Deployment Scripts Documentation

This document explains the two shell scripts that handle website generation and deployment for USDZ file galleries.

---

generate_index_with_USDZ.sh

Overview

Creates a responsive HTML gallery page that displays all USDZ files with thumbnails as downloadable links. Designed for GitHub Pages deployment with modern styling.

Purpose

- **Gallery Generation**: Creates visual catalog of 3D models
- **Thumbnail Integration**: Shows PNG previews alongside download links
- **GitHub Pages Ready**: Includes `.nojekyll` file for proper deployment
- **Responsive Design**: Works on desktop and mobile devices

Script Breakdown

1. Header and Setup

```bash
#!/bin/bash
echo "--- Generating index.html and .nojekyll file ---"
```

- Standard bash script with status messaging
- Indicates start of generation process

2. GitHub Pages Configuration

```bash
touch .nojekyll
echo ".nojekyll file ensured for GitHub Pages."
```

Why `.nojekyll` is needed:

- GitHub Pages uses Jekyll by default
- Jekyll ignores files starting with underscore
- `.nojekyll` tells GitHub to serve files directly
- Essential for proper asset loading (CSS, images, etc.)

3. HTML Template Structure

The script builds a complete HTML document with modern styling:

DOCTYPE and Meta Tags:

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- HTML5 standard structure
- UTF-8 encoding for international filenames
- Responsive viewport for mobile compatibility

Styling Framework:

```html
<script src="https://cdn.tailwindcss.com"></script>
```

- Uses Tailwind CSS for rapid, modern styling
- CDN delivery for reliability and speed
- No local dependencies required

Custom CSS:

```css
body {
  font-family: "Inter", sans-serif;
  background-color: #f0f4f8;
  color: #334155;
}
```

- **Inter font**: Clean, modern typeface
- **Light blue-gray background**: Professional appearance
- **Dark gray text**: Good contrast and readability

4. Page Layout Components

Header Section:

```html
<header class="bg-blue-600 text-white shadow-lg py-4">
```

- **Blue gradient header** with navigation
- **Shadow effect** for depth
- **Responsive navigation** with hover effects

Main Content Area:

```html
<main class="flex-grow py-8">
  <div class="container bg-white p-8 rounded-xl shadow-lg">
```

- **Flex-grow**: Ensures footer stays at bottom
- **White content area** with rounded corners
- **Subtle shadow** for card-like appearance

Footer:

```
html

<footer class="bg-gray-800 text-white py-6 mt-auto">
```

- **Dark footer** with copyright information
- **Auto-margin top** pushes to bottom of viewport

5. File Processing Logic

IFS Configuration:

```
bash

IFS=$'\n'
for filename in $(ls -1 *.usdz 2>/dev/null | sort); do
```

Why IFS matters:

- Default IFS splits on spaces, breaking filenames with spaces
- Setting `IFS=$'\n'` makes loop split only on newlines
- Essential for files like "w67b 106 bed and me.usdz"
- `2>/dev/null` suppresses error if no USDZ files exist

Thumbnail Detection:

```
bash

thumbnail_file="${filename%.usdz}.png"
if [ -f "$thumbnail_file" ]; then
    thumbnail_html="<img src=\"./${thumbnail_file// /%20}\"..."
else
    thumbnail_html="<div class=\"thumbnail-placeholder\">📦</div>"
fi
```

Process:

1. **Generate thumbnail filename** by replacing `.usdz` with `.png`
2. **Check if thumbnail exists** using file test
3. **Create appropriate HTML**:
    - If PNG exists: `<img>` tag with proper encoding
    - If no PNG: Placeholder div with package emoji
4. **URL encoding**: Spaces become `%20` for web compatibility

URL Encoding:

```
bash

ENCODED_FILENAME="${filename// /%20}"
```

- **Bash parameter expansion**: `${filename// /%20}` replaces all spaces with `%20`
- **Web standard**: Ensures links work with spaces in filenames
- **Download attribute**: Preserves original filename when downloading

6. HTML Generation

Dynamic List Building:

```bash
INDEX_HTML_CONTENT+="
  <li class=\"file-item\">
    ${thumbnail_html}
    <a href=\"./${ENCODED_FILENAME}\" download=\"${filename}\">
      ${filename}
    </a>
  </li>"
```

Structure:

- **Flex layout**: Thumbnail, filename, download arrow
- **Download attribute**: Forces download instead of browser preview
- **Hover effects**: Scale and color transitions
- **Accessibility**: Proper semantic markup

File Output:

```bash
echo "$INDEX_HTML_CONTENT" > index.html
```

- Writes complete HTML to `index.html`
- Overwrites existing file
- Creates web-ready gallery page

~~deploy.sh~~

~~Overview~~

~~Automates Git workflow for deploying changes to remote repositories. Handles the standard add-commit-push cycle with error checking and user feedback.~~

~~Purpose~~

- ~~**Automated Deployment**: Reduces manual Git commands~~
- ~~**Error Handling**: Provides clear feedback on failures~~
- ~~**GitHub Pages Integration**: Optimized for static site deployment~~
- ~~**Zero-downtime**: Checks for changes before committing~~

~~Script Breakdown~~

~~1. Configuration Variables~~