PyPotteryLayout

Release 1.0.0

Enzo Cocca

Sep 19, 2025

USER GUIDE

Welcome to PyPotteryLayout's documentation! PyPotteryLayout is a comprehensive Python desktop application designed for creating publication-quality layouts of archaeological pottery and artifact images.

CHAPTER

ONE

OVERVIEW

PyPotteryLayout provides automated grid and puzzle layouts with captions, scale bars, and metadata integration for academic publications. It offers both a user-friendly GUI and a powerful API for programmatic use.

1.1 Key Features

- **Multiple Layout Modes**: Grid, Puzzle (optimized), Masonry, and Manual positioning
- **Automatic Image Scaling**: Optimizes image sizes to fill pages without empty spaces
- **Multi-page Support**: Handles large collections across multiple pages
- **Metadata Integration**: Excel/CSV support for captions and custom sorting
- **Scale Bar Generation**: Automatic scale bars with configurable measurements
- **Multiple Export Formats**: PDF, SVG (editable), JPEG with professional quality
- **Cross-platform**: Works on Windows, macOS, and Linux

CHAPTER

TWO

DOCUMENTATION CONTENTS

2.1 Installation Guide

This guide will walk you through installing PyPotteryLayout on your system.

2.1.1 System Requirements

- Python 3.7 or higher
- Operating System: Windows, macOS, or Linux
- At least 4GB RAM (8GB recommended for large image collections)
- 500MB free disk space

2.1.2 Installing from Source

1. **Clone the repository**:

```
git clone https://github.com/enzococca/PyPotteryLayout.git
cd PyPotteryLayout
```

2. **Create a virtual environment** (recommended):

```
python -m venv venv

# On Windows:
venv\Scripts\activate

# On macOS/Linux:
source venv/bin/activate
```

3. **Install dependencies**:

```
pip install -r requirements.txt
```

2.1.3 Dependencies

PyPotteryLayout requires the following Python packages:

- **Pillow** (>=9.0.0): Image processing and manipulation
- **tkinter**: GUI framework (usually included with Python)

- **rectpack** (>=0.2.2): Rectangle packing algorithms for puzzle layouts
- **openpyxl** (>=3.0.0): Excel file reading for metadata
- **reportlab** (optional): PDF generation with vector graphics
- **cairosvg** (optional): SVG to other format conversion

2.1.4 Installing Optional Dependencies

For enhanced PDF and SVG support:

```
pip install reportlab cairosvg
```

For development and documentation:

```
pip install sphinx sphinx-rtd-theme sphinx-autodoc-typehints
```

2.1.5 Platform-Specific Notes

Windows

- Ensure Python is added to your PATH during installation
- You may need to install Visual C++ Redistributable for some dependencies

macOS

- Install Python via Homebrew for best compatibility:

  ```
  brew install python
  ```

- Tkinter should be included with the Python installation

Linux

- Install tkinter if not present:

  ```
  # Ubuntu/Debian:
  sudo apt-get install python3-tk

  # Fedora:
  sudo dnf install python3-tkinter

  # Arch:
  sudo pacman -S tk
  ```

2.1.6 Verifying Installation

To verify that PyPotteryLayout is properly installed:

```
python -c "import backend_logic; print('Backend OK')"
python -c "import gui_app; print('GUI OK')"
```

If both commands print “OK”, the installation is successful.

2.1.7 Running the Application

To launch the GUI application:

```
python gui_app.py
```

2.1.8 Building Standalone Executable

To create a standalone executable for distribution:

1. **Install PyInstaller**:

   ```
   pip install pyinstaller
   ```

2. **Run the build script**:

   ```
   python build_exe.py
   ```

The executable will be created in the `dist` folder.

2.1.9 Troubleshooting

Common Issues

ImportError: No module named 'PIL'
: Install Pillow: `pip install Pillow`

ImportError: No module named '_tkinter'
: Install tkinter for your platform (see Platform-Specific Notes)

Font-related warnings
: The application will use fallback fonts if system fonts are not found. This is normal and doesn't affect functionality.

Memory errors with large images
: Try reducing the scale factor or processing fewer images at once.

Getting Help

If you encounter issues:

1. Check the GitHub Issues page
2. Ensure all dependencies are correctly installed
3. Try running in a fresh virtual environment

2.2 Quick Start Guide

This guide will help you create your first pottery catalog layout in minutes.

2.2.1 Step 1: Prepare Your Images

1. **Organize your images** in a single folder
2. **Supported formats**: JPG, PNG, GIF, BMP, TIFF
3. **Naming convention**: Use descriptive filenames (they'll appear in captions)

Tip

Use consistent naming like `pottery_001.jpg`, `pottery_002.jpg` for natural sorting

2.2.2 Step 2: Launch the Application

Run the application from the command line:

```
python gui_app.py
```

The main window will open with two tabs:

- **Input/Output**: File selection and export settings
- **Layout Configuration**: Layout mode and parameters

2.2.3 Step 3: Select Your Images

1. Click **Browse...** next to "Images Folder"
2. Navigate to your images folder
3. Select the folder containing your pottery images

The preview panel will immediately load thumbnails of your images.

2.2.4 Step 4: Choose Layout Mode

In the Layout Configuration tab, select your preferred mode:

Grid Layout (Default)

- Images arranged in rows and columns
- Best for uniform presentations
- Set rows and columns in Preview Controls

Puzzle Layout

- Optimized space utilization
- Automatically fits images like a puzzle
- Great for mixed image sizes

Masonry Layout

- Pinterest-style vertical columns
- Images flow naturally
- Good for varied aspect ratios

Manual Layout

- Drag and drop images in preview
- Full control over positioning
- Perfect for custom arrangements

2.2.5 Step 5: Configure Basic Settings

Essential settings to adjust:

Page Settings

- **Page Format**: A4 or A3 (default: A4)
- **Margins**: Space from page edge (default: 50px)
- **Image Spacing**: Gap between images (default: 10px)
- **Images per Page**: 0 for automatic, or specify exact number

Scale Settings

- **Scale Factor**: Resize all images (default: 0.4x)
- **Auto-scaling**: Enabled when Images per Page > 0

> **Note**
>
> With auto-scaling, images automatically resize to fill the page optimally

2.2.6 Step 6: Add Captions and Scale Bar

In the Details section:

Captions

- Check "Add Captions" to include image filenames
- Load metadata Excel for custom captions
- Adjust font size and padding

Scale Bar

- Check "Add Scale Bar" for archaeological scale
- Set length in centimeters (default: 5cm)
- Configure pixels per cm based on your images

2.2.7 Step 7: Export Your Layout

1. Click **Save as...** to choose output location
2. Select format:
   - **PDF**: Best for printing and sharing
   - **SVG**: Fully editable vector format
   - **JPEG**: Standard image format
3. Click **Export Layout**

The process will run and save to the `export` folder.

2.2.8 Complete Example Workflow

Here's a typical workflow for creating a pottery catalog:

```
1. Images folder: /Documents/pottery_photos/
   ├── bowl_001.jpg
   ├── bowl_002.jpg
   └── jar_001.jpg

2. Settings:
   - Mode: Grid (3 columns × 4 rows)
   - Page: A4
   - Margins: 50px
   - Scale: 0.5x
   - Add Scale Bar: Yes (5cm)
   - Add Captions: Yes

3. Output: catalog.pdf
   - Professional layout
   - Scale bar at bottom
   - Captions under each image
```

2.2.9 Tips for Best Results

1. **Consistent Image Quality**: Use images with similar resolution
2. **Batch Processing**: Process similar artifacts together
3. **Test First**: Try with a few images before processing large collections
4. **Preview Usage**: Use preview to test settings before export
5. **Save Settings**: Keep note of successful configurations

2.2.10 Using the Preview Panel

The preview panel shows real-time updates:

- **Navigation**: Use Previous/Next buttons to navigate pages
- **Page Info**: Shows current page and image count
- **Auto-scaling**: Displays calculated scale factor
- **Manual Mode**: Drag images to reposition

2.2.11 Keyboard Shortcuts

While the application doesn't have extensive shortcuts, you can:

- **Tab**: Navigate between controls
- **Enter**: Apply values in entry fields
- **Space**: Toggle checkboxes when focused

2.2.12 Next Steps

Now that you've created your first layout:

1. Explore *Layout Modes Guide* for advanced layout options
2. Learn about *Metadata and Captions Guide* for custom captions
3. Read *Export Formats Guide* for format-specific tips
4. Check *Tips and Best Practices* for professional results

2.3 User Interface Guide

This guide provides a detailed overview of the PyPotteryLayout graphical user interface.

2.3.1 Main Window Layout

The application window is divided into three main areas:

1. **Control Panel** (Left): Contains all settings in tabbed sections
2. **Preview Panel** (Right): Real-time preview of your layout
3. **Status Bar** (Bottom): Shows progress and messages

2.3.2 Control Panel Sections

The control panel uses tabs to organize features:

Input/Output Tab

Images Folder

- Select the folder containing your pottery images
- Displays the selected path
- Updates preview automatically

Metadata File (Optional)

- Load Excel file with image metadata
- Must have filenames in first column
- Additional columns for captions and sorting

Sort Images

- Primary sorting field
- Secondary sorting field (hierarchical)
- Options: Alphabetical, Natural, Random, or metadata fields

Output File

- Choose save location and filename
- Extension determines format (.pdf, .svg, .jpg)
- Saves to `export` subfolder

Output Settings

- DPI: Resolution for raster exports (default: 300)
- Page Format: A4, A3, HD, 4K, Letter, or Custom
- Custom Size: Width × Height in pixels

Layout Configuration Tab

Layout Mode

- Grid: Regular rows and columns
- Puzzle: Optimized packing
- Masonry: Vertical columns
- Manual: Drag-and-drop positioning

Grid Settings (Grid mode only)

- Rows: Number of rows per page
- Columns: Number of columns per row

Spacing and Margins

- Margins (px): Distance from page edges
- Image Spacing (px): Gap between images
- Show Margin Border: Visual guide in output

Scale Settings

- Scale Factor: Resize multiplier (0.1-5.0)
- Images per Page: Auto-scaling trigger
- Auto-scale indicator in preview

Caption Settings

- Add Captions: Include text under images
- Font Size: Caption text size
- Caption Padding: Space around text

Scale Bar Settings

- Add Scale Bar: Include measurement reference
- Bar Length (cm): Physical measurement
- Pixels per cm: Calibration value

2.3.3 Preview Panel Features

The preview panel provides immediate visual feedback:

Navigation Controls

Located at the top of the preview:

- **Page Counter**: Shows "Page X of Y"
- **Previous/Next Buttons**: Navigate multi-page layouts
- **Status Text**: Shows loading info and auto-scale factor

Preview Canvas

The main preview area shows:

- **White background**: Represents the page
- **Margin guides**: Dotted lines (if enabled)
- **Image thumbnails**: Scaled representations
- **Layout preview**: Real-time arrangement

Manual Mode Features

When Manual layout is selected:

1. **Drag to Move**: Click and drag images
2. **Visual Feedback**: Selected image highlighted
3. **Position Memory**: Preserves custom positions
4. **Reset Option**: Return to automatic layout

2.3.4 Working with Different Tabs

Efficient Tab Usage

1. **Start with Input/Output**: Set files and basic options
2. **Move to Layout Configuration**: Fine-tune appearance
3. **Return to Input/Output**: Final export settings

Tab Memory

The application remembers your settings between tabs:

- Changes are preserved when switching
- Preview updates reflect all settings
- Export uses current configuration

2.3.5 Status Messages and Progress

The status bar provides feedback:

During Loading

- "Loading images from: [path]"
- "Loaded X images"

- "Loading metadata. . . "

During Processing

- Progress percentage
- Current operation
- Completion message

Error Messages

- Clear error descriptions
- Suggested solutions
- File path information

2.3.6 Advanced Interface Features

Responsive Design

- Preview scales to window size
- Controls adjust to content
- Scrollable sections for small screens

Value Validation

Entry fields validate input:

- Numeric fields accept only numbers
- Ranges enforced (e.g., scale 0.1-5.0)
- Invalid input highlighted

Dynamic Updates

Changes trigger immediate updates:

- Preview refreshes on setting changes
- Page count updates with layout
- Auto-scale calculates in real-time

Context-Sensitive Controls

Controls show/hide based on mode:

- Grid settings only in Grid mode
- Manual controls only in Manual mode
- Relevant options stay visible

2.3.7 Tooltips and Help

Hover Information

Key controls include tooltips:

- Scale factor explanation
- Metadata file format
- Export format details

Visual Indicators

- **Checkboxes**: Clear on/off states
- **Radio buttons**: Exclusive selection
- **Entry fields**: Focused highlighting

Color Coding

- **Active controls**: Standard colors
- **Disabled controls**: Grayed out
- **Error states**: Red highlighting

2.3.8 Best Practices for Interface Use

Workflow Tips

1. **Preview First**: Always check preview before export
2. **Test Settings**: Try different modes with preview
3. **Save Configurations**: Note successful settings

Performance Tips

- **Large Collections**: Process in batches
- **High Resolution**: Reduce scale for preview
- **Multiple Pages**: Use page navigation

2.3.9 Troubleshooting Interface Issues

Common Problems

Preview not updating

- Check if images loaded successfully
- Verify folder contains valid images
- Try refreshing with different settings

Controls not responding

- Ensure process isn't running
- Check for error messages

- Restart application if needed

Layout looks wrong

- Verify scale factor is appropriate
- Check margins aren't too large
- Ensure images fit page size

2.3.10 Keyboard Navigation

The interface supports standard keyboard navigation:

- **Tab**: Move forward through controls
- **Shift+Tab**: Move backward
- **Space**: Toggle checkboxes/radio buttons
- **Enter**: Apply entry field values
- **Arrow keys**: Navigate dropdowns

2.3.11 Accessibility Features

The interface includes accessibility support:

- **High contrast**: Clear control boundaries
- **Font scaling**: Readable at various sizes
- **Logical tab order**: Sequential navigation
- **Status announcements**: Clear feedback

2.4 Layout Modes Guide

PyPotteryLayout offers four distinct layout modes, each optimized for different presentation needs.

2.4.1 Grid Layout

The Grid layout arranges images in a regular matrix of rows and columns.

Features

- **Regular arrangement**: Uniform rows and columns
- **Center alignment**: Rows centered on page
- **Flexible sizing**: Adapts to image dimensions
- **Multi-page support**: Automatic pagination

Configuration

Essential Settings:

- **Rows**: Number of rows per page (1-10)
- **Columns**: Maximum images per row (1-10)

- **Auto-fit**: Images smaller than cell size center automatically

When to Use:

- Catalog presentations requiring uniformity
- Comparative displays of similar artifacts
- Academic publications with standard formats
- When consistent spacing is important

Example Configuration:

```
Mode: Grid
Rows: 4
Columns: 3
Result: Up to 12 images per page in a 4×3 grid
```

Tips for Grid Layout

1. **Aspect Ratios**: Works best with similar aspect ratios
2. **Scaling**: Use consistent scale factor for uniform appearance
3. **Spacing**: Increase spacing for visual breathing room
4. **Page Breaks**: Set images per page for controlled pagination

2.4.2 Puzzle Layout

The Puzzle layout uses rectangle packing algorithms to maximize space utilization.

Features

- **Optimized packing**: Minimal wasted space
- **Mixed sizes**: Handles varied image dimensions
- **Automatic arrangement**: Algorithm determines positions
- **Efficient use**: Fits more images per page

Configuration

How it Works:

The puzzle algorithm (MaxRectsBssf) finds optimal positions by:

1. Sorting images by area
2. Finding best position for each image
3. Minimizing gaps between images
4. Creating new pages when full

When to Use:

- Collections with varied image sizes
- Maximizing images per page

- Creating compact catalogs
- When space efficiency matters most

Advantages:

- 20-30% more efficient than grid
- Handles mixed orientations well
- Reduces page count

Tips for Puzzle Layout

1. **Similar Scales**: Pre-scale images for better packing
2. **Spacing**: Minimal spacing works best (5-10px)
3. **Page Size**: Larger pages show algorithm benefits
4. **Sorting**: Size-based sorting improves results

2.4.3 Masonry Layout

The Masonry layout arranges images in vertical columns, like Pinterest.

Features

- **Vertical flow**: Images stack in columns
- **Dynamic heights**: Preserves aspect ratios
- **Balanced columns**: Distributes images evenly
- **Natural appearance**: Organic, magazine-style layout

Configuration

Column Settings:

- Uses the "Columns" value from grid settings
- Default: 3 columns
- Range: 2-6 columns recommended

Algorithm:

1. Images placed in shortest column
2. Column heights tracked
3. New page when all columns full
4. Maintains reading order

When to Use:

- Web-style presentations
- Mixed portrait/landscape images
- Modern, dynamic layouts
- When variety is desired

Tips for Masonry Layout

1. **Column Count**: 3-4 columns usually optimal
2. **Image Order**: Important images first
3. **Consistent Widths**: All images sized to column width
4. **Vertical Space**: Allows varied heights

2.4.4 Manual Layout

The Manual layout provides complete control through drag-and-drop positioning.

Features

- **Full control**: Place images anywhere
- **Drag and drop**: Interactive positioning
- **Overlap support**: Images can overlap
- **Creative freedom**: No algorithmic constraints

How to Use Manual Mode

1. **Enable**: Select “Manual” from layout modes
2. **Preview Panel**: Becomes interactive
3. **Drag Images**: Click and hold to move
4. **Position**: Drop at desired location
5. **Export**: Positions preserved exactly

Controls:

- **Left Click**: Select image
- **Drag**: Move selected image
- **Release**: Place at new position

When to Use:

- Custom artistic arrangements
- Specific positioning requirements
- Creating poster layouts
- When automation isn’t sufficient

Tips for Manual Layout

1. **Grid Snapping**: Mentally align to grid
2. **Margins**: Respect page margins
3. **Overlapping**: Use for creative effects
4. **Templates**: Save successful layouts

2.4.5 Layout Mode Comparison

Table 1: Layout Mode Comparison

Feature	Grid	Puzzle	Masonry	Manual
Space Efficiency	Moderate	High	Moderate	Variable
Control Level	Low	Low	Low	Complete
Best For	Catalogs	Mixed Sizes	Magazines	Art
Speed	Fast	Fast	Fast	Slow
Predictability	High	Moderate	Moderate	N/A

2.4.6 Choosing the Right Layout

Decision Factors

Choose Grid when:

- Images have similar dimensions
- Formal presentation required
- Comparing similar artifacts
- Standard academic format needed

Choose Puzzle when:

- Space is limited
- Image sizes vary significantly
- Maximum efficiency required
- Creating compact catalogs

Choose Masonry when:

- Modern presentation desired
- Mixed orientations present
- Web/digital output planned
- Visual flow important

Choose Manual when:

- Specific arrangement needed
- Creating poster/display
- Artistic presentation
- Full control required

2.4.7 Multi-Page Handling

All layouts support multi-page documents:

Automatic Pagination

- **Grid**: Based on rows × columns
- **Puzzle**: When bin is full
- **Masonry**: When columns filled
- **Manual**: Based on position

Images Per Page Setting

When set to a specific number:

1. Overrides automatic pagination
2. Enables auto-scaling
3. Consistent page density
4. Better for printing

Page Navigation

- Preview shows current page
- Navigate with arrow buttons
- Page count updates dynamically
- Export includes all pages

2.4.8 Advanced Layout Techniques

Combining Layouts

Process different image groups separately:

1. Archaeological finds: Grid layout
2. Site photos: Masonry layout
3. Special artifacts: Manual layout
4. Combine PDFs afterward

Layout Preprocessing

Prepare images for better layouts:

1. **Standardize sizes**: Batch resize similar artifacts
2. **Crop consistently**: Remove unnecessary backgrounds
3. **Orient properly**: Rotate before importing
4. **Group by type**: Process categories separately

Optimization Strategies

For Print:

- Higher margins (100-150px)
- Moderate spacing (15-20px)
- Consider bleed areas
- Test with printer

For Digital:

- Smaller margins (50px)
- Tighter spacing (5-10px)
- Screen-optimized sizes
- Consider aspect ratios

For Presentations:

- Larger scale factors
- More spacing
- Fewer per page
- Bold captions

2.5 Metadata and Captions Guide

This guide explains how to use metadata files to add rich captions and custom sorting to your pottery catalogs.

2.5.1 Understanding Metadata

Metadata provides additional information about your images:

- **Captions**: Descriptions, measurements, dates
- **Sorting fields**: Categories, periods, types
- **References**: Catalog numbers, locations
- **Attributes**: Material, technique, provenance

2.5.2 Metadata File Format

PyPotteryLayout accepts Excel (.xlsx) files with specific structure:

Basic Structure

Filename	Description	Period	Location	Catalog_No
bowl_001.jpg	Red-figure bowl	450-400 BCE	Athens	CAT-0234
jar_002.jpg	Storage amphora	500-450 BCE	Corinth	CAT-0235
plate_003.jpg	Black-glaze plate	400-350 BCE	Athens	CAT-0236

Requirements:

1. **First column**: Must contain image filenames
2. **Headers**: First row contains field names
3. **Matching**: Filenames must match exactly (case-insensitive)

2.5.3 Creating Metadata Files

Using Excel

1. Open Microsoft Excel or LibreOffice Calc
2. Create headers in row 1
3. Enter filenames in column A
4. Add metadata in subsequent columns
5. Save as .xlsx format

Using CSV (Alternative)

1. Create CSV with comma separation
2. Include headers as first line
3. Save with .csv extension
4. Import to Excel if needed

Example Templates

Archaeological Catalog:

```
Filename | Type | Period | Culture | Dimensions | Find_Location | Catalog_ID
pot1.jpg | Amphora | LBA | Minoan | H:45cm | Knossos | KN-1234
```

Museum Collection:

```
Filename | Accession | Artist | Date | Medium | Provenance | Condition
vase1.jpg | 2024.1.1 | Unknown | 450 BCE | Ceramic | Donated 1925 | Good
```

2.5.4 Using Metadata in Captions

Simple Captions

With "Add Captions" checked, metadata appears as:

```
filename.jpg
Field1: Value1
Field2: Value2
```

Formatting Captions

Caption appearance is controlled by:

- **Font Size**: Text size (8-24pt)
- **Caption Padding**: Space around text
- **First Line**: Filename (bold)
- **Subsequent Lines**: Metadata fields

Selective Fields

Not all metadata needs to appear in captions:

- Include display fields: Description, Date
- Exclude internal fields: Internal_ID, Notes
- Use meaningful headers: "Period" not "col_B"

2.5.5 Custom Sorting with Metadata

Primary and Secondary Sorting

PyPotteryLayout supports two-level hierarchical sorting:

Primary Sort:

- Main grouping criteria
- Options: Any metadata field

Secondary Sort:

- Sorting within groups
- Applied after primary sort

Sorting Examples

Chronological Catalog:

```
Primary: Period
Secondary: Type
Result: Images grouped by period, then by type within each period
```

Geographic Organization:

```
Primary: Region
Secondary: Site
Result: Images grouped by region, then by site
```

Typological Arrangement:

```
Primary: Category
Secondary: natural_name
Result: Images grouped by category, naturally sorted within
```

Special Sorting Options

- **alphabetical**: A-Z by filename
- **natural_name**: Handles numbers properly (1, 2, 10, not 1, 10, 2)
- **random**: Randomize order
- **none**: Maintain existing order

2.5.6 Advanced Metadata Techniques

Multi-line Captions

Create rich captions with multiple fields:

```
Metadata columns:
- Description: "Red-figure kylix"
- Artist: "Python Painter"
- Date: "350-340 BCE"
- Museum: "British Museum"

Result caption:
kylix_001.jpg
Description: Red-figure kylix
Artist: Python Painter
Date: 350-340 BCE
Museum: British Museum
```

Conditional Fields

Leave cells empty to exclude from captions:

- Blank cells don't appear
- Useful for varied collections
- Maintains clean appearance

Special Characters

Supports Unicode for special characters:

- Greek: alpha, beta, gamma, delta
- Measurements: diameter, plus/minus, approximately
- Symbols: dagger, double dagger, section, paragraph

2.5.7 Metadata Best Practices

File Naming Conventions

Recommended patterns:

```
Simple:       pot_001.jpg, pot_002.jpg
Descriptive:  amphora_corinth_01.jpg
```

(continues on next page)

(continued from previous page)

```
Catalog:      BM_1842_0728_784.jpg
Hierarchical: type_period_number.jpg
```

Data Organization

Essential Fields:

1. Identification (catalog number)
2. Classification (type, category)
3. Dating (period, specific date)
4. Measurements (height, diameter)
5. Provenance (findspot, collection)

Optional Fields:

1. Description (free text)
2. Condition (preservation state)
3. Bibliography (references)
4. Technical (fabric, technique)
5. Notes (additional info)

Data Validation

Ensure consistency:

- **Dates**: Use consistent format (BCE/CE or BC/AD)
- **Measurements**: Include units (cm, mm)
- **Names**: Standardize spelling
- **Categories**: Use controlled vocabulary

2.5.8 Troubleshooting Metadata Issues

Common Problems

Images not matching metadata:

- Check filename spelling exactly
- Remove file extensions if included
- Ensure no extra spaces
- Case doesn't matter

Metadata not loading:

- Verify Excel file format (.xlsx)
- Check first column has filenames
- Ensure headers in first row

- No merged cells

Sorting not working:

- Check field name matches exactly
- Remove special characters from headers
- Ensure data type consistency
- No empty header cells

Excel File Tips

Optimization:

1. Keep files under 10MB
2. Avoid complex formulas
3. Remove formatting/colors
4. Use simple sheet names

Compatibility:

- Save as .xlsx (not .xls)
- Avoid macros
- No password protection
- Single sheet preferred

2.5.9 Metadata Workflow Examples

Academic Publication

1. **Prepare Excel** with scholarly data
2. **Include fields**: Type, Date, Provenance, Museum number
3. **Sort by**: Period (primary), Type (secondary)
4. **Export with**: Full captions for reference

Museum Display

1. **Simple metadata**: Name, Date, Gallery location
2. **Large captions**: 16pt font for readability
3. **Sort by**: Gallery, then accession number
4. **Clean layout**: Minimal technical details

Digital Archive

1. **Comprehensive metadata**: All available fields
2. **Small captions**: Space-efficient
3. **Sort by**: Catalog number
4. **Include**: Database IDs for cross-reference

Field Excavation

1. **Field data**: Context, Level, Find number
2. **Technical info**: Fabric, Weight, Dimensions
3. **Sort by**: Context, then find number
4. **Document**: Stratigraphic relationships

2.6 Export Formats Guide

PyPotteryLayout supports multiple export formats, each optimized for different use cases.

2.6.1 PDF Export

The PDF format is ideal for printing and sharing finished catalogs.

Features

- **Multi-page support**: All pages in single file
- **Vector text**: Searchable and selectable
- **Embedded images**: Self-contained document
- **Print-ready**: Professional quality output
- **Universal compatibility**: Opens everywhere

PDF Settings

Resolution (DPI):

- 300 DPI: Standard print quality
- 600 DPI: High-quality printing
- 150 DPI: Screen viewing/web

Optimization:

- File size vs. quality balance
- Compression applied automatically
- Fonts embedded for consistency

Best Practices

1. **For Print:** * Use 300+ DPI * Include margin borders for trim * Test print single page first * Consider bleed requirements
2. **For Digital:** * 150 DPI usually sufficient * Smaller file sizes * Optimized for screen viewing * Consider page orientation

2.6.2 SVG Export

SVG creates fully editable vector graphics ideal for further editing.

Features

- **Fully editable**: Every element separate
- **Vector format**: Infinite scaling
- **External images**: Linked, not embedded
- **Layer support**: Organized structure
- **Software compatible**: Inkscape, Illustrator

SVG Structure

The exported SVG contains:

```
/export/project_name/
├── project_name.svg
└── images/
    ├── img_001_pottery.png
    ├── img_002_pottery.png
    └── ...
```

Layers Created:

1. Background layer
2. Images layer
3. Captions layer
4. Scale bar layer
5. Margin guides (if enabled)

Editing SVG Files

In Inkscape (Recommended):

1. Open SVG file
2. Ungroup elements if needed
3. Edit text directly
4. Move/resize images
5. Adjust colors/styles
6. Save as Inkscape SVG

In Adobe Illustrator:

1. Open SVG file
2. Release clipping masks
3. Edit as needed

4. Maintain links to images
5. Save as AI or SVG

SVG Advantages

- **Editability**: Full post-processing control
- **Scalability**: Resolution independent
- **Flexibility**: Modify any aspect
- **Integration**: Import into other designs
- **Archival**: Future-proof format

2.6.3 JPEG Export

JPEG format provides standard raster images for broad compatibility.

Features

- **Universal support**: Opens anywhere
- **Reasonable size**: Compressed format
- **Web-ready**: Direct upload capability
- **Simple sharing**: Email, social media

JPEG Settings

Quality Considerations:

- File size increases with DPI
- 300 DPI for print
- 96-150 DPI for web
- Compression affects quality

Multi-page Handling:

```
/export/catalog/
├── catalog_page_01.jpg
├── catalog_page_02.jpg
└── catalog_page_03.jpg
```

Best Use Cases

- Quick previews
- Web galleries
- Email attachments
- Social media sharing
- Documentation

2.6.4 PNG Export

While not directly offered in the menu, PNG can be achieved:

1. Export as JPEG
2. Higher quality than JPEG
3. Supports transparency
4. Larger file sizes

2.6.5 Export Location Structure

PyPotteryLayout organizes exports systematically:

Default Structure

```
/your_project/
├── export/
│   ├── catalog_2024/
│   │   ├── catalog_2024.svg
│   │   └── images/
│   ├── final_pdf/
│   │   └── final.pdf
│   └── README.txt
```

Organization Benefits:

- Clean project structure
- Version management
- Easy backup
- No file clutter

2.6.6 Export Workflow Tips

Pre-Export Checklist

1. [x] Preview looks correct
2. [x] All images loaded
3. [x] Captions displaying properly
4. [x] Scale bar calibrated
5. [x] Page size appropriate
6. [x] Output format selected

Format Selection Guide

Choose PDF when:

- Final output needed
- Printing required

- Sharing complete document
- Professional presentation

Choose SVG when:

- Further editing planned
- Need vector graphics
- Custom modifications required
- Creating templates

Choose JPEG when:

- Quick sharing needed
- Web upload planned
- File size matters
- Compatibility crucial

Post-Export Options

After PDF Export:

- Open in PDF reader
- Check all pages
- Verify image quality
- Print test page

After SVG Export:

- Open in vector editor
- Check layer structure
- Verify image links
- Make final adjustments

After JPEG Export:

- Review image quality
- Check file sizes
- Batch process if needed
- Upload or share

2.6.7 Advanced Export Techniques

Batch Processing

For multiple catalogs:

1. Process each group separately
2. Use consistent settings

3. Organize by export folders
4. Maintain naming convention

Mixed Format Strategy

Create multiple formats:

1. **SVG**: Master editable version
2. **PDF**: Distribution version
3. **JPEG**: Preview/web version

Resolution Guidelines

Table 2: DPI Recommendations

Use Case	Recommended DPI	File Size
Professional Print	600	Large
Standard Print	300	Medium
Screen/Web	150	Small
Thumbnail	72	Minimal

2.6.8 Export Troubleshooting

Common Issues

Large file sizes:

- Reduce DPI setting
- Scale images before import
- Use appropriate format
- Consider compression

Missing images in SVG:

- Keep images folder together
- Don't rename image files
- Use relative paths
- Verify links intact

Poor print quality:

- Increase DPI to 300+
- Check original image quality
- Verify scale settings
- Test print settings

Export fails:

- Check disk space

- Verify write permissions
- Close other applications
- Try different location

2.6.9 Platform-Specific Notes

Windows

- Exports to Documents/export by default
- Paths use backslashes
- May need admin rights for some locations

macOS

- Exports to ~/Documents/export
- Maintains file attributes
- Preview app shows multi-page PDFs

Linux

- Exports to ~/Documents/export
- File permissions preserved
- Various PDF viewers available

2.6.10 Format Conversion

Converting Between Formats

SVG to PDF:

- Use Inkscape export
- Maintain vector quality
- Embed fonts

PDF to Images:

- Use PDF software
- Set desired DPI
- Export pages separately

SVG to PNG:

- Higher quality than JPEG
- Supports transparency
- Larger files

2.7 Tips and Best Practices

This guide provides professional tips for creating publication-quality pottery catalogs.

2.7.1 Image Preparation

Before Import

Standardize Your Images:

1. **Consistent Background**: Use neutral gray or white
2. **Uniform Lighting**: Avoid harsh shadows
3. **Same Distance**: Maintain camera-to-object distance
4. **Orientation**: Rotate images correctly beforehand
5. **Color Correction**: Apply before importing

Optimal Image Specifications:

- **Resolution**: 300 DPI minimum for print
- **Format**: JPEG or PNG (avoid compression artifacts)
- **Size**: 2000-4000px on longest side
- **Color Space**: sRGB for consistency
- **File Size**: 1-5 MB per image typical

Batch Processing Tips

Using Image Editing Software:

```
Photoshop Actions:
1. Create action for standard processing
2. Include: Resize, color correct, sharpen
3. Batch apply to folder
4. Export with consistent settings

ImageMagick Command:
convert *.jpg -resize 3000x3000 -quality 95 output_%03d.jpg
```

Naming Conventions:

```
Good:
- pottery_001_bowl_athens.jpg
- pottery_002_jar_corinth.jpg

Better:
- BM2024_001_typA_period3.jpg
- BM2024_002_typB_period3.jpg
```

2.7.2 Professional Layout Design

Composition Principles

Visual Balance:

1. **Rule of Thirds**: Align important elements
2. **White Space**: Don't overcrowd pages
3. **Consistency**: Maintain style throughout
4. **Hierarchy**: Size indicates importance
5. **Flow**: Guide the eye naturally

Page Density:

- **Academic**: 6-12 images per A4 page
- **Exhibition**: 4-6 images per page
- **Detail Study**: 1-4 images per page
- **Overview**: 12-20 thumbnails per page

Color Management

For Print:

1. Convert to CMYK if required
2. Calibrate monitor
3. Use color profiles
4. Test print colors
5. Adjust for paper type

For Digital:

1. Maintain sRGB
2. Check on multiple screens
3. Consider accessibility
4. Test contrast ratios

2.7.3 Scale Bar Guidelines

Calibration

Accurate Measurement:

1. **Photograph with ruler**: Include in one shot
2. **Measure in image**: Count pixels per cm
3. **Calculate ratio**: Pixels/physical cm
4. **Apply consistently**: Same scale for session
5. **Document settings**: Record for future

Standard Lengths:

- Small objects: 1-2 cm
- Medium pottery: 5 cm
- Large vessels: 10 cm
- Architectural: 50 cm or 1 m

Placement

Best Practices:

- Bottom left or right corner
- Consistent position throughout
- Clear background behind bar
- Adequate padding around
- Same size across pages

2.7.4 Caption Excellence

Writing Effective Captions

Information Hierarchy:

```
Primary (Bold):
Object Type/Name

Secondary:
Period/Date
Provenance/Location

Tertiary:
Dimensions
Catalog Number
Additional Notes
```

Style Guidelines:

1. **Consistency**: Same format throughout
2. **Brevity**: Essential information only
3. **Clarity**: Avoid jargon when possible
4. **Accuracy**: Double-check all data
5. **Completeness**: Include key identifiers

Academic Standards

Follow disciplinary conventions:

- Archaeology: Site, context, date, type
- Art History: Artist, title, date, medium

- Museums: Accession, culture, period
- Conservation: Condition, treatment, date

2.7.5 Performance Optimization

Large Collections

Strategies for 100+ images:

1. **Process in batches**: 50 images at a time
2. **Reduce preview quality**: Scale at 0.2-0.3x
3. **Pre-process images**: Resize before import
4. **Use efficient formats**: JPEG over PNG
5. **Clear memory**: Restart between batches

System Requirements

Recommended Specifications:

- **RAM**: 8GB minimum, 16GB optimal
- **Storage**: SSD for faster processing
- **CPU**: Multi-core for image operations
- **Display**: 1920×1080 or higher
- **Graphics**: Dedicated GPU helpful

Memory Management

Tips for smooth operation:

1. Close unnecessary applications
2. Process smaller batches
3. Use lower scale factors initially
4. Export incrementally
5. Monitor system resources

2.7.6 Quality Assurance

Pre-Export Checklist

Visual Review:

- [ ] All images displaying correctly
- [ ] Captions readable and accurate
- [ ] Scale bar properly sized
- [ ] Margins consistent
- [ ] No overlapping elements

- [ ] Page breaks logical

Technical Check:

- [ ] Resolution appropriate for use
- [ ] File format correct
- [ ] Color space verified
- [ ] Font embedding confirmed
- [ ] File size reasonable

Print Preparation

Professional Printing:

1. **Bleed**: Add 3mm if required
2. **Color**: Convert to CMYK
3. **Resolution**: Minimum 300 DPI
4. **Fonts**: Embed or outline
5. **Proof**: Request color proof

2.7.7 Common Mistakes to Avoid

Layout Errors

Avoid:

- Overcrowding pages
- Inconsistent spacing
- Mixed scales without reason
- Poor quality images
- Missing scale references

Solutions:

- Use white space effectively
- Maintain consistent gaps
- Group similar scales
- Pre-process all images
- Always include scale bar

Technical Issues

Common Problems:

1. **Memory errors**: Reduce batch size
2. **Slow processing**: Lower preview scale
3. **Export failures**: Check disk space

4. **Font issues**: Install system fonts
5. **Color shifts**: Verify color profiles

2.7.8 Workflow Optimization

Efficient Project Pipeline

```
1. Planning
   ├── Define requirements
   ├── Choose format/size
   └── Set quality standards

2. Preparation
   ├── Photograph objects
   ├── Process images
   └── Prepare metadata

3. Layout
   ├── Import and arrange
   ├── Add captions/scale
   └── Review preview

4. Export
   ├── Choose format
   ├── Set quality
   └── Generate output

5. Quality Control
   ├── Review output
   ├── Test print
   └── Final adjustments
```

Time-Saving Tips

1. **Create templates**: Save successful settings
2. **Batch operations**: Process similar items together
3. **Keyboard shortcuts**: Learn interface navigation
4. **Presets**: Document standard configurations
5. **Automation**: Use Excel formulas for metadata

2.7.9 Advanced Techniques

Multi-Catalog Projects

Managing Large Projects:

1. Divide by category/period
2. Process each separately
3. Maintain consistent style

4. Combine PDFs if needed
5. Create master index

Custom Modifications

Post-Processing SVG:

1. Open in Inkscape
2. Adjust individual elements
3. Add annotations
4. Include drawings/plans
5. Export final version

Integration with Other Tools

Workflow Integration:

- **Photography**: Lightroom → Export → PyPotteryLayout
- **Database**: Export metadata → Excel → PyPotteryLayout
- **Publication**: PyPotteryLayout → SVG → InDesign
- **Archive**: PyPotteryLayout → PDF/A for preservation

2.8 API Reference

This section contains the complete API documentation for PyPotteryLayout modules.

2.8.1 backend_logic module

Backend logic refactored and consolidated from notebook helpers.

Provides functions for loading images/metadata, placing images on pages in grid or puzzle modes, adding captions and scale bars, and saving output.

backend_logic.**get_page_dimensions_px**(*size_name_or_custom*, *custom_size_str=None*)

backend_logic.**get_font**(*size*)

Try to load common TTF fonts, fallback to default.

backend_logic.**get_metadata_headers**(*filepath*)

Get column headers from Excel metadata file for GUI dropdown options.

backend_logic.**load_metadata**(*filepath*, *status_callback=<built-in function print>*)

backend_logic.**load_images_with_info**(*folder_path*, *status_callback=<built-in function print>*)

backend_logic.**natural_sort_key**(*s*)

backend_logic.**sort_images_hierarchical**(*image_data*, *primary_sort*, *secondary_sort*, *metadata*, *status_callback=<built-in function print>*)

Sorts images with hierarchical two-level sorting.

Parameters

- **image_data** – Lista di dati immagine
- **primary_sort** – Campo di ordinamento primario
- **secondary_sort** – Campo di ordinamento secondario (può essere 'none', 'random', 'alphabetical', 'natural_name', o un campo metadati)
- **metadata** – Dizionario metadati
- **status_callback** – Function for status messages

backend_logic.**create_scale_bar**(*target_cm*, *pixels_per_cm*, *scale_factor*, *status_callback=<built-in function print>*)

backend_logic.**calculate_optimal_scale**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *images_per_page*, *mode*, *grid_size=None*)

Calculate optimal scale factor to fill the page with the given number of images.

backend_logic.**scale_images**(*image_data*, *scale_factor*, *status_callback=<built-in function print>*)

backend_logic.**add_captions_to_images**(*image_data*, *metadata*, *font_size*, *caption_padding*, *status_callback=<built-in function print>*)

backend_logic.**place_images_grid**(*image_data*, *page_size_px*, *grid_size*, *margin_px*, *spacing_px*, *images_per_page=0*, *status_callback=<built-in function print>*)

backend_logic.**place_images_puzzle**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *images_per_page=0*, *status_callback=<built-in function print>*)

backend_logic.**place_images_manual**(*image_data*, *page_size_px*, *margin_px*, *manual_positions*, *scale_factor*, *status_callback=<built-in function print>*, *images_per_page=0*)

Place images based on manual drag-and-drop positions with multi-page support.

backend_logic.**place_images_masonry**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *columns=3*, *images_per_page=0*, *status_callback=<built-in function print>*)

Place images in masonry layout (Pinterest-style vertical columns).

backend_logic.**draw_margin_border**(*page*, *margin_px*, *status_callback=<built-in function print>*)

Draw a border frame to visualize page margins.

backend_logic.**save_output**(*pages*, *output_file*, *output_dpi=300*, *status_callback=<built-in function print>*)

Save output pages to file, supporting PNG, JPG, PDF, and SVG formats.

backend_logic.**create_multipage_svg**(*all_image_positions*, *page_size_px*, *params*, *output_dir*, *metadata=None*, *status_callback=<built-in function print>*)

Create multi-page SVG optimized for Inkscape.

backend_logic.**create_lightweight_editable_svg**(*image_positions*, *page_size_px*, *params*, *output_dir*, *metadata=None*, *status_callback=<built-in function print>*)

Create lightweight SVG with external image references.

backend_logic.**create_editable_svg_layout_fixed**(*image_positions*, *page_size_px*, *params*, *output_dir*, *metadata=None*, *status_callback=<built-in function print>*)

Create SVG with separate, editable elements using external image files.

backend_logic.**create_editable_layout_positions_grid**(*image_data*, *page_size_px*, *grid_size*, *margin_px*, *spacing_px*, *params*, *metadata=None*, *status_callback=<built-in function print>*)

Calculate positions for grid layout, returning position data for editable output.

backend_logic.**save_editable_output**(*image_positions*, *page_size_px*, *output_file*, *params*, *metadata=None*, *status_callback=<built-in function print>*)

Save lightweight editable SVG output.

backend_logic.**run_layout_process**(*params*, *status_callback=<built-in function print>*)

Main orchestrator function that runs the complete layout process.

Core Functions

Image Loading and Processing

backend_logic.**load_images_with_info**(*folder_path*, *status_callback=<built-in function print>*)

backend_logic.**scale_images**(*image_data*, *scale_factor*, *status_callback=<built-in function print>*)

backend_logic.**calculate_optimal_scale**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *images_per_page*, *mode*, *grid_size=None*)

Calculate optimal scale factor to fill the page with the given number of images.

Metadata Handling

backend_logic.**load_metadata**(*filepath*, *status_callback=<built-in function print>*)

backend_logic.**get_metadata_headers**(*filepath*)

Get column headers from Excel metadata file for GUI dropdown options.

backend_logic.**sort_images_hierarchical**(*image_data*, *primary_sort*, *secondary_sort*, *metadata*, *status_callback=<built-in function print>*)

Sorts images with hierarchical two-level sorting.

Parameters

- **image_data** – Lista di dati immagine
- **primary_sort** – Campo di ordinamento primario
- **secondary_sort** – Campo di ordinamento secondario (può essere 'none', 'random', 'alphabetical', 'natural_name', o un campo metadati)
- **metadata** – Dizionario metadati
- **status_callback** – Function for status messages

Layout Functions

backend_logic.**place_images_grid**(*image_data*, *page_size_px*, *grid_size*, *margin_px*, *spacing_px*, *images_per_page=0*, *status_callback=<built-in function print>*)

backend_logic.**place_images_puzzle**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *images_per_page=0*, *status_callback=<built-in function print>*)

backend_logic.**place_images_masonry**(*image_data*, *page_size_px*, *margin_px*, *spacing_px*, *columns=3*, *images_per_page=0*, *status_callback=<built-in function print>*)

Place images in masonry layout (Pinterest-style vertical columns).

backend_logic.**place_images_manual**(*image_data*, *page_size_px*, *margin_px*, *manual_positions*, *scale_factor*, *status_callback=<built-in function print>*, *images_per_page=0*)

Place images based on manual drag-and-drop positions with multi-page support.

Caption and Scale Bar

backend_logic.**add_captions_to_images**(*image_data*, *metadata*, *font_size*, *caption_padding*, *status_callback=<built-in function print>*)

backend_logic.**create_scale_bar**(*target_cm*, *pixels_per_cm*, *scale_factor*, *status_callback=<built-in function print>*)

backend_logic.**draw_margin_border**(*page*, *margin_px*, *status_callback=<built-in function print>*)

Draw a border frame to visualize page margins.

Export Functions

backend_logic.**save_output**(*pages*, *output_file*, *output_dpi=300*, *status_callback=<built-in function print>*)

Save output pages to file, supporting PNG, JPG, PDF, and SVG formats.

backend_logic.**save_editable_output**(*image_positions*, *page_size_px*, *output_file*, *params*, *metadata=None*, *status_callback=<built-in function print>*)

Save lightweight editable SVG output.

backend_logic.**create_lightweight_editable_svg**(*image_positions*, *page_size_px*, *params*, *output_dir*, *metadata=None*, *status_callback=<built-in function print>*)

Create lightweight SVG with external image references.

Main Process

backend_logic.**run_layout_process**(*params*, *status_callback=<built-in function print>*)

Main orchestrator function that runs the complete layout process.

Utility Functions

backend_logic.**get_page_dimensions_px**(*size_name_or_custom*, *custom_size_str=None*)

backend_logic.**get_font**(*size*)

Try to load common TTF fonts, fallback to default.

backend_logic.**natural_sort_key**(*s*)

2.8.2 gui_app module

class gui_app.**LayoutApp**

Bases: Tk

__init__()

Return a new top level widget on screen SCREENNAME. A new Tcl interpreter will be created. BASENAME will be used for the identification of the profile file (see readprofile). It is constructed from sys.argv[0] without extensions if None is given. CLASSNAME is the name of the widget class.

LayoutApp Class

class gui_app.**LayoutApp**

Bases: Tk

Main GUI Application Class

The LayoutApp class inherits from tk.Tk and provides the complete graphical interface for PyPotteryLayout. It manages:

- User input through various widgets

- Preview panel with real-time updates
- File selection dialogs
- Layout parameter configuration
- Process execution in separate threads

Key Methods

_create_widgets()

_update_preview()
Update the preview canvas with current layout settings.

_start_process()

_run_backend_in_thread(*params*)

Preview Methods

_preview_grid_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview grid layout with image thumbnails.

_preview_puzzle_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview puzzle/optimized layout with image thumbnails.

_preview_masonry_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview masonry layout with image thumbnails.

_preview_manual_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview manual layout with drag-and-drop functionality.

Manual Layout Methods

_enable_manual_mode()
Enable drag-and-drop functionality in preview canvas.

_on_drag_start(*event*)
Handle drag start for manual positioning.

_on_drag_motion(*event*)
Handle drag motion for manual positioning.

_on_drag_release(*event*)
Handle drag release for manual positioning.

__init__()
Return a new top level widget on screen SCREENNAME. A new Tcl interpreter will be created. BASENAME will be used for the identification of the profile file (see readprofile). It is constructed from sys.argv[0] without extensions if None is given. CLASSNAME is the name of the widget class.

_create_widgets()

_create_header(*parent*)
Create application header with icon and title

_create_path_widgets(*parent*)

_create_layout_widgets(*parent*)

_create_details_widgets(*parent*)

_browse_output_file()

_browse_metadata_file()

_on_format_change(*event=None*)
Called when export format changes - updates output file extension.

_update_sort_options()

_update_ui_for_mode()

_on_scale_change(*value*)
Callback when scale slider changes.

_on_scale_entry_change(*event=None*)
Callback when scale entry changes.

_create_terminal_log(*parent*)
Create terminal-style log at the bottom.

_update_log(*message*, *tag='info'*)
Update terminal-style log with colored output.

_start_process()

_run_backend_in_thread(*params*)

_create_preview_panel(*parent*)
Create the preview panel for layout visualization.

_create_preview_controls(*parent*)
Create control panel under the preview.

_setup_preview_bindings()
Bind variable changes to preview updates with debouncing.

_schedule_preview_update()
Schedule a preview update with debouncing to avoid too frequent updates.

_load_preview_images()
Load thumbnail images for preview.

_prev_page()
Navigate to previous page.

_next_page()
Navigate to next page.

_update_preview()
Update the preview canvas with current layout settings.

_preview_grid_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview grid layout with image thumbnails.

_preview_puzzle_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview puzzle/optimized layout with image thumbnails.

_preview_masonry_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview masonry layout with image thumbnails.

_preview_manual_layout(*canvas_width*, *canvas_height*, *margin*, *spacing*, *scale_factor*, *preview_scale*)
Preview manual layout with drag-and-drop functionality.

_enable_manual_mode()
Enable drag-and-drop functionality in preview canvas.

_disable_manual_mode()
Disable drag-and-drop functionality.

_on_drag_start(*event*)
Handle drag start for manual positioning.

_on_drag_motion(*event*)
Handle drag motion for manual positioning.

_on_drag_release(*event*)
Handle drag release for manual positioning.

_browse_input_folder()

2.8.3 Module Overview

PyPotteryLayout consists of two main modules:

- **backend_logic**: Core image processing and layout generation functionality
- **gui_app**: Tkinter-based graphical user interface

The modules are designed to work together but can also be used independently. The backend_logic module can be imported and used programmatically without the GUI.

2.9 Changelog

2.9.1 Version 1.0.0 (2024)

Major Features

- **Initial Release**: Full-featured pottery layout application

- **Multiple Layout Modes**: Grid, Puzzle, Masonry, and Manual
- **Auto-scaling**: Automatic image scaling based on images per page
- **Multi-page Support**: Handle large collections across pages
- **Metadata Integration**: Excel/CSV support for captions and sorting
- **Scale Bar Generation**: Configurable archaeological scale bars
- **Export Formats**: PDF, SVG (editable), JPEG
- **Real-time Preview**: Interactive preview with drag-and-drop
- **Cross-platform**: Windows, macOS, Linux support

Recent Updates

- Added automatic image scaling feature
- Fixed multi-page export for all layout modes
- Improved manual layout with multi-page support
- Enhanced preview with auto-scale display
- Fixed puzzle layout indentation issue
- Scale bar now adapts to actual applied scale

Technical Improvements

- Optimized memory usage for large collections
- Improved font handling across platforms
- Enhanced SVG export with external image references
- Better error handling and user feedback
- Responsive preview panel updates

Known Issues

- Manual layout positions reset when changing pages
- Large collections (500+ images) may be slow
- Some font warnings on certain systems (cosmetic)

Future Plans

- Database integration for large catalogs
- Batch export with different settings
- Additional layout algorithms
- 3D pottery visualization support
- Cloud storage integration
- Mobile companion app

2.10 Contributing Guide

We welcome contributions to PyPotteryLayout! This guide explains how to contribute.

2.10.1 Getting Started

Development Setup

1. **Fork the repository** on GitHub
2. **Clone your fork**:

```
git clone https://github.com/YOUR_USERNAME/PyPotteryLayout.git
cd PyPotteryLayout
```

3. **Create virtual environment**:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

4. **Install in development mode**:

```
pip install -r requirements.txt
pip install -e .
```

5. **Create feature branch**:

```
git checkout -b feature-name
```

2.10.2 Code Style

Python Guidelines

- Follow PEP 8 style guide
- Use meaningful variable names
- Add docstrings to all functions
- Keep functions focused and small
- Write self-documenting code

Docstring Format

```
def function_name(param1, param2):
    """
    Brief description of function.

    Args:
        param1 (type): Description of param1
        param2 (type): Description of param2

    Returns:
        type: Description of return value
```

(continues on next page)

(continued from previous page)

```
    Raises:
        ExceptionType: When this exception occurs
    """
```

2.10.3 Testing

Running Tests

Execute tests with:

```
pytest tests/
```

Writing Tests

- Write tests for new features
- Maintain existing test coverage
- Use descriptive test names
- Test edge cases

2.10.4 Submitting Changes

Pull Request Process

1. **Update documentation** for changes
2. **Add tests** for new features
3. **Ensure tests pass** locally
4. **Commit with clear messages**
5. **Push to your fork**
6. **Create Pull Request** on GitHub

Commit Messages

Format:

```
type: Brief description (50 chars max)

Longer explanation if needed. Wrap at 72 characters.
Explain what and why, not how.

- Bullet points for multiple changes
- Keep related changes together
```

Types:

- feat: New feature
- fix: Bug fix

- docs: Documentation only
- style: Code style changes
- refactor: Code restructuring
- test: Test additions/changes
- chore: Build/auxiliary changes

2.10.5 Areas for Contribution

Current Needs

- **Documentation**: Tutorials, examples, translations
- **Testing**: Unit tests, integration tests
- **Features**: New layout algorithms, export formats
- **Bug Fixes**: Check GitHub issues
- **Performance**: Optimization for large datasets
- **UI/UX**: Interface improvements

Feature Ideas

- Additional layout modes
- More metadata formats
- Export format options
- Batch processing tools
- Command-line interface
- Web-based version

2.10.6 Reporting Issues

Bug Reports

Include:

1. System information (OS, Python version)
2. Steps to reproduce
3. Expected behavior
4. Actual behavior
5. Error messages/logs
6. Sample data if applicable

Feature Requests

Describe:

1. Use case/problem solved
2. Proposed solution
3. Alternative approaches
4. Mockups/examples if applicable

2.10.7 Community

Communication

- GitHub Issues: Bug reports, features
- Discussions: General questions
- Pull Requests: Code contributions

Code of Conduct

- Be respectful and inclusive
- Welcome newcomers
- Provide constructive feedback
- Focus on what's best for the community
- Show empathy towards others

2.10.8 License

By contributing, you agree that your contributions will be licensed under the same license as the project (MIT License).

2.10.9 Recognition

Contributors are recognized in:

- GitHub contributors page
- README acknowledgments
- Release notes

Thank you for contributing to PyPotteryLayout!

2.11 License

2.11.1 MIT License

and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

2.11.2 Third-Party Licenses

PyPotteryLayout uses the following open-source libraries:

Pillow (PIL Fork)

- License: HPND License
- Copyright: 1997-2011 Secret Labs AB, 1995-2011 Fredrik Lundh and Contributors
- https://github.com/python-pillow/Pillow

rectpack

- License: Apache License 2.0
- Copyright: 2013 Maximilian Ritter
- https://github.com/secnot/rectpack

openpyxl

- License: MIT License
- Copyright: 2010 openpyxl
- https://github.com/openpyxl/openpyxl

tkinter

- License: Python Software Foundation License
- Part of Python standard library
- https://docs.python.org/3/library/tkinter.html

Optional Dependencies

reportlab (optional)

- License: BSD License
- For enhanced PDF generation

cairosvg (optional)

- License: LGPLv3
- For SVG conversion

Documentation

Sphinx

- License: BSD License
- Documentation generator

sphinx-rtd-theme

- License: MIT License
- ReadTheDocs theme

2.11.3 Acknowledgments

Special thanks to:

- Archaeological community for feedback and testing
- Open source contributors
- Python Software Foundation
- All third-party library maintainers

2.11.4 Data and Images

- Example images and data are for demonstration only
- Users are responsible for rights to their own images
- No archaeological data is included in the distribution

CHAPTER

THREE

QUICK START EXAMPLE

3.1 Using the GUI

1. Launch the application:

```
python gui_app.py
```

2. Select your images folder
3. Choose a layout mode (Grid, Puzzle, Masonry, or Manual)
4. Configure settings (margins, spacing, scale)
5. Export to your desired format

3.2 Using the API

```python
import backend_logic

# Set up parameters
params = {
    'input_folder': '/path/to/images',
    'output_file': 'catalog.pdf',
    'mode': 'grid',
    'grid_rows': 4,
    'grid_cols': 3,
    'margin_px': 50,
    'spacing_px': 10,
    'add_scale_bar': True,
    'scale_bar_cm': 5,
    'pixels_per_cm': 118
}

# Run the layout process
backend_logic.run_layout_process(params)
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

B

C

D

G

L

M

N

P

R

S