Here's comprehensive documentation for your No Man's Sky Glyph Detector web application:

No Man's Sky Glyph Detector Documentation

Overview

A web application that detects and decodes portal glyph sequences from No Man's Sky screenshots using computer vision (OpenCV.js). Users can paste screenshots or navigate through a gallery of images to extract glyph sequences.

Key Features

- Glyph detection from screenshots using OpenCV.js
- Interactive glyph string editing
- Image gallery navigation
- Dynamic background color extraction
- Integration with NMS Portals Decoder

Technical Stack

- Vanilla JavaScript (ES Modules)
- OpenCV.js (v4.x) for computer vision
- HTML5 Canvas for image processing
- JSON for image metadata storage

File Structure

```
V5
v5
-v5.
|←
```

Detailed Module Documentation

glyphUtils.js

`waitForOpenCV()`

```
export async function waitForOpenCV()
```

- **Purpose**: Ensures OpenCV.js is fully loaded
- **Returns**: Promise resolving to OpenCV instance
- **Usage**: Called during initialization

`isGlyphs(tempImage, glyphSprite, cvInstance)`

```
export async function isGlyphs(tempImage, glyphSprite, cvInstance)
```

- **Purpose**: Detects glyph sequence from an image

- **Parameters**:
  - `tempImage`: Input image element
  - `glyphSprite`: Reference glyph sprite sheet
  - `cvInstance`: OpenCV instance
- **Process**:
  1. Extracts logo area (bottom-left quarter)
  2. Converts to grayscale
  3. Uses template matching to identify glyphs
  4. Returns 12-character glyph string
- **Returns**: Promise resolving to glyph string (e.g., "0123456789ab")

`renderGlyphString(glyphs, setSelectedGlyphIndex)`

```
export function renderGlyphString(glyphs, setSelectedGlyphIndex)
```

- **Purpose**: Renders interactive glyph string
- **Features**:
  - Clickable glyphs for selection
  - Visual selection indicator
  - Automatic decoder link update

`setupGlyphSpriteClick()`

```
export function setupGlyphSpriteClick(getSelectedGlyphIndex, setSelectedGlyphIndex)
```

- **Purpose**: Enables glyph editing by clicking sprite sheet
- **Behavior**:
  - Updates selected glyph in string
  - Maintains selection state

imageNavigation.js

Image Management Functions

```
export function setImages(data)  // Sets image gallery
export function nextImage()      // Navigates to next image
export function prevImage()      // Navigates to previous image
export function showImage()      // Displays image at index
```

`setBodyBgColor(canvas)`

```
export function setBodyBgColor(canvas)
```

- **Purpose**: Extracts dominant color from logo area
- **Process**:
  1. Samples color from specific coordinates
  2. Lightens color by 50%
  3. Applies as background
  4. Marks sample point on canvas

main.js

- **Entry Point**: `onOpenCVLoaded()`
- **Features**:

  - OpenCV initialization
  - Clipboard paste handling
  - Image gallery loading
  - Keyboard navigation (← → arrows)

Setup Instructions

1. **Requirements**:

   - Modern browser (Chrome/Firefox recommended)
   - Local HTTP server (due to CORS restrictions)

2. **Installation**:

   ```
   npm install -g http-server
   http-server
   ```

3. **Usage**:

   - Paste screenshots (Ctrl+V)
   - Navigate images with arrow keys
   - Click glyphs to edit
   - Click decoder link for portal address

Known Issues

- Cross-origin restrictions require local server
- OpenCV.js loading may fail on slow connections
- Glyph detection accuracy varies with image quality

Future Improvements

- Drag-and-drop image upload
- Glyph detection confidence indicators
- Batch processing for multiple images
- Mobile responsiveness

Example Usage Flow

1. User pastes screenshot
2. App detects 12-glyph sequence
3. User can:
   - Click glyphs to select
   - Click sprite to modify
   - Navigate gallery with arrows
   - Visit decoder with current sequence

This documentation covers the core functionality and architecture of your glyph detection system. Would you like me to elaborate on any specific aspect or add additional sections?