger über einem Steuerelement befand.

property `OnMouseUp`;

Das Ereignis `OnMouseWheelDown` wird ausgelöst, wenn das Mäusrad nach unten gedreht wird.

property `OnMouseWheelDown`;

Das Ereignis `OnMouseWheelUp` wird ausgelöst, wenn das Mäusrad nach oben gedreht wird.

property `OnMouseWheelUp`;



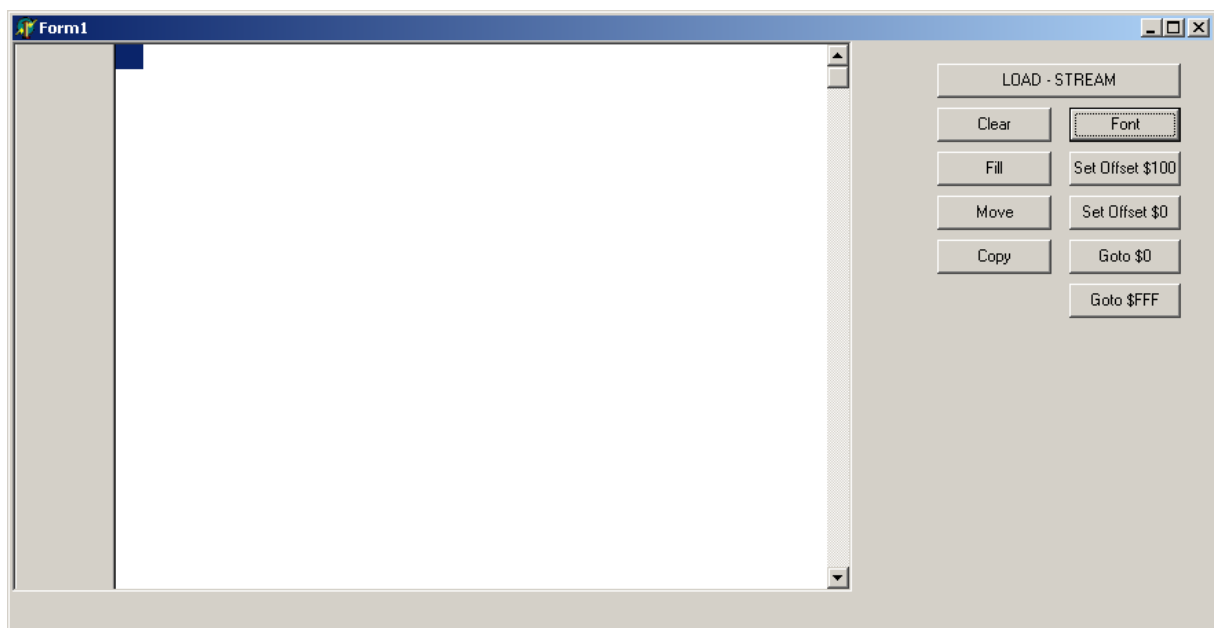
Komponenten - Fehler

Standard Fehlerarten können bei der Verwendung der Komponente auftreten.

Beispiel - Anwendung

Eine Beispielanwendung ist der Komponente beigelegt und sollte die grundsätzlichen Funktionen erklären.

Es wurden eine Hexgrid - Komponente und ein Paar Buttons verwendet.



Die neuen Funktionen (Gotostream) sind im Quellcode nicht beschrieben.

Das von mir geschriebene Beispiel, ist eine klassische Komponenten - Testumgebung. Wenn die Komponente in der Komponentenleiste eingebunden ist, so muss man die SELF - Variante nicht machen.

Sie auf den folgenden Seiten das Beispiel.

Und viel Spaß beim ausprobieren.

Ich übernehme keine Haftung bei der Fehlerhaften Bedienung der Komponente. Sollte es durch die Komponente, einen gravieren Schaden auf Ihren PC geben, so kann ich auch hier nicht Haften, da nicht ausgeschlossen werden kann, das andere Programme die Sie verwenden, hierfür die Ursache sind.



Beispiel - Quellcode

nit Kompo;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Hexgrid, ExtCtrls, Grids;

type

```
TForm1 = class(TForm)
  OpenFileDialog1: TOpenDialog;
  SaveDialog1: TSaveDialog;
  Button1: TButton;
  FontDialog1: TFontDialog;
  Button3: TButton;
  Button2: TButton;
  Button4: TButton;
  Button5: TButton;
  Button6: TButton;
  procedure FormCreate(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
  procedure Button5Click(Sender: TObject);
  procedure Button6Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
```

private

```
{ Private-Deklarationen }
  mem : Tmemorystream;
```

public

```
{ Public-Deklarationen }
end;
```

var

```
Form1: TForm1;
```

```
TK: THexgrid;
```

implementation

```
{ $R *.DFM }
```



```
procedure TForm1.FormCreate(Sender: TObject);
begin
    mem := tmemorystream.create;
    mem.size := $10;

    tk := Thexgrid.create(self);
    tk.parent := self;
    tk.height := 400;
    tk.width := 610;
    tk.Visible := true;
    tk.font.name := 'Courier New';
    tk.font.Size := 10;
    tk.hint := 'Test';
    tk.hexhint := 'Hexzeile';
    tk.showhexhint := true;
    tk.showhint := false;
    tk.options := [goFixedVertLine, goRangeSelect, goRowMoving, goColMoving, goEditing];
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    tk.free;
    mem.free;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
    F: Textfile;
begin
    OpenFileDialog1.Title := 'Datei - STREAM laden';
    if OpenFileDialog1.Execute then
        begin
            AssignFile(F, OpenFileDialog1.FileName);
            try
                Reset(F);
                mem.size := FileSize(f);
                CloseFile(F);
                mem.loadfromfile(OpenDialog1.FileName);
                tk.Stream := mem;
            except
                on EInOutError do
                    MessageDlg('Datei-E/A-Fehler', mtError, [mbOk], 0);
            end;
        end;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    tk.ClearStream;
end;

procedure TForm1.Button4Click(Sender: TObject);
var
    x : integer;
begin
    for x:= 0 to 255 do tk.streambyte[longint(x)] := x;
    repaint;
end;
```



```
procedure TForm1.Button5Click(Sender: TObject);
begin
    tk.MoveStream( 0, $1f, $53 );
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
    tk.copystream( 0, $1f, $d1 );
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
    if (FontDialog1.execute) then
        begin
            tk.font := FontDialog1.font;
            repaint;
        end;
end;

end.
```