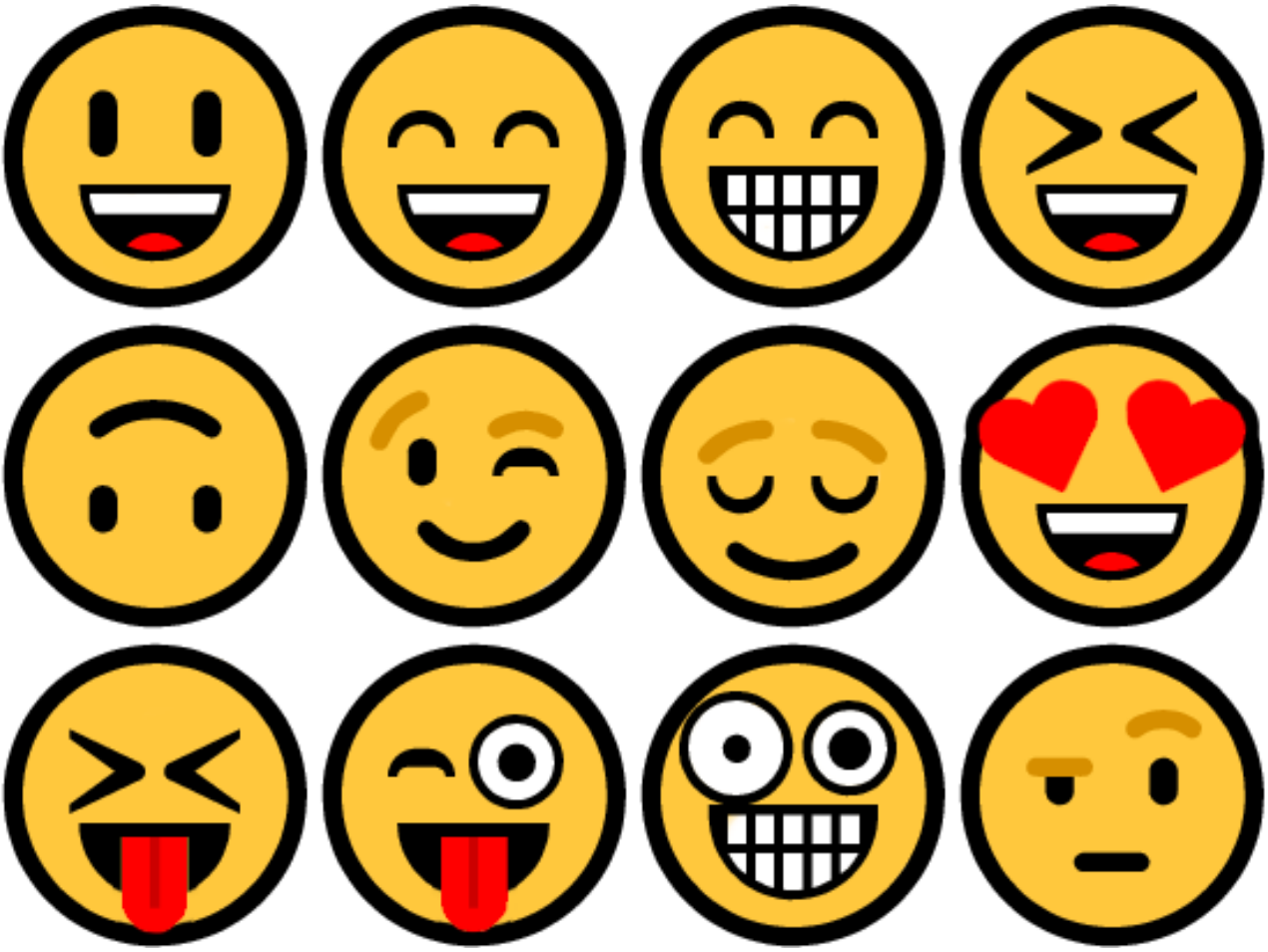

Emoji System Base

Documentation

_____

Creator: Lucas Gomes Cecchini

Online Name: AGAMENOM

Overview

The **Emoji System Base** is an emoji library for **Unity**.

With it you can add a whole list of emojis to your games as well as a tool to make editing **SpriteAsset** and **TMP_SpriteAsset** easier.

Emoji to Unicode Converter

https://lingojam.com/EmojitoUnicodeConverter

Emoji	Unicode	Emoji	Unicode	Emoji	Unicode	Emoji	Unicode	Emoji	Unicode
	1f600		1f603		1f604		1f601		1f606
	1f605		1f602		1f923		263a		1f60a
	1f607		1f642		1f643		1f609		1f60c
	1f60d		1f970		1f618		1f617		1f619
	1f61a		1f60b		1f61b		1f61d		1f61c
	1f92a		1f928		1f9d0		1f913		1f60e
	1f929		1f973		1f60f		1f612		1f61e
	1f614		1f61f		1f615		1f641		2639
	1f623		1f616		1f62b		1f629		1f97a
	1f622		1f62d		1f925		1f636		1f624
	1f620		1f621		1f92c		1f92f		1f633
	1f975		1f976		1f631		1f628		1f630
	1f625		1f613		1f917		1f914		1f92d
	1f92b		1f610		1f611		1f62c		1f644
	1f62f		1f626		1f627		1f62e		1f632
	1f634		1f924		1f62a		1f635		1f910
	1f974		1f922		1f92e		1f927		1f637
	1f912		1f915		1f911		1f920		1f608
	1f47f		1f479		1f47a		1f921		1f4a9
	1f47b		1f480		2620		1f47d		1f47e
	1f916		1F383		1f63a		1f638		1f639
	1f63b		1f63c		1f63d		1f640		1f63f
	1f63e		1f44b		1f91a		1f590		270b
	1f596		1f44c		270c		1f91e		1f91f
	1f918		1f919		1F448		1F449		1F446
	1F595		1F447		261D		1F44D		1F44E
	270A		1F44A		1F91B		1F91C		1F44F
	1F64C		1F450		1F932		1F91D		1F64F
	270D		1F485		1F933		1F4AA		1F9B5
	1F9B6		1F463		1F442		1F443		1F9E0
	1F9B7		1F9B4		1F440		1F441		1F445
	1F444		1F48B		1F476		1F467		1F9D2
	1F466		1F469		1F9D1		1F468		1F9D4
	1F475		1F9D3		1F474		1F472		1F9D5
	1F9F3		1F302		2602		1F9F5		1F9F6
	1F453		1F576		1F97D		1F97C		1F454
	1F455		1F456		1F9E3		1F9E4		1F9E5
	1F9E6		1F457		1F458		1F459		1F45A
	1F45B		1F45C		1F45D		1F392		1F45E

	1F45F		1F97E		1F97F		1F460		1F461
	1F462		1F451		1F452		1F3A9		1F393
	1F9E2		26D1		1F484		1F48D		1F4BC
	1F436		1F431		1F42D		1F439		1F430
	1F98A		1F43B		1F43C		1F43B		2744
	1F428		1F42F		1F981		1F42E		1F437
	1F43D		1F438		1F435		1F648		1F649
	1F64A		1F412		1F414		1F427		1F426
	1F424		1F423		1F425		1F986		1F985
	1F989		1F987		1F43A		1F417		1F434
	1F984		1F41D		1F41B		1F98B		1F40C
	1F41E		1F41C		1F99F		1F997		1F577
	1F578		1F982		1F422		1F40D		1F98E
	1F996		1F995		1F419		1F991		1F990
	1F99E		1F980		1F421		1F420		1F41F
	1F42C		1F433		1F40B		1F988		1F40A
	1F405		1F406		1F993		1F98D		1F418
	1F99B		1F98F		1F42A		1F42B		1F992
	1F998		1F403		1F402		1F404		1F40E
	1F416		1F40F		1F411		1F999		1F410
	1F98C		1F415		1F429		1F408		1F413
	1F983		1F99A		1F99C		1F9A2		1F54A
	1F407		1F99D		1F9A1		1F401		1F400
	1F43F		1F994		1F43E		1F409		1F432
	1F335		1F384		1F332		1F333		1F334
	1F331		1F33F		2618		1F340		1F38D
	1F38B		1F343		1F342		1F341		1F344
	1F41A		1F33E		1F490		1F337		1F339
	1F940		1F33A		1F338		1F33C		1F33B
	1F31E		1F31D		1F31B		1F31C		1F31A
	1F315		1F316		1F317		1F318		1F311
	1F312		1F313		1F314		1F319		1F30E
	1F30D		1F30F		1F4AB		2B50		1F31F
	2728		26A1		2604		1F4A5		1F525
	1F32A		1F308		2600		1F324		26C5
	1F325		2601		1F326		1F327		26C8
	1F329		1F328		1F300		2603		26C4
	1F32C		1F4A8		1F4A7		1F4A6		2614
	26F1		1F30A		1F32B		1F34F		1F34E
	1F350		1F34A		1F34B		1F34C		1F349
	1F347		1F353		1F348		1F352		1F351
	1F96D		1F34D		1F965		1F95D		1F345
	1F346		1F951		1F966		1F96C		1F952
	1F336		1F33D		1F955		1F954		1F360
	1F950		1F96F		1F35E		1F956		1F968
	1F9C0		1F95A		1F373		1F95E		1F953

	1F969		1F357		1F356		1F9C6		1F32D
	1F354		1F35F		1F355		1F96A		1F959
	1F32E		1F32F		1F957		1F958		1F96B
	1F35D		1F35C		1F372		1F35B		1F363
	1F371		1F95F		1F364		1F359		1F35A
	1F358		1F365		1F960		1F96E		1F362
	1F361		1F367		1F368		1F366		1F967
	1F9C1		1F370		1F382		1F36E		1F36D
	1F36C		1F36B		1F37F		1F369		1F36A
	1F330		1F95C		1F36F		1F95B		1F37C
	2615		1F375		1F964		1F376		1F37A
	1F37B		1F942		1F377		1F943		1F378
	1F379		1F37E		1F944		1F374		1F37D
	1F963		1F961		1F962		1F9C2		26BD
	1F3C0		1F3C8		26BE		1F94E		1F3BE
	1F3D0		1F3C9		1F94F		1F3B1		1F3D3
	1F3F8		1F3D2		1F3D1		1F94D		1F3CF
	1F945		26F3		1F3F9		1F3A3		1F94A
	1F94B		1F3BD		1F6F9		1F6F7		26F8
	1F94C		1F3BF		26F7		1F3C2		1F3C6
	1F947		1F948		1F949		1F3C5		1F396
	1F3F5		1F397		1F3AB		1F39F		1F3AA
	1F939		1F3AD		1F3A8		1F3AC		1F3A4
	1F3A7		1F3BC		1F3B9		1F941		1F3B7
	1F3BA		1F3B8		1F3BB		1F3B2		265F
	1F3AF		1F3B3		1F3AE		1F3B0		1F9E9
	1F697		1F695		1F699		1F68C		1F68E
	1F3CE		1F693		1F691		1F692		1F690
	1F69A		1F69B		1F69C		1F6F4		1F6B2
	1F6F5		1F3CD		1F6A8		1F694		1F68D
	1F698		1F696		1F6A1		1F6A0		1F69F
	1F683		1F68B		1F69E		1F69D		1F684
	1F685		1F688		1F682		1F686		1F687
	1F68A		1F689		2708		1F6EB		1F6EC
	1F6E9		1F4BA		1F6F0		1F680		1F6F8
	1F681		1F6F6		26F5		1F6A4		1F6E5
	1F6F3		26F4		1F6A2		2693		26FD
	1F6A7		1F6A6		1F6A5		1F68F		1F5FA
	1F5FF		1F5FD		1F5FC		1F3F0		1F3EF
	1F3DF		1F3A1		1F3A2		1F3A0		26F2
	1F3D6		1F3DD		1F3DC		1F30B		26F0
	1F3D4		1F5FB		1F3D5		26FA		1F3E0
	1F3E1		1F3D8		1F3DA		1F3D7		1F3ED
	1F3E2		1F3EC		1F3E3		1F3E4		1F3E5
	1F3E6		1F3E8		1F3EA		1F3EB		1F3E9
	1F492		1F3DB		26EA		1F54C		1F54D

	1F54B		26E9		1F6E4		1F6E3		1F5FE
	1F391		1F3DE		1F305		1F304		1F320
	1F387		1F386		1F307		1F306		1F3D9
	1F303		1F30C		1F309		1F301		231A
	1F4F1		1F4F2		1F4BB		2328		1F5A5
	1F5A8		1F5B1		1F5B2		1F579		1F5DC
	1F4BD		1F4BE		1F4BF		1F4FC		1F4F7
	1F4F8		1F4F9		1F3A5		1F4FD		1F39E
	1F4DE		260E		1F4DF		1F4E0		1F4FA
	1F4FB		1F399		1F39A		1F39B		1F9ED
	23F1		23F2		23F0		1F570		231B
	23F3		1F4E1		1F50B		1F50C		1F4A1
	1F526		1F56F		1F9EF		1F6E2		1F6CD
	1F4B8		1F4B5		1F4B4		1F4B6		1F4B7
	1F4B0		1F4B3		1F48E		2696		1F9F0
	1F527		1F528		2692		1F6E0		26CF
	1F529		2699		1F9F1		26D3		1F9F2
	1F52B		1F4A3		1F9E8		1F52A		1F5E1
	2694		1F6E1		1F6AC		26B0		26B1
	1F3FA		1F52E		1F4FF		1F9FF		1F488
	2697		1F52D		1F52C		1F573		1F48A
	1F489		1F9EC		1F9A0		1F9EB		1F9EA
	1F321		1F9F9		1F9FA		1F9FB		1F6BD
	1F6B0		1F6BF		1F6C1		1F6C0		1F9FC
	1F9FD		1F9F4		1F6CE		1F511		1F5DD
	1F6AA		1F6CB		1F6CF		1F6CC		1F9F8
	1F5BC		0BA3		1F6D2		1F381		1F388
	1F38F		1F380		1F38A		1F389		1F38E
	1F3EE		1F390		1F9E7		2709		1F4E9
	1F4E8		1F4E7		1F48C		1F4E5		1F4E4
	1F4E6		1F3F7		1F4EA		1F4EB		1F4EC
	1F4ED		1F4EE		1F4EF		1F4DC		1F4C3
	1F4C4		1F4D1		1F9FE		1F4CA		1F4C8
	1F4C9		1F5D2		1F5D3		1F4C6		1F4C5
	1F5D1		1F4C7		1F5C3		1F5F3		1F5C4
	1F4CB		1F4C1		1F4C2		1F5C2		1F5DE
	1F4F0		1F4D3		1F4D4		1F4D2		1F4D5
	1F4D7		1F4D8		1F4D9		1F4DA		1F4D6
	1F516		1F9F7		1F517		1F4CE		1F587
	1F4D0		1F4CF		1F9EE		1F4CC		1F4CD
	2702		1F58A		1F58B		2712		1F58C
	1F58D		1F4DD		270F		1F50D		1F50E
	1F50F		1F510		1F512		1F513		2764
	1F494		2763		1F495		1F49E		1F493
	1F497		1F496		1F498		1F49D		1F49F
	262E		271D		262A		1F549		2638

	2721		1F52F		1F54E		262F		2626
	1F6D0		26CE		2648		2649		264A
	264B		264C		264D		264E		264F
	2650		2651		2652		2653		1F194
	269B		1F251		2622		2623		1F4F4
	1F4F3		1F236		1F21A		1F238		1F23A
	1F237		2734		1F19A		1F4AE		1F250
	3299		3297		1F234		1F235		1F239
	1F232		1F170		1F171		1F18E		1F191
	1F17E		1F198		274C		2B55		1F6D1
	26D4		1F4DB		1F6AB		1F4AF		1F4A2
	2668		1F6B7		1F6AF		1F6B3		1F6B1
	1F51E		1F4F5		1F6AD		2757		2753
	203C		2049		1F505		1F506		303D
	26A0		1F6B8		1F531		269C		1F530
	267B		2705		1F22F		1F4B9		2747
	2733		274E		1F310		1F4A0		24C2
	1F4A4		1F3E7		1F6BE		267F		1F17F
	1F233		1F202		1F6C2		1F6C3		1F6C4
	1F6C5		1F6B9		1F6BA		1F6BC		26A7
	1F6BB		1F6AE		1F3A6		1F4F6		1F201
	1F523		2139		1F524		1F521		1F520
	1F196		1F197		1F199		1F192		1F195
	1F193		1F522		1F0CF		1F3B4		1F004
	23CF		25B6		23F8		23EF		23F9
	23FA		23ED		23EE		23E9		23EA
	23EB		23EC		25C0		1F53C		1F53D
	27A1		2B05		2B06		2B07		2197
	2198		2199		2196		2195		2194
	21AA		21A9		2934		2935		1F500
	1F501		1F502		1F504		1F503		1F3B5
	1F3B6		2795		2796		2797		2716
	267E		1F4B2		1F4B1		2122		00A9
	00AE		3030		27B0		27BF		1F51A
	1F519		1F51B		1F51D		1F51C		2714
	2611		1F508		1F507		1F509		1F50A
	1F514		1F515		1F4E3		1F4E2		1F4AC
	1F4AD		1F5EF		2660		2663		2665
	2666		1F550		1F551		1F552		1F553
	1F554		1F555		1F556		1F557		1F558
	1F559		1F55A		1F55B		1F55C		1F55D
	1F55E		1F55F		1F560		1F561		1F562
	1F563		1F564		1F565		1F566		1F567

Script Explanations

Tools

Emoji System Editor

Here's a detailed explanation of how the **`EmojiSystemEditor`** script works, including its variables, methods, and usage:

Class and Variables:

- **`EmojiSystemEditor`**: Inherits from **`EditorWindow`**, allowing it to create a custom editor window in Unity's editor.

- **`SerializedObject serializedObject`**: Used for handling serialized properties, which allows the editor window to work with Unity's serialization system.

- **`Vector2 scrollPosition`**: Tracks the current scroll position of the editor window, allowing for scrolling when there are many GUI elements.

- **`SpriteAsset spriteAssetEdit`**: Represents the Sprite Asset to be edited if the **`emojiType`** is set to **`Legacy`**.

- **`TMP_SpriteAsset TMP_spriteAssetEdit`**: Represents the TMP (TextMeshPro) Sprite Asset to be edited if the **`emojiType`** is set to **`TextMeshPro`**.

- **`Type emojiType`**: Enum that determines which type of asset is being edited (**`Legacy`** or **`TextMeshPro`**).

- **`float w, h, bx, by, ad, scale`**: Variables for glyph metrics. These include width (**`w`**), height (**`h`**), horizontal bearing (**`bx`** and **`by`**), advance (**`ad`**), and scale (**`scale`**).

- **`int atlasIndex`**: Index of the atlas in which the sprite is located.

- **`Texture spriteEdit`**: The texture of the sprite being edited.

- **`int size`**: The size of the squares when cropping the image using the Python script.

- **`string folderPath`**: Path to the folder where the changes will be applied.

- **`string pythonPath`**: Path to the Python executable.

- **`enum Type`**: Defines two types of assets—**`Legacy`** and **`TextMeshPro`**.

Methods:

1. **`ShowWindow()`**: A static method to open the custom editor window. It also tries to load an icon for the window from the assets.

2. **`OnEnable()`**: Called when the editor window is initialized. Sets up the **`SerializedObject`** for handling serialized properties.

3. **`OnGUI()`**: Draws the GUI elements for the editor window. It includes:
   - **Asset Type Selection**: Allows users to choose between **`Legacy`** and **`TextMeshPro`** assets.
   - **Sprite Asset Fields**: Displays fields for selecting and editing the sprite assets based on the selected type.
   - **Glyph Metrics Editor**: Provides fields to edit glyph metrics such as width, height, and other parameters.
   - **Save and Apply Buttons**: Allows users to save the edited asset or apply changes to all objects in the scene (for TextMeshPro).

- **Sprite Editor**: Provides fields for selecting the sprite and its size, along with buttons to apply changes to a folder and select the Python executable.

4. **`SaveFile()`**: Saves the sprite or TMP sprite asset with updated glyph metrics. It uses **`UpdateGlyphMetrics()`** and **`SaveAsset()`** to apply changes and save the asset.

5. **`UpdateGlyphMetrics<T>(List<T> spriteGlyphTable)`**: Updates the metrics of glyphs in the provided glyph table. It sets custom metrics for each glyph based on the editor's inputs.

6. **`SaveAsset(Object asset)`**: Marks the asset as dirty, saves it, and refreshes the AssetDatabase to apply changes.

7. **`ApplyToAllObjects()`**: Applies the selected TMP sprite asset to all **`TMP_Text`** components in the current scene.

8. **`RunPythonScript(string folder)`**: Executes the Python script for processing the sprite. It sets up a **`ProcessStartInfo`** to run the Python script with the required arguments.

9. **`GetPython()`**: Opens a file panel to select the Python executable. Updates the **`pythonPath`** variable with the selected path.

Usage:

1. **Open the Editor Window**: Go to the Unity menu bar and select **`Window`** > **`Emoji System Editor`** to open the custom editor window.

2. **Select Asset Type**: Choose between **`Legacy`** or **`TextMeshPro`** to determine which type of sprite asset you are editing.

3. **Edit Sprite Assets**: Depending on the selected asset type, you can select and edit the sprite asset and its glyph metrics.

4. **Save Changes**: Click the "Save Asset" button to apply changes to the selected sprite asset.

5. **Apply Changes to Scene**: If using TextMeshPro, use the "Apply to Scene" button to apply changes to all relevant **`TMP_Text`** components in the scene.

6. **Process Sprites**: Use the "Apply to Folder" button to select a folder and run the Python script to process the sprite image.

7. **Select Python Executable**: Click the "python.exe" button to select the Python executable used for running the script.

This script is a comprehensive tool for managing and editing sprite assets and their glyph metrics, integrating both Unity editor functionalities and Python scripting for advanced image processing tasks.

crop_image.py

Here's a detailed explanation of the Python script that cuts an image into smaller squares and saves them:

Overview:
This script takes an image file, divides it into smaller square sections of a specified size, and saves these sections as individual PNG files in a designated folder.

Imports:
- `**sys**`: Provides access to command-line arguments and system-specific parameters.
- `**os**`: Provides a way to interact with the operating system, such as creating directories.
- `**Image**` from `**PIL**`: The Python Imaging Library (Pillow) is used for opening, manipulating, and saving image files.

Function: `cut_image_into_squares`

Parameters:
- `**image_png**`: Path to the input image file.
- `**square_size**`: The size of each square section (in pixels).
- `**destination_folder**`: The folder where the cropped squares will be saved.

Steps:
1. **Open the Image**:

```
img = Image.open(image_png)
width, height = img.size
```

   - Opens the image file specified by `**image_png**` and retrieves its dimensions.

2. **Calculate Number of Squares**:

```
number_squares_horizontal = width // square_size
number_squares_vertical = height // square_size
```

   - Determines how many squares fit horizontally and vertically in the image.

3. **Create Destination Folder**:

```
if not os.path.exists(destination_folder):
    os.makedirs(destination_folder)
```

   - Checks if the `**destination_folder**` exists. If not, it creates the folder.

4. **Loop Through and Crop Squares**:

```
for j in range(number_squares_vertical):
    for i in range(number_squares_horizontal):
        left = i * square_size
        top = j * square_size
        right = left + square_size
        bottom = top + square_size

        square = img.crop((left, top, right, bottom))
        file_name = f'square_{j}_{i}.png'
        destination_path = os.path.join(destination_folder, file_name)
        square.save(destination_path)
```

   - Iterates over the calculated number of squares.

- For each square, calculates its coordinates and crops the image accordingly.
- Saves each cropped square to the **`destination_folder`** with a filename indicating its position.

5. **Print Success Message**:

```
print("Images successfully applied to folder.")
```

- Outputs a message indicating that the operation is complete.

Main Execution Block:

```
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("Usage: script.py <image_path> <square_size> <destination_folder>")
        sys.exit(1)

    image_path = sys.argv[1]
    square_size = int(sys.argv[2])
    destination_folder = sys.argv[3]

    cut_image_into_squares(image_path, square_size, destination_folder)
```

- **Check Command-Line Arguments**:
  - Ensures that exactly three arguments are provided: the image path, the square size, and the destination folder.
  - Prints a usage message and exits if the arguments are incorrect.

- **Retrieve Arguments**:
  - Assigns the command-line arguments to **`image_path`**, **`square_size`**, and **`destination_folder`**.

- **Call the Function**:
  - Calls **`cut_image_into_squares`** with the provided arguments to perform the image processing.

Usage:

To use this script, run it from the command line with the following syntax:

```
python script.py <image_path> <square_size> <destination_folder>
```

- **`<image_path>`**: Path to the image file to be processed.
- **`<square_size>`**: Size of each square section (in pixels).
- **`<destination_folder>`**: Folder where the resulting squares will be saved.

This script is useful for splitting large images into smaller, manageable pieces, which can be helpful in various applications like texture atlases or image processing tasks.