

Le Club Developpez.com n'affiche que des publicités IT, discrètes et non intrusives.

Afin que le Club puisse rester gratuit, nous vous serions reconnaissant d'ajouter Developpez.com dans la liste d'exceptions de votre bloqueur de publicité.

Jouez au Puissance 4 ! La solution du vainqueur: Cl@udius



Date de publication : 3 avril 2007 , Date de mise à jour : 3 avril 2007

Par l'Equipe DELPHI (Les Défis DELPHI)

Cl@udius nous propose de nous faire découvrir sa solution !

I. Introduction
I-A. DrawGrid
I-B. TGrille
II. Déroulement du jeu
III. Algorithme général
IV. Partie graphique
V. Conclusion

I. Introduction

Le programme est basé essentiellement sur l'utilisation d'un **DrawGrid** pour la partie graphique et sur la manipulation de la variable de type **TGrille** qui représente l'état du 'plateau' sur lequel évoluent les deux joueurs.

I-A. DrawGrid

J'ai choisi d'utiliser le composant **DrawGrid** afin de bénéficier le plus simplement possibles de son comportement naturel, notamment les événements SelectCell et DrawCell.

I-B. TGrille

Le type TGrille est un tableau d'entiers aux dimensions du jeu Puissance 4 (7 colonnes x 6 lignes).

Ce tableau peut recevoir les valeurs constantes suivantes:

```
const
  VIDE = 0;
  JAUNE = 1;
  ROUGE = -1;
  GAGNANT = 2;
```

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [J'accepte](#)

1. La grille est initialisée avec la valeur VIDE
2. La variable **FTour** détermine la couleur du joueur qui a la main
3. Un click sur le DrawGrid déclenche l'évènement **SelectCell** qui dépose un jeton dans la première case vide de la colonne sur laquelle le click a été effectué, puis vérifie si ce jeton est gagnant.
4. Le tour est inversé et retour à l'étape 3, jusqu'à une situation gagnante pour l'un des joueurs ou un match nul.

III. Algorithme général

Différentes fonctions contenues dans l'unité **uDefs** permettent de connaître la situation actuelle sur le plateau.

Notamment :

- **DispoCol** : Nombre de cases vides dans une colonne
- **DispoGrille** : Nombre de cases vides dans la grille
- **PointDeChute** : Calcul du point de chute d'un jeton dans une colonne

Et enfin

- **JoueurGagnant** : Déterminer le dernier jeton déposé donne une situation au joueur

Analyse du dernier jeton joué

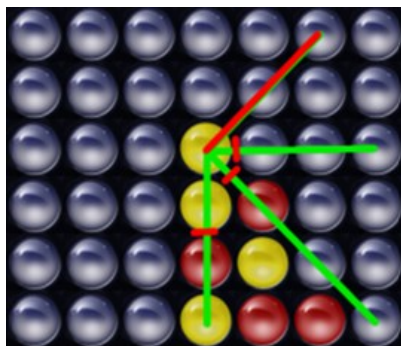
Contrairement à **TicTacToe** qui analyse uniquement la périphérie de jeton joué, la fonction **JoueurGagnant** scrute l'ensemble du plateau afin de détecter une situation gagnante pour le joueur.

```
var
  Col, Row: Byte;
begin
  result := False;

  Col := 0;
  while (Col <= MAX_COL) and (not result) do
  begin
    Row := 0;
    while (Row <= MAX_ROW) and (not Result) do
    begin
      result := (DirectionGagnante(1, 0) or // Horizontal
                DirectionGagnante(0, 1) or // Vertical
                DirectionGagnante(1, 1) or // Diagonal \
                DirectionGagnante(1,-1)); // Diagonal /
      Inc(Row);
    end;
    Inc(Col);
  end;
end;
```

La procédure imbriquée 'DirectionGagnante' examine pour chaque direction si le jeton posé est gagnant.

Le comptage s'interrompt dès que la couleur est différente à celle du jeton de référence. Le comptage ne sera pas effectué sur le diagonale / puisque dans ce cas l'alignement de 4 jetons ne peut-être effectué sans sortir de la grille de jeu.



Comptage des jetons

IV. Partie graphique

Le dessin du plateau est effectué par l'implémentation de l'évènement **OnDrawCell** du **TDrawGrid**.

Les images bitmaps : case vide, jeton jaune, rouge ou gagnant (vert) sont inclus au projet comme ressources.

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [J'accepte](#)

```
myBitmap: TBitmap;  
begin  
  myBitmap := TBitmap.Create;  
  try  
    case FGrille[ACol, ARow] of  
      GAGNANT: myBitmap.LoadFromResourceName(hInstance, 'JETON_VERT');  
      JAUNE:   myBitmap.LoadFromResourceName(hInstance, 'JETON_JAUNE');  
      ROUGE:   myBitmap.LoadFromResourceName(hInstance, 'JETON_ROUGE');  
      else     myBitmap.LoadFromResourceName(hInstance, 'JETON_VIDE');  
    end;  
    with Grille do  
      begin  
        Canvas.StretchDraw(Rect, myBitmap);  
      end;  
    finally  
      myBitmap.Free;  
    end;  
  end;  
end;
```

V. Conclusion

Elaborer un tel programme m'a procuré beaucoup de plaisir, comme je suppose à chacun des participants qui furent particulièrement nombreux sur ce défi.

Rendez-vous au prochain défi !

Cl@udius

Copyright © 2006 Equipe Delphi Developpez LLC. Tous droits réservés Developpez LLC. Aucune reproduction, même partielle, ne peut être faite de ce site ni de l'ensemble de son contenu : textes, documents et images sans l'autorisation expresse de Developpez LLC. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Responsables bénévoles de la rubrique Delphi : Gilles Vasseur - Alcatiz - [Contacter par email](#)

[Nous contacter](#)

[Participez](#)

[Hébergement](#)

[Informations légales](#)

[Partenaire : Hébergement Web](#)

© 2000-2019 - www.developpez.com

En utilisant ce site, vous acceptez l'utilisation de cookies permettant de vous proposer des contenus et des services adaptés à vos centres d'intérêts - [J'accepte](#)