.lib fileformat specifications

by Philipp Pammler

1. .lib (.ter)

.lib is just a collection of files:

a library file looks like this

```
.lib
{
	headerinformation (major-, minorversion, name of the creator)
	directoryitems (filename, description, filetype)
	{
		item1
		item2
		item3
		item4
		...
	}
}
```

2. cell specification(s)

TCell = array[0..132, 0..132] of single; // 132 = CellSize(128) + 4

each item in the .terrain file is one cell

```
.ter
{
	headerinfo
	directory
	{
		terrain/mapname-cellindexX-cellindexZ
		...
	}
}
```

e.g. “montagne.ter” it consists of 4 cells

```
...
directory
{
	montagne-0-0
	montagne-1-0
	montagne-0-1
	montagne-1-1
}
```

...

3. TerrainRenderer & CustomHeightDataHDS + .StartPreparingData

```
const
  DEF_CellSize = 128;

type
  TCell = array[0..DEF_CellSize + 4, 0..DEF_CellSize + 4] of single;

var
  VAR_HDS: TCell;
  VAR_Mapname: string;
  VAR_CellNames: Tstringlist;
  INT_FileSystem: TFileSystem; // libFileSystemU.pas
  DIR_Data: string; // = ExtractFilePath(Application.ExeName) + 'data\';

{
  the map is loaded with on the program startup or everywhere else
  INT_FileSystem := TFileSystem.Create(DIR_Data + 'heightdata\VAR_Mapname + '.ter', False);
  VAR_CellNames := Tstringlist.Create;
  INT_FileSystem.GetFileList(VAR_CellNames);

  The terraindata is stored in \data\heightdata\*.ter ... so VAR_Mapname is just the filename witout
pathinfo and without its extension '.ter'
}

procedure PrepareHDSData(Mapname : string; X, Y : integer);
var
  CellItem : string;
  TempCell : TCell;
  iX       : integer;
  iY       : integer;
  MS       : TMemoryStream;

begin
  CellItem := Format('%s-%d-%d', [VAR_Mapname, Y, X]);
  if VAR_CellNames.IndexOf(CellItem) > -1 then begin
    MS := TMemorystream.Create;
    INT_FileSystem.ItemToStream(VAR_CellNames.IndexOf(CellItem), MS);
    TempCell := TCell(MS.Memory^);
    FreeAndNil(MS);
    for iY := 0 to DEF_CellSize do
      for iX := 0 to DEF_CellSize do
        VAR_HDS[iX, iY] := TempCell[iY, iX];
  end
  else
    ZeroMemory(@VAR_HDS, SizeOf(TCell));
end;

procedure T<xyz>.GLCustomHDS.StartPreparingData;
var
  Y          : integer;
  X          : integer;
  RasterLine : GLHeightData.PSingleArray;
  OldType    : THeightDataType;

begin
  HeightData.DataState := hdsPreparing;
  with HeightData do begin
    OldType := DataType;
    Allocate(hdtSingle);
//    HeightData.MaterialName := 'default'; dunno if we can get the texturing stuff working on a
terrainrenderer
    PrepareHDSData(VAR_Mapname, (XLeft div (DEF_CellSize - 1)), (YTop div (DEF_CellSize - 1)));
    for Y := YTop to YTop + Size - 1 do begin
      RasterLine := SingleRaster[Y - YTop];
      for X := XLeft to XLeft + Size - 1 do
        RasterLine[X - XLeft] := VAR_HDS[X - XLeft, Y - YTop] / 2457;
    end;
    if OldType <> hdtSingle then
      DataType := OldType;
  end;
  inherited;
end;
```