# Technical Documentation - Hexagonal Grid

## Table of Contents

1. [Project Architecture](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#project-architecture)
2. [Data Structures](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#data-structures)
3. [Modules and Units](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#modules-and-units)
4. [Key Algorithms](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#key-algorithms)
5. [Coordinate System](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#coordinate-system)
6. [Neighborhood Calculation](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#neighborhood-calculation)
7. [State Management](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#state-management)
8. [File Formats](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#file-formats)
9. [API and Main Functions](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#api-and-main-functions)
10. [Optimizations and Performance](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#optimizations-and-performance)
11. [Extension and Maintenance](https://claude.ai/chat/770bd2e0-81eb-44cc-8839-911bedc548a0#extension-and-maintenance)

## Project Architecture

### Technology

* **Language**: Free Pascal 3.2.2
* **Graphics Framework**: RayLib + RayGUI
* **Paradigm**: Procedural programming with structures

### File Structure

src/

├── hexagongridflattop.lpr # Main program

├── initVariable.pas # Global variables and structures

├── HexagonLogic.pas # Hexagonal logic

├── DetectionLogic.pas # AI detection system

├── BoutonClic.pas # Button management

└── traceastar.pas # A\* algorithm (pathfinding)

## Data Structures

### THexCell - Main Structure

THexCell = record

Number: integer; // Unique ID

center: TVector2; // Center in pixels

Vertices: array[0..5] of TPoint; // 6 vertices

Color: TColor; // Display color

ColorPt: TColor; // Sampled color

Selected: boolean; // Selection state

Neighbors: array[1..6] of integer; // Neighbors (0 = none)

Colonne, ligne: integer; // Position in grid

Poshexagone: TEmplacement; // Location type

PairImpaircolonne, PairImpairLigne: boolean; // Parity

// Detection system

TypeTerrain: Integer; // Detected type (0-N)

IsReference: Integer; // Reference number

// Suppression/exemption system

Supprime: Boolean; // Invisible + inaccessible

Exempt: Boolean; // Visible + inaccessible

// A\* pathfinding

GCost, HCost, FCost: Integer;

Parent: Integer;

Closed, Open: Boolean;

end;

### TEmplacement - Position Enumeration

TEmplacement = (

inconnu, // Undefined

CoinHG, CoinHD, // Top left/right corners

CoinBG, CoinBD, // Bottom left/right corners

BordH, BordB, // Top/bottom edges

BordG, BordD, // Left/right edges

Classic, // Central hexagon

Bloque // Blocked hexagon

);

### THexOrientation - Orientation Types

THexOrientation = (

hoFlatTop, // Flat side on top (⬡)

hoPointyTop // Point on top (⬢)

);

## Modules and Units

### initVariable.pas

**Responsibilities**:

* Global variables and constants
* Main data structures
* Map and save management
* User interface

**Key Functions**:

procedure RecalculerDimensionsHex;

procedure SauvegarderParametresAjustement;

procedure ChargerParametresAjustement;

procedure LoadDetectionDataFromCSV;

procedure SaveHexGridToCSV;

### HexagonLogic.pas

**Responsibilities**:

* Hexagonal grid generation
* Neighborhood calculation
* Suppression/exemption logic

**Key Functions**:

procedure GenerateHexagons;

procedure CalculateNeighbors;

procedure ExempterHexagone(hexNumber: Integer);

procedure RestaurerHexagoneExempt(hexNumber: Integer);

### DetectionLogic.pas

**Responsibilities**:

* AI terrain detection system
* Color analysis
* Automatic classification

**Key Functions**:

procedure InitDetectionSystem;

function AnalyzeHexagonColors(hexNumber: Integer): TColorSignature;

procedure ClassifyAllHexagons;

## Key Algorithms

### 1. Hexagonal Grid Generation

// Calculate spacing according to orientation

case HexOrientation of

hoFlatTop:

begin

horizontalSpacing := hexWidth \* 3/4;

verticalSpacing := hexHeight;

end;

hoPointyTop:

begin

horizontalSpacing := hexWidth;

verticalSpacing := hexHeight \* 3/4;

end;

end;

// Positioning with offset for odd rows/columns

offsetX := Hex1ReferenceX + (x - 1) \* horizontalSpacing;

offsetY := Hex1ReferenceY + (y - 1) \* verticalSpacing;

if (parity) then

offset += adjustment; // According to CoinIn and orientation

### 2. Neighborhood Calculation

Calculation depends on 4 parameters:

* **Orientation** (Flat/Pointy)
* **CoinIn** (even/odd row offset)
* **Position** (corner, edge, center)
* **Parity** (even or odd row/column)

// Example for central hexagon, Flat Top, CoinIn=False, even column

HexGrid[i].Neighbors[1] := HexGrid[i].Number - columns; // North

HexGrid[i].Neighbors[2] := HexGrid[i].Number + 1; // North-East

HexGrid[i].Neighbors[3] := HexGrid[i].Number + columns + 1; // South-East

HexGrid[i].Neighbors[4] := HexGrid[i].Number + columns; // South

HexGrid[i].Neighbors[5] := HexGrid[i].Number + columns - 1; // South-West

HexGrid[i].Neighbors[6] := HexGrid[i].Number - 1; // North-West

### 3. AI Terrain Detection

// 1. Color analysis in a circle

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

// 2. Signature comparison

function CompareSignatures(sig1, sig2: TColorSignature): Boolean;

// 3. Automatic classification

procedure ClassifyAllHexagons;

## Coordinate System

### Screen Coordinates

* **Origin**: Top-left corner (0,0)
* **Unit**: Pixels
* **Reference**: Hex1ReferenceX/Y for first hexagon

### Grid Coordinates

* **Columns**: 1 to N (left to right)
* **Rows**: 1 to N (top to bottom)
* **Conversion**: Number = (row-1) \* columns + column

### Relative Coordinates

// Save relative to map position

DeltaRelativeX := Hex1ReferenceX - lacarte.position.x;

DeltaRelativeY := Hex1ReferenceY - lacarte.position.y;

// Restore

Hex1ReferenceX := lacarte.position.x + DeltaRelativeX;

Hex1ReferenceY := lacarte.position.y + DeltaRelativeY;

## Neighborhood Calculation

### Neighborhood Matrix (Flat Top)

1 2

6 H 3

5 4

### Cleanup Algorithm

procedure NettoyerVoisinagesExempts;

begin

for i := 1 to TotalNbreHex do

if HexGrid[i].Exempt or HexGrid[i].Supprime then

for voisin in HexGrid[i].Neighbors do

RemoveReferenceInNeighbor(voisin, i);

end;

## State Management

### Hexagon States

| **State** | **Visible** | **Accessible** | **Display** |
| --- | --- | --- | --- |
| Normal | ✓ | ✓ | Normal hexagon |
| Suppressed | ✗ | ✗ | Invisible |
| Exempt | ✓ | ✗ | Red "O" |
| Reference | ✓ | ✓ | Red number |

### State Machine

case AppMode of

amNormal: // Navigation, selection

amDetection: // AI, classification, correction

amSuppression: case AppModeSuppressionIndex of

0: // Suppression mode (red X)

1: // Exemption mode (red O)

end;

end;

## File Formats

### CSV (hexgridplat.csv)

Number,CenterX,CenterY,ColorR,ColorG,ColorB,...,TypeTerrain,IsReference,Supprime,Exempt

1,50,50,0,255,0,...,1,0,False,False

2,87,50,128,128,128,...,2,1,False,True

### Parameters (adjustments.txt)

HexDiameter=70.67

HexScale=0.97

HexOrientation=0

CoinIn=False

DeltaRelativeX=83.00

DeltaRelativeY=38.00

Columns=25

Rows=20

## API and Main Functions

### Grid Management

procedure GenerateHexagons; // Generate grid

procedure CalculateNeighbors; // Calculate all neighborhoods

procedure RecalculerDimensionsHex; // Recalculate sizes by scale

### State Management

procedure ExempterHexagone(hexNumber: Integer); // Exempt a hexagon

procedure RestaurerHexagoneExempt(hexNumber: Integer); // Restore an exempt

procedure NettoyerVoisinagesExempts; // Clean neighborhoods

### AI Detection

procedure InitDetectionSystem; // Initialize system

procedure StartReferenceSelection; // Start selection

procedure ClassifyAllHexagons; // Automatic classification

### Save/Load

procedure SaveHexGridToCSV; // Save to CSV

function LoadDetectionDataFromCSV: Boolean; // Load from CSV

procedure SauvegarderCarteUniverselle; // Complete save

## Optimizations and Performance

### Neighborhood Calculation

* **Complexity**: O(N) where N = number of hexagons
* **Frequency**: Only during structural changes
* **Optimization**: Result caching, incremental calculation

### Display

* **Culling**: Off-screen hexagons not drawn
* **Suppressed**: Completely ignored in Normal/Detection modes
* **Batch rendering**: Grouping of graphics operations

### Memory

* **Dynamic array**: SetLength(HexGrid, TotalNbreHex + 1)
* **Reuse**: State preservation during regeneration
* **Garbage collection**: Explicit texture liberation

## Extension and Maintenance

### Adding New Hexagon Types

1. Extend TEmplacement enumeration
2. Add cases in TrouveLesVoisins()
3. Update EmplacementToString()
4. Test all neighborhood modes

### Adding New Modes

1. Extend TAppMode
2. Add logic in HandleDragAndDrop()
3. Create interface in DrawGUIPanel()
4. Implement specific functions

### Save Compatibility

// CSV version management

if elementCount >= 36 then

// Format with Exempt

else if elementCount >= 35 then

// Format without Exempt (backward compatibility)

### Recommended Tests

* **Neighborhoods**: All orientation/CoinIn combinations
* **States**: Transitions between Normal/Suppressed/Exempt
* **Save**: Backward compatibility and integrity
* **Performance**: Large grids (>1000 hexagons)

Technical Documentation v1.0 - Free Pascal 3.2.2 + RayLib