Documentation Programmeur - Utilitaire Cartes Hexagonales

🏗️ Architecture générale

Technologies utilisées

- **Langage** : Free Pascal 3.2.2 (Object Pascal)
- **Graphique** : raylib + raygui (guvacode)
- **Paradigme** : Programmation procédurale avec structures

Structure des fichiers

```
hexagongridflattop.lpr      // Programme principal
├── initvariable.pas        // Variables globales et types
├── hexagonlogic.pas        // Logique de génération d'hexagones
├── detectionlogic.pas      // Algorithmes de détection de terrain
├── boutonclic.pas          // Gestion des boutons
└── traceastar.pas          // Algorithme A* (pathfinding)
```

📦 Types de données principaux

THexCell - Structure d'un hexagone

pascal

```pascal
THexCell = record
  Number: integer;                  // Numéro unique (1..TotalNbreHex)
  center: TVector2;                 // Position du centre (x, y)
  Vertices: array[0..5] of TPoint;      // 6 sommets de l'hexagone
  Color: TColor;                    // Couleur d'affichage (damier)
  ColorPt: TColor;                  // Couleur échantillonnée de la carte
  Selected: boolean;                // État de sélection
  Neighbors: array[1..6] of integer;    // Numéros des 6 voisins
  Colonne, ligne: integer;          // Position dans la grille
  Poshexagone: TEmplacement;            // Type d'emplacement (coin, bord, centre)
  PairImpaircolonne: boolean;           // Parité pour calcul des voisins
  TypeTerrain: Integer;             // Type détecté (0=aucun, 1,2,3...=types)
  IsReference: Integer;             // Numéro de référence (0=pas référence)
  // + Champs A* (GCost, HCost, FCost, Parent, Closed, Open)
end;
```

TColorSignature - Signature couleur pour détection

```pascal
TColorSignature = record
  DominantColors: array[0..2] of TColor;  // 3 couleurs principales
  ColorCounts: array[0..2] of Integer;    // Fréquence de chaque couleur
  TotalPixels: Integer;                   // Nombre total de pixels analysés
  IsValid: Boolean;                       // Signature valide
end;
```

Variables globales importantes

```pascal
HexGrid: array of THexCell;          // Grille dynamique d'hexagones
columns, rows: integer;              // Dimensions de la grille
TotalNbreHex: integer;               // Nombre total d'hexagones
HexOrientation: THexOrientation;     // hoFlatTop ou hoPointyTop
AppMode: TAppMode;                   // amNormal, amDetection, amSuppression
DetectionActive: Boolean;            // Mode sélection références actif
NombreReferences: Integer;           // Nombre de références sélectionnées
ReferenceSignatures: array of TColorSignature;  // Signatures des références
```

🔧 Modules principaux

1. initvariable.pas

Rôle : Variables globales, types, utilitaires de sauvegarde/chargement

Fonctions clés

```pascal
procedure RecalculerDimensionsHex;          // Calcule HexRadius, HexWidth, HexHeight
procedure RedimensionnerHexGrid;            // Redimensionne le tableau HexGrid
procedure SauvegarderParametresAjustement;  // Sauve dans ajustements.txt
procedure ChargerParametresAjustement;      // Charge depuis ajustements.txt
procedure LoadCarteComplete(carteName);     // Charge carte + paramètres + CSV
function LoadDetectionDataFromCSV: Boolean; // Charge TypeTerrain et IsReference
```

Système de fichiers

- **./save/NomCarte/** : Dossier par carte
  - `NomCarte.png/.jpg` : Image de la carte
  - `ajustements.txt` : Paramètres de grille
  - `hexgridplat.csv` : Données complètes des hexagones

2. hexagonlogic.pas

Rôle : Génération et calcul des voisins

Fonctions principales

```pascal
procedure GenerateHexagons;                 // Génère tous les hexagones
procedure CalculateNeighbors;               // Calcule tous les voisins
procedure CalculateHexVertices(var Hex);    // Calcule les 6 sommets
function PairOuImpairCol(Number): boolean;  // Teste parité colonne
```

Algorithme de positionnement

1. **Espacement** calculé selon orientation (Flat/Pointy Top)
2. **Décalage CoinIn** pour colonnes/lignes impaires
3. **Référence Hex1** comme point d'origine
4. **Échelle HexScale** appliquée uniformément

Calcul des voisins

- **6 directions** numérotées 1-6 (sens horaire)
- **Algorithmes séparés** pour Flat Top et Pointy Top
- **Gestion des bords** avec `TEmplacement` (coin, bord, centre)
- **Parité** : comportement différent selon colonne/ligne paire/impaire

3. detectionlogic.pas

Rôle : Analyse d'image et classification de terrain

Pipeline de détection

```pascal
1. StartReferenceSelection()      // Active le mode sélection
2. HandleDetectionClick(hexNumber) // Ajoute une référence
3. StopReferenceSelection()       // Lance l'analyse
4. AnalyzeAllReferences()         // Crée les signatures couleur
5. ClassifyAllHexagons()          // Classifie tous les hexagones
```

Algorithme d'analyse couleur

pascal

```
function CountColorsInCircle(centerX, centerY, radius): TColorSignature;
```

1. **Zone d'analyse** : Cercle de rayon 80% du HexRadius
2. **Échantillonnage** : Tous les pixels dans le cercle
3. **Regroupement** : Couleurs similaires (seuil=40) fusionnées
4. **Tri** : Par fréquence décroissante
5. **Signature** : 3 couleurs dominantes + leurs fréquences

Classification

pascal

```
function CompareSignatures(sig1, sig2, strictOrder): Boolean;
```

- **Mode strict** : Couleurs dans le même ordre
- **Mode non-strict** : Couleurs présentes, ordre libre
- **Seuil similarité** : Distance Manhattan ≤ 40

Réinitialisation

pascal

```
procedure ResetDetectionComplete;  // Efface tout : références, types, signatures
```

🎮 Gestion des événements

HandleDragAndDrop()

Logique multi-mode :

```pascal
case AppMode of
  amNormal:
    // Sélection classique pour informations
  amDetection:
    if DetectionActive then
      HandleDetectionClick(i)    // Ajouter référence
    else if NombreReferences > 0 then
      HexGrid[i].TypeTerrain := ValeurSpinnerCorrection  // Correction manuelle
  amSuppression:
    // Non implémenté
end;
```

Interface conditionnelle

- **Spinner correction** : Visible si `AppMode = amDetection` ET `NombreReferences > 0`
- **MessageBox reset** : Apparaît si nouvelle détection avec références existantes
- **Statut dynamique** : Mise à jour temps réel pendant classification

---

🔄 Workflow de développement

Ajouter un nouveau type de terrain

1. **Étendre TColorSignature** si nécessaire
2. **Modifier CompareSignatures()** pour logique spécifique
3. **Ajuster l'affichage** dans DrawHexGrid()

Ajouter un nouveau mode d'application

1. **TAppMode** : Ajouter enum
2. **ToggleGroup** : Étendre la chaîne de caractères
3. **HandleDragAndDrop()** : Ajouter case
4. **DrawGUIPanel()** : Interface spécifique

Optimisations possibles

- **Threading** : Analyse couleur en arrière-plan
- **Cache signatures** : Éviter recalculs identiques
- **Algorithmes de clustering** : K-means au lieu du seuil fixe
- **GPU** : Analyse parallèle avec OpenGL/compute shaders

📊 Format CSV de sauvegarde

Structure du fichier hexgridplat.csv

csv

```
Number,CenterX,CenterY,ColorR,ColorG,ColorB,ColorPtR,ColorPtG,ColorPtB,
BSelected,Colonne,Ligne,Emplacement,PairImpairLigne,
Vertex1X,Vertex1Y,Vertex2X,Vertex2Y,Vertex3X,Vertex3Y,
Vertex4X,Vertex4Y,Vertex5X,Vertex5Y,Vertex6X,Vertex6Y,
Neighbor1,Neighbor2,Neighbor3,Neighbor4,Neighbor5,Neighbor6,
TypeTerrain,IsReference
```

Champs importants pour la détection

- **TypeTerrain** : 0=non déterminé, 1,2,3...=types détectés
- **IsReference** : 0=hexagone normal, 1,2,3...=numéro de référence
- **ColorPt** : Couleur échantillonnée (RGB) pour validation

🔧 API des fonctions utilitaires

Gestion de grille

pascal

```
procedure RecalculerTotalHex;          // TotalNbreHex := columns * rows
procedure GenererNouvelleGrille;       // Redimensionne + régénère + recalcule voisins
function AppliquerParametresGrille: boolean;  // Valide et applique nouveaux paramètres
```

Détection et signatures

pascal

```
function AnalyzeHexagonColors(hexNumber): TColorSignature;    // Analyse un hexagone
function FindMatchingReference(hexSignature): Integer;       // Trouve référence correspondante
function ColorsAreSimilar(color1, color2, threshold): Boolean;  // Test similarité couleur
```

Utilitaires d'affichage

pascal

```
procedure DrawHexGrid(dessineLesNombres);    // Affiche grille avec numéros optionnels
function EmplacementToString(Emplacement): string;  // Convertit enum en chaîne
function ColorToString(Color): string;       // Convertit TColor en "R,G,B"
```

⚠️ Points d'attention

Threading et performance

- **Raylib** n'est pas thread-safe : toutes les opérations graphiques sur thread principal
- **Analyse couleur** peut être lente sur grandes grilles : progression visible recommandée

Gestion mémoire

- **HexGrid dynamique** : SetLength() à chaque changement de taille
- **ReferenceSignatures** : Taille = TotalNbreHex + 1 (index 1..n)
- **Pas de fuites** : raylib gère automatiquement textures/images

Compatibilité formats

- **Images supportées** : PNG, BMP, JPG, JPEG (raylib)
- **Chemins relatifs** : ./save/, ./ressources/
- **Séparateur décimal** : Point (.) pour compatibilité internationale

Extensions futures

- **Formats cartes** : Support de formats vectoriels (SVG)
- **Export** : Formats de jeu (Vassal, ASLSK)
- **IA** : Classification par réseaux de neurones
- **Multirésolution** : Grilles hiérarchiques

Cette documentation technique couvre l'architecture complète du système. Consultez le code source pour les détails d'implémentation spécifiques.