**User Documentation - Image Cutter**

**Program Overview**

The **Image Cutter** is an application developed in Pascal with Free Pascal/Lazarus using Raylib and RayGUI libraries. It allows cutting an image into multiple smaller images according to a customizable grid.

**Main Objective**

Automatically cut a large image (such as an old map, comic strip sheet, etc.) into several small numbered images or named according to a coordinate system.

**Installation and Setup**

**System Requirements**

**Technical Prerequisites**

* **Operating System**: Windows, Linux, macOS
* **Free Pascal** with Lazarus IDE
* **Libraries**: Raylib and RayGUI
* **Supported Image Formats**: PNG, JPG, BMP
* **Memory**: minimum 512 MB RAM
* **Disk Space**: 100 MB free

**Complete Installation Structure**

ImageCutter/ (Main project folder)

├── Sources/ (Source code)

│ ├── imagecutter.lpr (Main program)

│ ├── init.pas (Initialization and structures)

│ ├── gui\_interface.pas (User interface)

│ └── gridrenderer.pas (Graphics rendering)

├── resources/ (Required files)

│ ├── carte1870.png (MANDATORY example image)

│ └── roboto.ttf (Optional custom font)

├── build/ (Compilation files)

│ └── imagecutter.exe (Final executable - Windows)

├── YourImages/ (Folder for your images)

│ ├── image1.png (Your images to cut)

│ ├── image2.jpg (Supported formats: PNG, JPG, BMP)

│ └── ...

└── Results/ (Created automatically)

├── savedpictures/ (Default output folder)

├── myproject/ (Other projects with different prefixes)

└── ...

**Step-by-Step Installation**

**Step 1: Environment Preparation**

1. **Install Free Pascal and Lazarus**
   * Download from https://www.lazarus-ide.org/
   * Install with default components
2. **Install Raylib/RayGUI Libraries**
   * Follow official Raylib documentation for Pascal
   * Verify libraries are accessible

**Step 2: File Preparation**

1. **Create folder structure**
2. mkdir ImageCutter
3. mkdir ImageCutter/Sources
4. mkdir ImageCutter/resources
5. mkdir ImageCutter/YourImages
6. **Place source files** in Sources/
   * Copy the 4 .pas/.lpr files
7. **MANDATORY: Test image file**
   * Place carte1870.png in resources/
   * ⚠️ **WITHOUT THIS FILE, THE PROGRAM WON'T START**
8. **Optional: Custom font**
   * Place roboto.ttf in resources/
   * If absent, system font will be used

**Step 3: Compilation**

1. **Open Lazarus IDE**
2. **Open project**: Sources/imagecutter.lpr
3. **Compile**: Run Menu → Compile (F9)
4. **Executable** is created in project folder

**Preparing Images to Cut**

**Where to place your images**

1. **Recommended folder**: YourImages/
2. **Accepted formats**: PNG (recommended), JPG, BMP
3. **Maximum size**: depends on your RAM (recommended < 50 MB)
4. **File names**: avoid special characters and accents

**Modify source file for your images**

**IMPORTANT**: By default, the program loads resources/carte1870.png

To load YOUR images, modify in gui\_interface.pas:

function LoadImageDialog(): string;

begin

// REPLACE this line:

Result := 'resources/carte1870.png';

// WITH the path to your image:

Result := 'YourImages/your\_image.png';

// Or for selection interface (advanced):

// Result := SelectFileDialog();

end;

**Practical modification example**

function LoadImageDialog(): string;

begin

// To cut a postcard:

Result := 'YourImages/postcard.jpg';

// To cut a stamp sheet:

Result := 'YourImages/stamp\_sheet.png';

// Automatic file verification

if not FileExists(pchar(Result)) then

begin

WriteLn('ERROR: File not found: ', Result);

Result := '';

end;

end;

**User Interface**

**Main Area (Left)**

* **Image display** centered and automatically resized
* **Cutting grid** overlaid (when activated)
* **Visual selection** of active cell (red rectangle)
* **Coordinates** displayed above selected cell

**Control Panel (Right - 300px)**

Interface organized in vertical sections with all necessary controls.

**Commands to Run the Program**

**Program Startup**

**Method 1: From executable (recommended)**

# Windows

cd ImageCutter

build/imagecutter.exe

# Linux/macOS

cd ImageCutter

./build/imagecutter

# Or double-click on executable

**Method 2: Direct compilation and execution**

# In Lazarus IDE

1. Open Sources/imagecutter.lpr

2. Press F9 (Compile and Run)

3. Or Menu: Run → Compile and Run

**Method 3: Command line**

# Manual compilation

cd ImageCutter/Sources

fpc imagecutter.lpr

# Execution

./imagecutter

**Startup Verifications**

**Normal console messages**

=== IMAGE CUTTER ===

Controls:

- G: Show/Hide grid

- Arrows: Move grid

- Space: Start cutting

- Esc: Quit

Custom font loaded with French characters

**Possible error messages**

ERROR: File not found: resources/carte1870.png

→ Solution: Place mandatory image file

Default font used - roboto.ttf file not found

→ Normal if no custom font

ERROR: Unable to load texture

→ Check image format (PNG/JPG/BMP)

**Step-by-Step Usage Guide**

**Step 1: Startup and Verification**

1. **Launch application**
2. cd ImageCutter
3. build/imagecutter.exe
4. **Check interface**
   * 1280×800 pixel window opens
   * Right panel with visible controls
   * Main area displays "No image loaded"
5. **Active console**
   * Startup messages displayed
   * No critical error messages

**Step 2: Load Your Image**

**Option A: Use default image**

1. **Click "Load Image"** in right panel
2. **Automatic loading** of resources/carte1870.png
3. **Verification**: centered image with displayed information

**Option B: Modify for your image**

1. **Stop program** (Esc)
2. **Modify code** in gui\_interface.pas:
3. function LoadImageDialog(): string;begin Result := 'YourImages/your\_image.png'; // ← Change hereend;
4. **Recompile** (F9 in Lazarus)
5. **Restart** and click "Load Image"

**Option C: Copy your image in place**

1. **Rename your image** to carte1870.png
2. **Copy it** to resources/
3. **Replace** existing image
4. **Run program** normally

**Step 3: Grid Configuration**

**Initial configuration**

1. **Choose mode**:
   * Uncheck "Fixed square mode" = adaptive rectangles
   * Check "Fixed square mode" = fixed-size squares
2. **Define dimensions**:
   * Adjust "Rows" (spinner 1-50)
   * Adjust "Columns" (spinner 1-50)
3. **Show grid**:
   * Click "Show Grid" OR
   * Press **G**

**Fine adjustment**

1. **Fixed square mode only**:
   * "Square size" slider (10-200 px)
   * Mouse wheel for precise adjustment
2. **Positioning**:
   * Keyboard arrows to move grid
   * Hold down for continuous movement

**Step 4: Selection and Preview**

1. **Click on a cell** in the grid
   * Red rectangle with transparency appears
   * Coordinates displayed above
2. **Change display mode**:
   * Press **M** to toggle coordinates ↔ number
   * "Cell: 0,1" or "Box: 2"

**Step 5: Save Configuration**

**Naming parameters**

1. **Modify prefix**:
   * Clear "savedpictures" in text box
   * Type your prefix: "map", "stamp", "sheet", etc.
2. **Choose traversal mode**:
   * Uncheck "Verso mode" = normal traversal (left→right)
   * Check "Verso mode" = reversed traversal (right→left)
3. **Check displayed example**:
   * "map\_1.png (recto)" or "map\_L0C0.png (verso)"

**Step 6: Image Cutting**

**Option A: Complete cutting**

1. **Button method**:
   * Click "Start Cutting"
   * Progress displayed in console
2. **Keyboard method**:
   * Press **Space**
   * Faster for experienced users

**Option B: Cell-by-cell cutting**

1. **Double-click** on desired cell
2. **Immediate save** of that cell only
3. **Ideal** for selective extractions

**Process tracking**

Console displays:

=== CUTTING START ===

Grid: 3 rows × 4 columns

Mode: Adaptive rectangles

Traversal: Recto

Prefix: map

Cell saved: ./map/map\_1.png

Cell saved: ./map/map\_2.png

...

=== CUTTING COMPLETED ===

Cells saved: 12

Cells ignored: 0

Total processed: 12

**Step 7: Retrieving Results**

**File location**

ImageCutter/

└── [prefix]/ # Automatically created folder

├── [prefix]\_1.png # Numbering mode

├── [prefix]\_2.png

└── ...

# OR

├── [prefix]\_L0C0.png # Coordinates mode

├── [prefix]\_L0C1.png

└── ...

**Commands to check results**

# Windows

dir [prefix]

dir map

# Linux/macOS

ls [prefix]/

ls map/

# Count created files

# Windows

dir map /b | find /c ".png"

# Linux/macOS

ls map/\*.png | wc -l

**Advanced Commands and Troubleshooting**

**Diagnostic Commands**

**Check file structure**

# From ImageCutter folder

echo "=== Project structure ==="

dir /S # Windows

find . -type f # Linux/macOS

echo "=== Available images ==="

dir YourImages # Windows

ls YourImages/ # Linux/macOS

echo "=== Generated results ==="

dir /AD # Windows (folders only)

ls -d \*/ # Linux/macOS (folders only)

**Check configuration**

echo "=== Compilation test ==="

cd Sources

fpc -l imagecutter.lpr # List dependencies

echo "=== Image test ==="

file resources/carte1870.png # Linux/macOS - image info

**Cleanup Commands**

**Clean temporary files**

# Remove compilation files

rm \*.o \*.ppu # Linux/macOS

del \*.o \*.ppu # Windows

# Clean all results

rm -rf savedpictures/ map/ stamp/ # Linux/macOS

rmdir /S savedpictures map stamp # Windows

**Reset for new project**

# Keep only sources and resources

mkdir Backup

cp Sources/\* Backup/ # Linux/macOS

copy Sources\\* Backup\ # Windows

cp resources/\* Backup/ # Linux/macOS

copy resources\\* Backup\ # Windows

# Clean everything else

rm -rf build/ \*/ # Linux/macOS

rmdir /S build # Windows

**Automation Scripts**

**Automatic launch script (Windows)**

@echo off

echo === Image Cutter ===

cd /d "%~dp0"

if exist "build\imagecutter.exe" (

echo Launching program...

build\imagecutter.exe

) else (

echo ERROR: Executable not found

echo First compile the project in Lazarus

pause

)

**Automatic launch script (Linux/macOS)**

#!/bin/bash

echo "=== Image Cutter ==="

cd "$(dirname "$0")"

if [ -f "build/imagecutter" ]; then

echo "Launching program..."

./build/imagecutter

else

echo "ERROR: Executable not found"

echo "First compile the project in Lazarus"

read -p "Press Enter to continue..."

fi

**Maintenance Commands**

**Backup completed project**

# Create complete archive

mkdir Archive\_$(date +%Y%m%d) # Linux/macOS

mkdir Archive\_%date:~-4,4%%date:~-7,2%%date:~-10,2% # Windows

# Copy results + configuration

cp -r [prefix]/ Archive\_\*/ # Linux/macOS

xcopy [prefix] Archive\_\*\ /E # Windows

**Change image quickly**

# Method by file replacement

cp YourImages/new\_image.png resources/carte1870.png

# Or create symbolic link (Linux/macOS)

ln -sf ../YourImages/new\_image.png resources/carte1870.png

**Keyboard Shortcuts**

| **Key** | **Action** |
| --- | --- |
| **G** | Show/Hide grid |
| **↑↓←→** | Move grid (hold = continuous) |
| **M** | Toggle display mode (coordinates ↔ number) |
| **Space** | Start complete cutting |
| **Esc** | Quit application |
| **Wheel** | Adjust square size (fixed square mode) |

**Operating Modes**

**Adaptive Rectangle Mode**

* **Principle**: divides image into equal rectangles
* **Calculation**: cell\_width = image\_width ÷ columns
* **Advantages**: covers exactly entire image
* **Usage**: maps, uniform sheets

**Fixed Square Mode**

* **Principle**: all squares have same size
* **Setting**: size in pixels (10-200)
* **Advantages**: constant size, can extract beyond edges
* **Usage**: playing cards, stamps, regular objects

**Verso Mode**

* **Recto**: 1→2→3 / 4→5→6 (normal)
* **Verso**: 3→2→1 / 6→5→4 (reversed)
* **Usage**: to digitize card backs in correct order

**File Management**

**Output Structure**

Project/

└── [prefix]/

├── [prefix]\_1.png

├── [prefix]\_2.png

└── ... (or coordinates format)

**File Names**

**Coordinates mode**:

* Recto: prefix\_L0C0.png, prefix\_L0C1.png, prefix\_L1C0.png...
* Verso: prefix\_L0C2.png, prefix\_L0C1.png, prefix\_L0C0.png...

**Numbering mode**:

* Recto: prefix\_1.png, prefix\_2.png, prefix\_3.png...
* Verso: prefix\_3.png, prefix\_2.png, prefix\_1.png...

**Error Handling**

* **Cells exceeding image**: automatically ignored
* **Non-existent folder**: automatically created
* **Invalid name**: error message in console

**Displayed Information**

**Information Panel**

* **Image file name** loaded
* **Image dimensions** (width × height)
* **Calculated cell sizes**
* **Total number** of cells
* **Current mode** (rectangle/square, recto/verso)

**Debug Console**

* Image loading messages
* Grid calculations
* Cutting progress
* Errors and warnings

**Selected Cell Display**

* **Red rectangle** with transparency
* **Coordinates**: "Cell: X,Y" or "Box: N"
* **Position**: above selected cell

**Usage Examples**

**Case 1: Old Map (3×4)**

1. Load historical map
2. Adaptive rectangle mode
3. 3 rows × 4 columns
4. Prefix "map1870"
5. Cutting → 12 files map1870\_1.png to map1870\_12.png

**Case 2: Playing Card Sheet**

1. Load sheet image
2. Fixed square mode, 64×64 pixels
3. Adjust position with arrows
4. Verso mode for card backs
5. Double-click to extract card by card

**Case 3: Detail Extraction**

1. Fixed square mode, large size (150px)
2. Position on area of interest
3. Double-click to extract specific detail

**Troubleshooting**

**Image won't load**

* **Check** that file resources/carte1870.png exists
* **Supported formats**: PNG, JPG, BMP only
* **Path**: must be relative to execution directory

**Grid poorly positioned**

* Use **arrows** to adjust finely
* **Fixed square mode** allows more positioning freedom
* Readjust **after** size change

**Files not created**

* Check **write permissions** in folder
* **Sufficient disk space**
* **Special characters** in prefix may cause problems

**Slow performance**

* **Very large images**: reduce size before cutting
* **Many cells**: processing may take time
* **Close** other applications to free memory

**Known Limitations**

**Technical**

* **Maximum image size** limited by available memory
* **Number of cells**: maximum 50×50 by interface
* **Formats**: reading limited to PNG, JPG, BMP

**Interface**

* **File dialog**: currently hardcoded to carte1870.png
* **Single undo**: no operation cancellation
* **Resizing**: fixed size window

**Features**

* **Rotation**: no integrated image rotation
* **Filters**: no image processing
* **Batch**: single image processing at a time

**Technical Architecture**

**Modular Structure**

* **imagecutter.lpr**: main loop and Raylib initialization
* **init.pas**: data structures and base functions
* **gui\_interface.pas**: user interface and event handling
* **gridrenderer.pas**: graphic rendering of image and grid

**Dependencies**

* **Raylib**: graphic display and window management
* **RayGUI**: user interface components
* **Free Pascal**: compiler and standard libraries

**Performance**

* **60 FPS**: smooth interface refresh
* **Optimized calculations**: real-time scaling
* **Memory**: automatic texture cleanup