HexMap Component for Delphi - Version 1.2

Properties Methods

Description:

The HexMap component for Delphi was developed by Patrick Kemp and Bob Dalton to make it *much* easier for Delphi programmers and developers to design game applications which use a Hex Grid map.

While you will notice that there are many properties available for your use there are only a few methods. Why?

The component has been designed to be almost totally automatic in it's functioning once the correct properties are set, so only a few public methods are required or available for your use.

No events are listed as they are all normal Delphi events and none are special to this component.

Only properties unique to this component are shown in this help file. All the rest are stock Delphi component properties.

Please pardon the size of this help file but I believe that a picture explains things so much easier and that's where the size is coming from - the bitmaps!

INSTALLATION OF THE COMPONENT:
(NOTE! See your Delphi documentation for detailed Help on installing components)

#1 Place Hexmap.pas and Hexmap.dcr into the appropriate Delphi 16 or 32 bit compiler sub-directory where you normally place your component sources (ie; usually the “Lib” sub-directory).

#2 Select <Options>-<Install Components> from the menu.

#3 Select <Add> and then hit the <browse> button to find and enter the path to the hexmap component.

#4 Select <OK> to accept then <OK> to rebuild library.

That’s it!

How To Reach Us

Patrick Kemp can be reached by email at: `Cynewulf@qadas.com`

Inquiries or comments about the HexMap component should be directed to the person below:

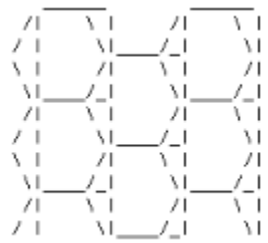
Bob Dalton
Developer Relations Manager
Mustang Software Inc. (at www.mustang.com)
bob.dalton@mustang.com

Comments and Notes on the Component

The demo's executable is included so there is no need to compile it from the source. However, we have included the source in case you decide you want to go that route.

NOTE from Patrick Kemp: The Hexagons drawn by this unit are NOT perfect Hexagons. When I originally designed this, I used graph paper to draw the Hexagons and Hexgrids in order to see the relationships. Because of the graph paper, the Hexagons were elongated slightly. This didn't bother me all that much as it made the Math easier.

Hit testing is done similar to the technique of finding a mouse hit within a grid (dividing mouse x/y by number of rows/cols - except in this case every other col is lower by half a hex .) Basically, it's like overlaying each hex with a rectangle....



I added a 1/8 hex offset to the x in order to center the rects better over each hex. There is still some places where the mouse will return a wrong row/col but these are only at the extreme left/right corners. I feel that the error is small enough not to be an issue as most hits will be close to a hexes center. Take a look at the demo program to see how insignificant the problem is...

Known problems:

#1 A problem changing font attributes. To see this, run the demo and change the font. You will notice that nothing happens until you change some other property of the hex grid... We are working on this. If anyone has any Ideas or a fix...PLEASE let us know!

#2 A slight screen flicker is present when the map is scrolled. If anyone has ideas on how to eliminate it we would like to hear about it.

#3 Drawing a Delphi brush pattern in an individual hex does not work for some reason. If anyone has ideas on how to fix this we would like to hear about it.

Future plans for the component:

#1 Add built in sprite routines (add a sprite, move a sprite, delete a sprite, sprite collided) to this component. This would probably be based on an array which would support up to 200 or more sprites. The array would hold the sprite ID number and sprite.bmp name or alternately hold the x/y location for a routine to do a copyrect from a single large bitmap to take out a “cookie cutter” bite of the image or even both methods. If anyone would like to help us out it would be appreciated.

#2 Add support for tiling background maps to decrease memory requirements, especially in 16 bit windows. Again if anyone would like to help us out it would be appreciated.

Remember: We all benefit when we share our knowledge and expertise!!

Files that should be in the HexMap Archive

In the HexMap archive package you should find:

⇒ Hex16.zip - 16 bit version of the component with source code.
⇒ Hex32.zip - 32 bit version of the component with source code.
⇒ File_id.diz - standard file description
⇒ hexgrid.dsk - part of the demo project.
⇒ hexgrid.dpr - part of the demo project.
⇒ hexgrid.exe - the compiled version of the demo showing how the component works.
⇒ hexgrid.opt - part of the demo project.
⇒ hexgrid.res - part of the demo project.
⇒ Rome.bmp - sample map background image used by the demo and component.
⇒ unit1.dfm - part of the demo project.
⇒ unit1.pas - part of the demo project.
⇒ This help file.

Terms of Use

This component is being released as freeware. If you find it useful all we ask is you let us know. If you modify it we ask you share any changes with us.

Version History and Changes

Version Changes and Comments

1.0 - First release by Patrick Kemp.
1.1 - The following changes were made (by Bob Dalton):
#1 Substantially revised the component to allow for an optional background map image on the component canvas.
#2 Revised the Demo app to show all changes to date.
1.2 - The following changes were made:
#1 Added a property to let you turn on or off the display of the hex grid (Bob Dalton).
#2 Reorganized the properties and renamed a few to make them clearer (Bob Dalton).
#3 Added a windows help file (Bob Dalton).
#5 Updated the Demo to show all changes to date(Bob Dalton).
#6 Eliminated the grid offset (Patrick Kemp).
#7 Added the UseSavedMap property(Bob Dalton). This allows for pre-drawn hex maps to be used by the component.
#8 Added a SaveMapHex method(Bob Dalton). This allows you to save a map you create or overlay a hex grid on.
#9 Added a LoadHexMap method(Bob Dalton). This allows you to load and display a pre-drawn map.
#10 Added a PaintAHex method(Bob Dalton). This allows individual hexes to be drawn in different colors if you desire to do that.
#11 Added a RangeInHexes method(Bob Dalton). This allows you to find the range in hexes from a starting hex to an ending hex.

Properties

MapGridOn	MapImageName	NoMapImage
HexColumns	HexRows	HexRadius
HexShowLabels	Hex3d	HexColor
BackColor	LineColor	UseSavedMap

Methods

ConvertCoords
SaveHexMap
LoadHexMap
PaintAHex
RangeInHexes

MapGridOn Property

Declaration
Property MapGridOn : Boolean;

Description
If True then the Hex Grid is displayed. Like this:

If false then the Hex Grid is **not** displayed. The picture above would now look like:

The default setting is: True.

Note: Even though the grid is not displayed the coordinate system is still there.

MapImageName

Declaration
Property MapImageName : String;

Description
The name of the bitmap you want to use as a background on the component canvas.
Example: rome.bmp

HexRows

Declaration
Property HexRows : Integer;

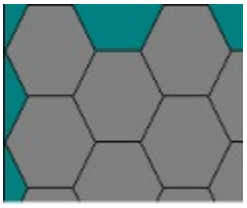
Description
The number of hex rows you want expressed as a integer.

Hex3d

Declaration
Property Hex3d : Boolean;

Description
If True then the Hex Grid are shown in a 3d format. Like this:

If false then the Hex Grid is **not** in shown in 2d format. Like this:



The default setting is: True.

UseSavedMap

Declaration
Property UseSavedMap: Boolean;

Description:
If true you are using a pre-drawn map.
If false you are not using a pre-drawn map.
Default is: False

NoMapImage

Declaration
Property NoMapImage : Boolean;

Description
If True then the a background bitmap image is **not** is displayed on the component canvas.
If false then a background bitmap image **is** displayed on the component canvas. Default is True.

Note: If set to **true** then the display routine will automatically draw on the image the number of hex columns and rows that will fit on that particular bitmap size based on the properties involved (ie; HexColums, HexRows, HexRadius).

HexRadius

Declaration
Property HexRadius : Integer;

Description
The size you want your hexes to be expressed as an integer. Even numbers must be used. The default is 30.

HexColor

Declaration
Property HexColor : `TColor`;

Description
The color you want your solid hexes to be. Will not work with background images.

LineColor

Declaration
Property LineColor : `TColor`;

Description
The color you want your hexes to be drawn in.

SaveHexMap

Declaration
Procedure SaveHexMap(MapFileName : String);

The parameters are:

MapFileName = Name of the file you want to save provided in string format.

Success is assumed.

Description
This function saves a created hex map in the current directory and creates an INI file called HexMap.Ini in your Windows directory to store the information on the saved HexMap so you won't have to try and remember what the critical stats were.

Important Notes: Since these bitmaps are not DIBs be sure to change your screen resolution to 256 color or lower prior to editing a grid map and saving it or the image file size will be way to large!

HexColumns

Declaration
Property HexColumns : Integer;

Description
The number of hex columns you want expressed as an integer.

HexShowLabels

Declaration
Property HexShowLabels : Boolean;

Description
If true you will display numerical labels for each hex, if false you won't. The default is: False.

With Labels on it would look like:

With labels off the same image would look like:

BackColor

Declaration
Property BackColor : `TColor`;

Description
The color you want your background canvas to be. This will **not** work with background images on the canvas.

ConvertCoords

Declaration
Function ConvertCoords(point:Tpoint;pointtype:Tpointtype):Tpoint;

The parameters are:

point = pair to be converted.
Pointtype = type to be converted (either ptXy or ptRowCol) as follows:

```
ptXY : {Converts from x/y and returns value as Row/Col}

or

ptRowCol:  { Converts from Row/Col and returns value as X/Y }
```

Description
Use this to find the coordinates of where the mouse pointer is. See the demo application for an example of how this is used and see the demo source code for how it is implemented.

Important note: Offmap areas (ie; spaces other then the hexes) will show a -1 result returned for the col/row coordinates.

RangeInHexes

Declaration
function RangeInHexes(Bpoint:Tpoint;Epoint:TPoint):Integer;

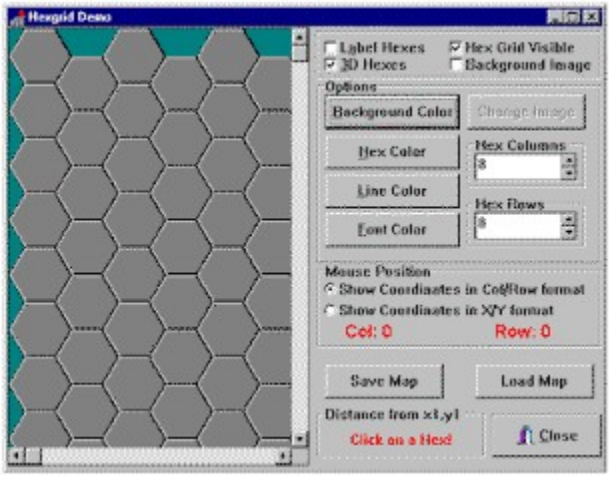
The parameters are:

Bpoint = beginning points in type Tpoint format.
Epoint = end point in type Tpoint format.

Returns an integer value of the number of hexes from Bpoint to Epoint.

Description
You give it the beginning hex point in x,y coordinates (Bpoint) and the ending hex point x,y coordinates (Epoint) and it returns the distance in hexes from the beginning hex point.

The Demo Application



Here is a screen shot from the HexGrid demonstration application.

To launch the HexGrid demo just double click on HexGrid.exe. Enjoy!

LoadHexMap

Declaration
Function LoadHexMap(MapFileName : String):Boolean;

The parameters are:

MapFileName = Name of the file you want to save provided in string format.

If you were successful the function returns true otherwise false.

Description
This function loads a previously created hex map in the current directory.

Notes:
This will allow you to work with saved images already possessing a hexgrid instead of dynamically creating them each time if you want that option. This allows for a pre-drawn gridmap image but still gives you the coordinate system assuming you have set the HexColumn, HexRow and HexRadius properties correctly to match your image….

PaintAHex

Declaration
procedure PaintAHex(HexColorWanted : Tcolor; HexPatternWanted: TBrushStyle; MapLocation:TPoint);

The parameters are:

HexColorWanted = color of type TColor.
HexPatternWanted = style of type TBrushStyle.
MapLocation = hex x, y coordinates of type Tpoint.

Description
This procedure allows you to paint an individual hex on a non-image background hex map.