How to create a TDrawGrid where all cells act as buttons

Question

Is there anybody who knows how to subclass the existing *TDrawGrid* so that all the cells act as buttons? I would like *OnDrawCell* to return the inner rectangle of the button look set the colors of the bevel so that they look like a button.

```
unit ButtonDrawGrid;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids;

type
  TPBButtonDrawGrid = class(TDrawGrid)
  private
    FCellDown: TGridCoord;
  protected
    procedure DrawCell(ACol, ARow: Longint; ARect: TRect;
      AState: TGridDrawState); override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
      override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    function SelectCell(ACol, ARow: Longint): Boolean; override;
  public
    constructor Create( aOwner: TComponent ); override;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('PBGoodies', [TPBButtonDrawGrid]);
end;

{ TButtonDrawGrid }

constructor TPBButtonDrawGrid.Create(aOwner: TComponent);
begin
  inherited;
  FCellDown.X := - 1;
  FCellDown.Y := - 1;
end;

procedure TPBButtonDrawGrid.DrawCell(ACol, ARow: Integer; ARect: TRect;
AState: TGridDrawState);
var
  r: TRect;
  style: DWORD;
begin
  r := ARect;
  if not (gdFixed In aState) then
  begin
    Canvas.Brush.Color := clBtnFace;
    Canvas.Font.Color := clBtnText;
    style := DFCS_BUTTONPUSH or DFCS_ADJUSTRECT;
    if (FCellDown.X = aCol) and (FCellDown.Y = aRow ) then
      style := style or DFCS_PUSHED;
    DrawFrameControl( Canvas.Handle, r, DFC_BUTTON, style );
  end;
  inherited DrawCell( ACol, aRow, r, aState );
end;

procedure TPBButtonDrawGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
var
  cell: TGridCoord;
begin
  if (Button = mbLeft) and ((Shift - [ssLeft]) = []) then
  begin
    MousetoCell( X, Y, cell.X, cell.Y );
    if (cell.X >= FixedCols) and (cell.Y >= FixedRows) then
    begin
      FCellDown := cell;
      InvalidateCell( cell.X, cell.Y );
    end;
  end;
  inherited;
end;

procedure TPBButtonDrawGrid.MouseMove(Shift: TShiftState; X, Y: Integer);
var
  cell: TGridCoord;
begin
  if Shift = [ssLeft] then
  begin
    MousetoCell( X, Y, cell.X, cell.Y );
    if not CompareMem( @cell, @FCellDown, Sizeof(cell)) then
    begin
      if ( FCellDown.X >= 0) and (FCellDown.Y >= 0) then
        InvalidateCell( FCellDown.X, FCellDown.Y );
      FCellDown := cell;
      InvalidateCell( cell.X, cell.Y );
    end;
  end;
  inherited;
end;

procedure TPBButtonDrawGrid.MouseUp(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
begin
  if (Button = mbLeft) and (Shift = []) then
  begin
    InvalidateCell( FCellDown.X, FCellDown.Y );
    FCellDown.X := - 1;
    FCellDown.Y := - 1;
  end;
  inherited;
end;

function TPBButtonDrawGrid.SelectCell(ACol, ARow: Integer): Boolean;
begin
  result := false;
end;

end.
```

Original resource:	The Delphi Pool
Author:	Peter Below
Added:	2013-01-27
Last updated:	2013-01-27