QR Camera Preview Tool

A lightweight camera preview application for the Raspberry Pi that shows real-time QR code detection with visual overlays. Perfect for setting up your QR token tracking system and monitoring during gameplay.

Features

📹 Live Camera Preview

- Adjustable frame rate (0.5-5 FPS)
- Multiple resolution options
- Fullscreen mode support
- Efficient performance for Raspberry Pi

🎯 QR Code Detection & Overlays

- **Corner Markers**: Green bounding box around the calibrated surface
- **Player Tokens**: Red circles and shapes over detected players
- **Real-time Classification**: Automatically distinguishes between corner and player QR codes
- **Confidence Indicators**: Visual feedback on detection quality

📊 Information Display

- Live statistics (FPS, resolution, detection counts)
- Surface calibration status
- Current timestamp
- Toggleable help overlay

💾 Frame Capture

- Save frames for debugging or documentation
- Timestamped filenames
- Organized in `saved_frames/` directory

Visual Overlay Guide

Corner QR Codes (`CORNER_TL`, `CORNER_TR`, `CORNER_BL`, `CORNER_BR`)

- **Yellow outline** around QR code

- **Yellow center dot** at QR position
- **Green bounding box** connecting all four corners (when all detected)
- **"SURFACE" label** in center of calibrated area

Player QR Codes (any other QR data)

- **Red shapes** based on detection confidence:
  - **Filled circle**: High confidence (>80%)
  - **Circle outline**: Medium confidence (50-80%)
  - **Dashed circle**: Low confidence (<50%)
- **White text label** with QR code ID
- **Confidence value** displayed nearby
- **Red outline** around QR code boundaries

Installation

Prerequisites

```bash
# Ensure camera is enabled
sudo raspi-config
# Navigate to Interface Options > Camera > Enable

# Install required packages
sudo apt update
sudo apt install python3-opencv python3-numpy
pip3 install picamera2 pyzbar opencv-python numpy
```

Setup

```bash
# Make launcher executable
chmod +x launch_preview.sh

# Test camera
python3 -c "from picamera2 import Picamera2; print('Camera OK')"
```

Usage

Quick Start with Launcher

```bash
# Setup mode (recommended for first time)
./launch_preview.sh setup

# Monitor mode (for gameplay)
./launch_preview.sh monitor

# Debug mode (for troubleshooting)
./launch_preview.sh debug

# Fullscreen mode (for displays)
./launch_preview.sh fullscreen

# Custom configuration
./launch_preview.sh custom
```

Direct Command Line

```bash
# Basic preview
python3 camera_preview.py

# High resolution, 2 FPS
python3 camera_preview.py --fps 2.0 --resolution 1920x1080

# Start in fullscreen without help
python3 camera_preview.py --fullscreen --no-help

# Disable corner detection
python3 camera_preview.py --no-corners
```

Command Line Options

```
--fps FLOAT         Frame rate (default: 1.0)
--resolution WxH    Camera resolution (default: 1280x720)
--no-corners        Start with corner detection disabled
--no-players        Start with player detection disabled
--no-help           Start with help overlay hidden
--fullscreen        Start in fullscreen mode
```

Keyboard Controls

Key	Action
q or ESC	Quit application
h	Toggle help overlay
c	Toggle corner detection
p	Toggle player detection
s	Save current frame
f	Toggle fullscreen mode

Use Cases

🔧 Setup & Calibration

Use **setup mode** when:

- Positioning your camera for the first time
- Testing QR code readability
- Placing corner markers
- Verifying surface detection

Recommended settings: 1 FPS, help overlay enabled, all detection on

👁️ Game Monitoring

Use **monitor mode** when:

- Monitoring token positions during gameplay
- Verifying tracking accuracy
- Troubleshooting detection issues

Recommended settings: 2 FPS, minimal UI, player detection focus

🐛 Debugging & Documentation

Use **debug mode** when:

- Investigating detection problems
- Documenting your setup
- Creating support materials
- Testing different lighting conditions

Recommended settings: 1 FPS, all overlays, frame saving enabled

🖥️ Display & Presentation

Use **fullscreen mode** when:

- Showing the setup to others
- Wall-mounted displays
- Streaming or recording
- Demonstrations

Recommended settings: Higher resolution, fullscreen, clean UI

Tips for Best Results

🎯 QR Code Preparation

- **Size**: Minimum 2cm x 2cm for reliable detection
- **Contrast**: Black QR codes on white background
- **Quality**: Clean printing, avoid wrinkles or damage
- **Lighting**: Even, bright lighting across entire surface

📷 Camera Positioning

- **Height**: Position camera high enough to see entire play area
- **Angle**: Camera should be roughly perpendicular to surface
- **Focus**: Ensure QR codes are in camera's focus range
- **Stability**: Mount camera securely to avoid movement

🎨 Visual Setup

- **Background**: Use solid color, non-reflective surface

- **Corners**: Place corner markers at exact table edges
- **Spacing**: Ensure corner markers are clearly separated
- **Clean area**: Keep surface free of other high-contrast patterns

Troubleshooting

Camera Issues

```bash
# Test camera module
python3 -c "from picamera2 import Picamera2; cam = Picamera2(); cam.start(); print('Camera working')"

# Check camera permissions
sudo usermod -a -G video $USER
# Log out and back in

# Enable camera interface
sudo raspi-config
# Interface Options > Camera > Enable
```

Detection Issues

QR Codes Not Detected:

- Check lighting - too dark or too bright affects detection
- Verify QR codes are large enough and high contrast
- Ensure QR codes are not damaged or wrinkled
- Try different camera focus or distance

Poor Detection Confidence:

- Increase QR code size
- Improve lighting uniformity
- Clean camera lens
- Check for camera shake or vibration

Corner Calibration Problems:

- Ensure all four corner QR codes are visible
- Use exact text: `CORNER_TL`, `CORNER_TR`, `CORNER_BL`, `CORNER_BR`

- Place corners at the actual edges of your play area
- Check that corners form a reasonable quadrilateral

Performance Issues

Low Frame Rate:

- Use lower resolution (`--resolution 640x480`)
- Reduce target FPS (`--fps 0.5`)
- Close other applications
- Check CPU temperature: `vcgencmd measure_temp`

High CPU Usage:

- Lower resolution and FPS
- Disable unused detection (`--no-corners` or `--no-players`)
- Ensure adequate cooling for Raspberry Pi

Integration with Main Tracker

This preview tool is designed to complement the main Foundry QR tracker:

1. **Setup Phase**: Use preview tool to position camera and test detection
2. **Calibration**: Verify corner markers are working correctly
3. **Testing**: Check player token detection before starting game
4. **Monitoring**: Run preview alongside main tracker for visual feedback

The preview tool uses the same QR detection code as the main tracker, so what you see in the preview should match what the Foundry integration detects.

Saved Frames

Frames are automatically saved to `saved_frames/` directory when you press `s`:

```
saved_frames/
├── qr_preview_20241127_143022_001.jpg
├── qr_preview_20241127_143055_002.jpg
└── ...
```

Filename format: `qr_preview_YYYYMMDD_HHMMSS_NNN.jpg`

Use saved frames for:

- Documenting your setup
- Sharing issues for support
- Creating training materials
- Analyzing detection problems

Advanced Configuration

Custom Shapes for Different Token Types

You can modify the `draw_player_overlays()` function to use different shapes based on QR content:

```python
# Example: Different shapes for different player types
if qr.id.startswith('PLAYER_'):
    # Draw circle for players
    cv2.circle(overlay_frame, qr.center, 20, self.colors['player_circle'], -1)
elif qr.id.startswith('ENEMY_'):
    # Draw square for enemies
    cv2.rectangle(overlay_frame,
                  (qr.center[0]-15, qr.center[1]-15),
                  (qr.center[0]+15, qr.center[1]+15),
                  (255, 0, 255), -1)
```

Color Customization

Modify colors in the `__init__` method:

```python
self.colors = {
    'corner_box': (0, 255, 0),     # Green
    'player_circle': (255, 0, 0),   # Red
    'player_text': (255, 255, 255), # White
    # Add your custom colors here
}
```

Support

If you encounter issues:

1. Check this README for common solutions
2. Test with the debug mode to gather information
3. Save frames showing the problem
4. Check system requirements and dependencies
5. Verify QR code quality and camera setup

The preview tool is designed to make QR tracking setup and debugging as easy as possible. Happy gaming! 🎲