Contents

- Get Started
  - Compatibility
  - Windows
  - What is a “web server” exactly?
  - Common pitfalls & quick fixes
  - Windows + Docker Desktop
- Configuring TVS
  - Conceptual overview
  - What’s a YML file?
  - Reading the example config
  - Channels and content tutorial
- Channels
  - Channel Properties
- Compatibility
  - Browser Configuration
- Keyboard Shortcuts
  - Overriding Keyboard Shortcuts
- Content Engine Primer
  - Examples
  - Properties
- Images
  - Properties

Get Started

Television Simulator '99 *(TVS)* is a configurable web-based media frontend that simulates an analog TV, tuner, receiver and cable headend. It is a web-based application and needs to be hosted on a web server to be used. There are two currently supported ways of running TVS:

- Via your favorite web server
- Through the Docker image

Compatibility 🔗

TVS is designed to run on all platforms that support the latest version of the Chromium browser. Browsers like Google Chrome, Brave, Edge, etc. will work. Firefox and Safari may run but support is not guaranteed.

Using a Web Server 🔗

You can run Television Simulator on any web server that serves static files. This includes Apache, Nginx, IIS, and more. TVS is distributed as a collection of web files (HTML, CSS, JS, etc.) and can be hosted anywhere you'd like. Some features like Twitch integration require TVS to run in a "secure context", meaning either via HTTPS or locally on your computer using 🌎 `localhost`.

Windows 🔗

Download and unzip the app 🔗

1. Get the latest **release** from GitHub.
2. Click the latest **`tvs-{version}.zip`** file to download it.
3. In **File Explorer**, right-click the zip → **Extract All…**
    - Pick a simple folder, e.g. 📁 `C:\TVS99`

- When you open 📁 `C:\TVS99`, you should see files like 📄 `index.html`, `placeholders/`, etc. (If there's an extra top-level folder, go into it; the folder that contains 📄 `index.html` is what we'll serve.)

Option A — Use the built-in Windows web server (IIS)

Good if you want a "set it and forget it" setup that starts with Windows.

Turn on IIS (one-time)

1. Press **Start**, type **Windows Features**, open **Turn Windows features on or off**.
2. Check **Internet Information Services** (leave default sub-items as they are) → **OK**.
3. Wait for it to install.

Point IIS at your folder

1. Press **Start**, type **IIS** and open **Internet Information Services (IIS) Manager**.
2. In the left tree, right-click **Sites** → **Add Website…**
3. Fill in:
   - **Site name:** TVS99
   - **Physical path:** 📁 `C:\TVS99` (the folder with 📄 `index.html`)
   - **Port:** `8080` (to avoid conflicts—any free port is fine)
4. Click **OK**. If it didn't auto-start, click **Start** on the right.
5. Open your browser and go to **http://localhost:8080/** You should see Television Simulator '99.

> To stop later: in IIS Manager, select **TVS99** → **Stop**. To remove it: right-click the site → **Remove**.

Option B — Use a tiny “temporary” server (Python)

Great for quick testing or one-off demos. You run a command, use the app, then close the window.

Install Python (one-time)

1. Open the **Microsoft Store**.
2. Search **Python 3** (from the Python Software Foundation) and click **Get** / **Install**.

Start a simple server

1. Open **File Explorer** and go to your folder (e.g. 📁 `C:\TVS99`).
2. Click the **address bar**, type `powershell` and press **Enter** (opens PowerShell **in that folder**).
3. Run:

   ```
   py -m http.server 8000
   ```

4. Open your browser and go to **http://localhost:8000/**

> Keep that PowerShell window open while you’re using the app. Press **Ctrl + C** in that window to stop the server.

What is a “web server” exactly?

It’s just a program that shows the files in a folder to your browser as a website.

- In the first example, IIS is that program. You told it: “serve 📁 `C:\TVS99` on port 8080”.
- In the second example, the Python command temporarily serves the same folder until you close it.

The **“root”** of your web server is simply the folder you serve (the one with 📄 `index.html`).

Common pitfalls & quick fixes

- **I only see a file list, not the app.** Make sure you're serving the folder that directly contains 📄 `index.html` (not a parent folder).
- **Port already in use.** Try a different port (e.g. 8081) when adding the site in IIS or in your Python command (`py -m http.server 8081`).
- **Firewall prompt.** If Windows asks, click **Allow** so your browser can connect to the local server.
- **Blank page / 404 on refresh.** Try the site root (e.g. 🌎 `http://localhost:8080/`). If you bookmarked a deep link, go to the homepage first.
- **Updating the app later.** Stop your server, replace the files in 📁 `C:\TVS99` with the new zip contents, then start again.

Using Docker

Windows + Docker Desktop

Install Docker Desktop and set it to auto-start when you sign in

1. Install **Docker Desktop for Windows** (from docker.com).
2. Open Docker Desktop → **Settings** (gear) → **General** → turn on **Start Docker Desktop when you sign in to your computer**. ([Docker Documentation][1])

> Note: Docker Desktop starts **after** you log in; that's normal on Windows. Containers marked `restart: unless-stopped` will auto-start once Docker is running.

Make a folder structure

Create a simple folder for your setup, for example:

```
C:\TVS99\
├─ docker-compose.yml
├─ config\
│   └─ config.tvs.yml        ← your TVS config file
└─ content\                  ← any media you want TVS to serve
    ├─ images\
    └─ videos\
```

- You can grab or author your 📄 `config.tvs.yml` as usual.
- Inside your config, you can reference files in the `content` folder using paths like **`/content/your-media.jpg`**.

Create `docker-compose.yml`

In `C:\TVS99\docker-compose.yml`, paste:

```yaml
services:
  tvs:
    image: zshall/television-simulator:latest
    container_name: tvs99
    restart: unless-stopped
    ports:
      - "8080:3000"               # visit http://localhost:8080
    volumes:
      - ./config/config.tvs.yml:/home/static/config.tvs.yml:ro
      - ./content:/home/static/content:ro
```

Why these paths? The image serves the app from a static web server on **port 3000**, and it expects your config file to be available from the site root. Mapping to `/home/static/...` makes the file available at `http://localhost:8080/config.tvs.yml`, and mapping the folder to `/home/static/content` makes your content reachable at `/content/...` as referenced in your config.

> If you already run something on port 8080, change `"8080:3000"` to another free port like `"8081:3000"`.

Start it up

1. Open **PowerShell**.
2. Run:

```
cd C:\TVS99
docker compose pull
docker compose up -d
```

3. Open your browser to **http://localhost:8080/**. You should see Television Simulator '99.

Everyday use

- **Start with Windows sign-in:** Docker Desktop will launch after you log in (because you enabled that toggle). Your `tvs99` container will auto-start thanks to `restart: unless-stopped`.
- **Stop the app:**

```
docker compose down
```

- **View logs:**

```
docker compose logs -f
```

- **Update to the latest image later:**

```
docker compose pull
docker compose up -d
```

Common pitfalls & quick fixes 🔗

- **Blank page / 404:** Make sure your 📄 path is exactly as mounted (`/home/static/config.tvs.yml`) and that your browser can fetch `http://localhost:8080/config.tvs.yml`.
- **Media not found:** In your config, reference media using paths like `/content/...` (e.g., `/content/images/logo.png`). Confirm the file really exists under `C:\TVS99\content\...`. ([Docker Hub][2])
- **Port already in use:** Change `8080:3000` to another free port.
- **Compose file location:** The `./...` paths are **relative to the folder that contains `docker-compose.yml`**—keep your files under `C:\TVS99\` so the mounts work.

Configuring TVS

Now that you've got TVS running, it's time to make it your own! Everything in TVS can be set up in a YAML-based config file named 📄 `config.tvs.yml`, located in the root of your installation directory. This file tells TVS what channels to show, where to find your media, and how everything should look.

Conceptual overview 🔗

Television Simulator simulates the following components of a traditional television experience:

- **TV Screen**: we simulate a television screen with configurable aspect ratios and screen effects, such as CRT scanlines, picture noise, picture blur and shadow masks. Standard definition, high definition and custom resolutions can be set. The screen will automatically resize to fill available space in your browser window and works in full screen as well. All of these effects can be disabled or customized, and depending on the hardware you're running TVS on you may want to disable some effects for better performance.

- **Tuner and Receiver**: the look and feel of the on-screen displays can be skinned to look like a typical built-in TV tuner and a few different models of set top boxes. Colors can be customized and program information can be shown on-screen (if available). Volume controls and muting indicators get their own set of on-screen controls as well. You can mix and match tuner and receiver themes.

- **Headend**: In the cable TV business, a headend is a facility where channels are received from different sources and combined into the channel lineup that's transmitted to you. Television Simulator lets you define your own channels by number and to decide what content to put on each channel. Most of your configuration file will likely consist of channels and content.

What's a YML file? 🔗

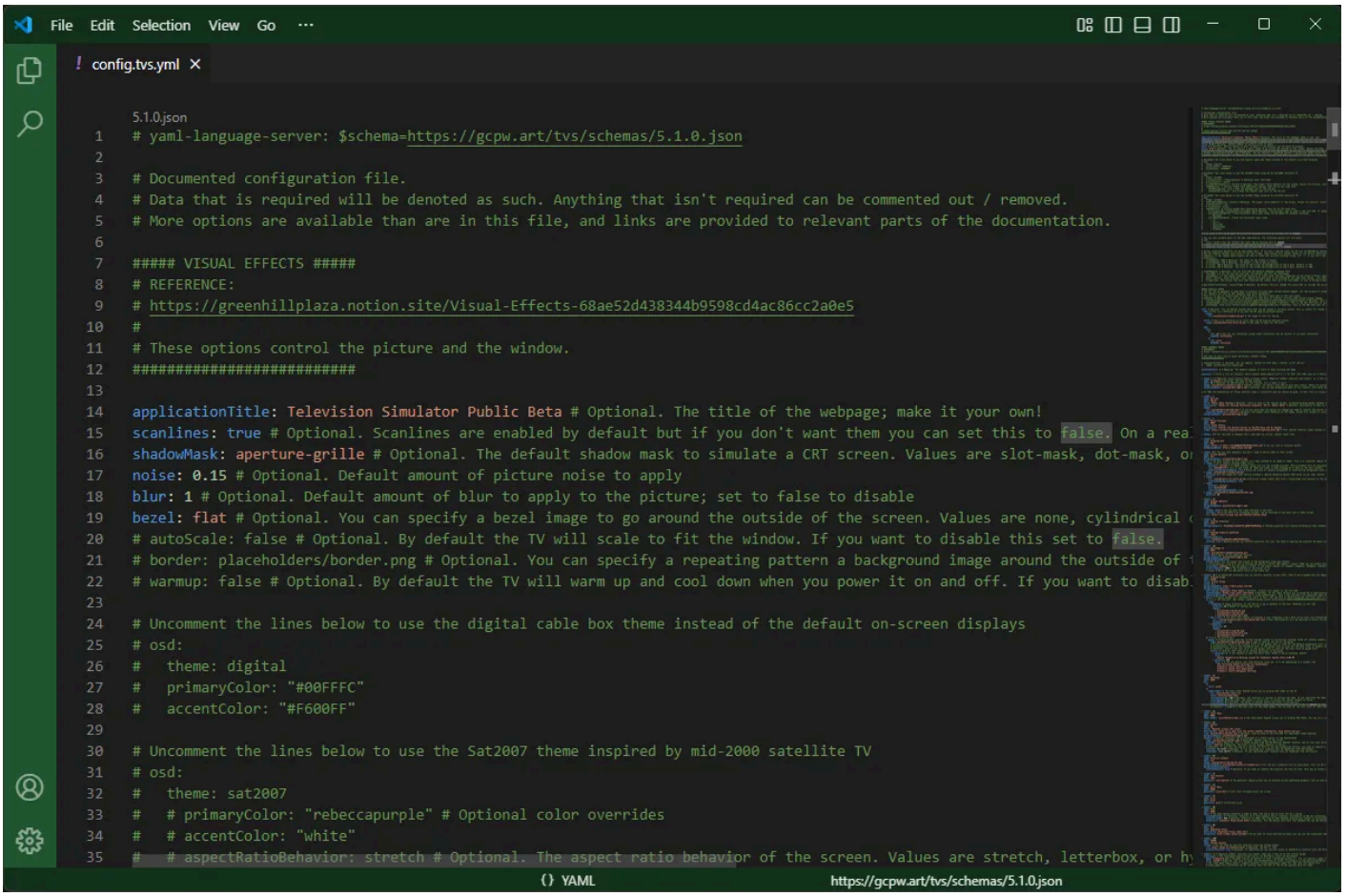
The TVS configuration file is written in the YAML language. It's not a programming language but rather a way to represent data in a human-readable format. YAML is often used for configuration files because it's easy to read and write. To make it even

easier to read and write, you can use a program like Visual Studio Code (known as VS Code) to edit your config file. VS Code can highlight your YAML syntax and help you avoid mistakes.

Configuring Visual Studio Code

To configure VS Code for editing YAML files, you can install the YAML extension from the Visual Studio Code Marketplace. This extension provides features like syntax highlighting, autocompletion, and validation for YAML files, making it easier to work with your TVS configuration.

> **Caution**
>
> The YAML extension has a known bug in version 1.18 that can lead to incorrectly saying that your config file is invalid. This should be fixed in version 1.19, but until then install version 1.17 of the extension.



Automatic validation

In Visual Studio Code you can validate your config file with the YAML extension installed by making sure that it begins with the following text which points VS Code to the TVS *schema* file:

```
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json
```

As TVS evolves, the schema version may change. All previous schemas will remain available for backward compatibility.

Reading the example config

Television Simulator ships with an example configuration file (📄 `config.tvs.yml`) that showcases different features. More example configurations are available in the 📁 `config-examples` directory, and can be previewed by navigating to 🌎 `/examples` in TVS itself.

> **🚀 Tip**
> It's a good idea to back up the example config file, as ultimately you'll be editing or replacing it with your custom configuration.

Channels and content tutorial

Minimum example

At a minimum, your configuration file must have at least one channel defined. Channels must have unique channel numbers. To learn more, backup your 📄 `config.tvs.yml` file and replace it with the following contents:

```yaml
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json

channels:

- number: 1
```

This configuration has a single channel, channel 1. It has no content so when you start TVS up with this config you'll see nothing but static. The `- number: 1` line begins

defining the channel. Each channel will start with a line like this after the `channels:` block which starts the list.

We can make this a bit more interesting by adding some content to the channel. **Channels can only have one top-level content item**, but this isn't as limiting as it sounds (we'll get into that later). Add some content like this:

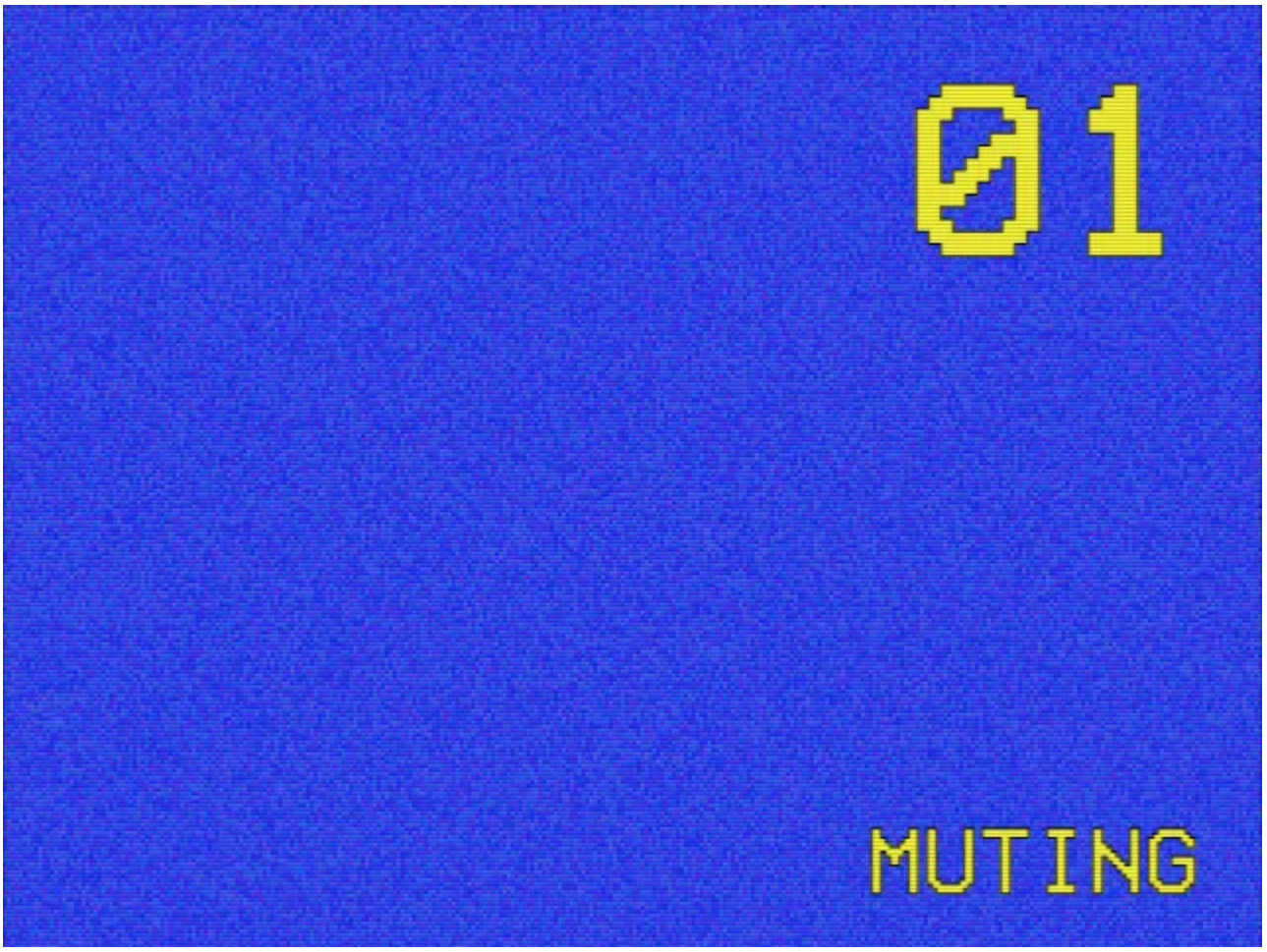
```yaml
# yaml-language-server: $schema=https://gcpw.art/tvs/schemas/5.1.0.json

channels:

- number: 1
  color: blue
```

> ⚠ **Caution**
>
> Note the indentation! YAML requires consistent indentation to work correctly. Here we're using two spaces for indentation.

All lines that are indented under the `- number: 1` line are part of the channel's definition. Here we've added the `color` content engine, which simply displays a solid background color.

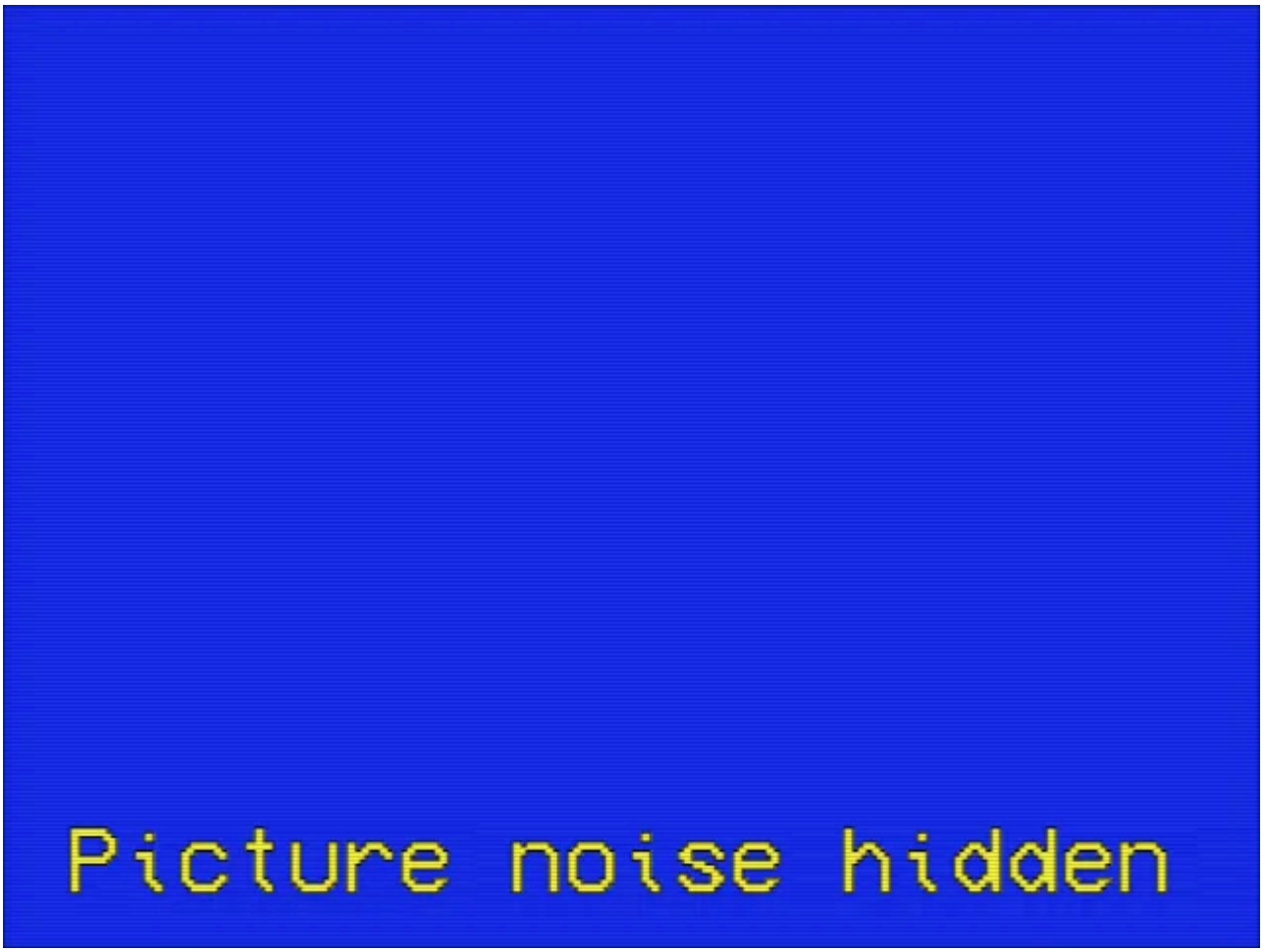


Defaults

As you can see in the screenshot, we're using a standard-definition picture with the default tuner and receiver themes which provide an old-school look. A bit of picture noise, scanlines and blur are also enabled by default. Say you'd prefer a more modern appearance, you can change the defaults in your configuration file (to change the defaults for everyone) or by using keyboard shortcuts (to change how it looks for your browser only). You may want to do this to find out how you'd prefer things to look, then set the defaults in your configuration file.

Picture controls

Let's try changing the way the simulated TV screen looks. With TVS running, press the keyboard shortcut `Alt2` `Alt2` to turn off picture noise. You'll see a message in the bottom of the screen showing that the picture changed:



Similarly, if you use `Alt1` `Alt1` you'll see the scanlines effect turn on and off. Alt commands from 1 through 7 are currently implemented and will change picture settings, so try them out to find out how they work!

In the Keyboard Shortcuts section you can find a full list of keyboard shortcuts and what they do.

Resetting the picture

You can screw up the picture settings quite a bit, but don't worry! You can always reset to the defaults (as defined in the config file) by pressing `AltR` `AltR` . This will reset all picture settings and reload the page.

Graphical Setup

Instead of using keyboard shortcuts you can also use the graphical setup interface. Press `F4` `F4` to open the setup menu or go to 🌎 `/setup` (example). These settings are only changed for your browser, not for any other users.

Setup

Television Simulator '99

Configuration

TV Examples Help

Visual Effects

- [ ] Disable Blur
- [ ] Disable Noise
- [ ] Disable Scanlines
- [ ] Disable Shadow Mask
- [ ] Override Auto Picture Scaling

Shadow Mask Type: Configuration Default

- [ ] Disable Screen Bezel

Bezel Type: Configuration Default

Border Image:

Dimensions

Be careful when changing these values.
I've only tested on a 640x480 resolution.

Scale X:	Scale Y:
100	100
Screen Width:	**Screen Height:**
640	480

Integrations

YouTube API Key:

Public URL for Remote:

Save Cancel

Reset to default

Channels

Channel Properties

`number` `required`

Type: `number`

The channel number. Must be unique across all channels. Negative channels are allowed and will be treated as line inputs. Channel `0` is displayed as “MENU”.

`abbr`

Type: `string`

A short abbreviation for the channel. Will be displayed in program guides and in OSDs that support it. If not provided it may show up as “UNKNOWN”.

`name`

Type: `string`

The full name of the channel.

`title`

Type: `string`

The title of the currently playing program.

`description`

Type: `string`

A short description of the currently playing program.

---

icon

Type: `string`

The URL of an icon to display for the channel. In some OSDs this icon will display when the info screen is visible.

---

blur

Type: `number`

The amount of blur to apply to the channel's video output, in pixels. Overrides the global blur setting.

---

noise

Type: `number`

The amount of noise to apply to the channel's video output, from `0.0` to `1.0`. Overrides the global noise setting.

---

noiseBlendMode

Type: `normal` | `multiply` | `screen` | `overlay` | `darken` | `lighten` | `color-dodge` | `color-burn` | `hard-light` | `soft-light` | `difference` | `exclusion` | `hue` | `saturation` | `color` | `luminosity` | `plus-darker` | `plus-lighter`

The CSS blend mode to use for the noise effect. Defaults to `screen`.

---

backgroundAudio

Type: `string`

The URL of an audio file to play in the background while the channel is active.

---

backgroundAudioOptions

Options for customizing the background audio playback.

`volume`

Type: `number` | `false`
Default: `1`

Volume level from `0.0` (muted) to `1.0` (full volume). If set to `false` or `0`, the audio will be muted.

`filterType`

Type: `'lowpass'` | `'highpass'` | `'bandpass'` | `'lowshelf'` | `'highshelf'` | `'notch'` | `'allpass'`

Type of audio filter to apply; if none is specified, no filtering will be applied.

`filterFrequency`

Type: `number`
Default: `0`

The frequency, in Hz, for the audio filter.

`noise`

Type: `number` | `false`
Default: `0`

Amount of noise to add to the audio, from `0` (none) to `1` (maximum). Set to `false` to disable.

`corsMitigation`

Type: `true` | `false`

If enabled, disables audio processing for sources without proper CORS headers. Filter options will not work if this is set.

`description`

Type: `string`

A description of the audio source. Use the station name for live radio, or song and artist for music.

`nowPlayingUrl`

Type: `string`

URL to fetch “now playing” information.

`nowPlayingProvider`

Type: `string`

Provider to fetch “now playing” information from. See documentation for supported providers.

`shufflePlaylist`

Type: `true` | `false`

If the background audio is a playlist, enables shuffling when set to `true`.

Compatibility

TVS is designed to run on all platforms that support the latest version of the Chromium browser. Browsers like Google Chrome, Brave, Edge, etc. will work. Firefox and Safari may run but support is not guaranteed.

> **Tip**
>
> As of publication, Baseline 2023 web features are observed. If you're using an out of date browser you may encounter problems.

Browser Configuration

For the best user experience, you'll likely need to change permission settings to allow autoplaying audio and video. Since the project is designed around flipping through channels, clicking a play button would make it less enjoyable, therefore autoplay is essential.

Without modifying any browser settings, the simulator starts off muted and you must press M to unmute, which hopefully will meet the browser's threshold for interaction to allow videos and audio to play. Similarly, to connect a phone remote you must press R to display the QR code, which allows audio to play unmuted.

Keyboard Shortcuts

Press `F1 F1` to view the interactive keyboard help page.

Category	Combo	Description	Override ID
General	`P P`	Power	`power`
General	`R R`	Toggles display of Remote pairing code	`toggleRemotePairingCode`
General	`F1 F1`	Help (this page)	`help`
General	`F4 F4`	Go to the Setup page	`setup`
General	`V V`	Displays version information	`version`
General	`AltV AltV`	About TVS	`about`
Picture Adjustments	`AltR AltR`	Reset Picture	`resetPicture`
Picture Adjustments	`Alt1 Alt1`	Toggle Scanlines visual effect	`toggleScanlines`
Picture Adjustments	`Alt2 Alt2`	Toggle Noise visual effect	`toggleNoise`
Picture Adjustments	`Alt3 Alt3`	Toggle Blur visual effect	`toggleBlur`
Picture Adjustments	`Alt4 Alt4`	Toggle Shadow Mask visual effect	`toggleShadowMask`
Picture Adjustments	`Alt5 Alt5`	Toggle Static on channel change	`toggleChangeChannelNoise`
Picture Adjustments	`Alt6 Alt6`	Toggle Bezels visual effect	`toggleBezels`

Category	Combo	Description	Override ID
Picture Adjustments	`Alt7 Alt7`	Toggle automatic screen scaling	`toggleAutoScale`
Picture Adjustments	`ShiftAlt4 ShiftAlt4`	Toggle Shadow Mask Type	`toggleShadowMaskType`
Picture Adjustments	`ShiftAlt6 ShiftAlt6`	Toggle Bezel Type	`toggleBezelType`
Picture Adjustments	`ShiftAltLeft ShiftAltLeft`	Scales screen horizontal, negative	`scaleDownX`
Picture Adjustments	`ShiftAltRight ShiftAltRight`	Scales screen horizontal, positive	`scaleUpX`
Picture Adjustments	`ShiftAltUp ShiftAltUp`	Scales screen vertical, positive	`scaleUpY`
Picture Adjustments	`ShiftAltDown ShiftAltDown`	Scales screen vertical, negative	`scaleDownY`
Tuner	`Up Up`	Channel Up	`channelUp`
Tuner	`Down Down`	Channel Down	`channelDown`
Tuner	`I I`	Info	`info`
Tuner	`0 0 Numpad0 Numpad0`	Number 0	`numpad0`
Tuner	`1 1 Numpad1 Numpad1`	Number 1	`numpad1`
Tuner	`2 2 Numpad2 Numpad2`	Number 2	`numpad2`
Tuner	`3 3 Numpad3 Numpad3`	Number 3	`numpad3`
Tuner	`4 4 Numpad4 Numpad4`	Number 4	`numpad4`
Tuner	`5 5 Numpad5 Numpad5`	Number 5	`numpad5`
Tuner	`6 6 Numpad6 Numpad6`	Number 6	`numpad6`

Category	Combo	Description	Override ID
Tuner	`7 7 Numpad7 Numpad7`	Number 7	`numpad7`
Tuner	`8 8 Numpad8 Numpad8`	Number 8	`numpad8`
Tuner	`9 9 Numpad9 Numpad9`	Number 9	`numpad9`
Receiver	`Left Left`	Volume Down	`volumeDown`
Receiver	`Right Right`	Volume Up	`volumeUp`
Receiver	`M M`	Mute	`mute`
Video Game Controls	`E E`	Reset Console	-
Video Game Controls	`U U`	Up	-
Video Game Controls	`H H`	Left	-
Video Game Controls	`J J`	Down	-
Video Game Controls	`K K`	Right	-
Video Game Controls	`D D`	Start	-
Video Game Controls	`F F`	Select	-
Video Game Controls	`X X`	A	-
Video Game Controls	`Z Z`	B	-

Category	Combo	Description	Override ID
Video Game Controls	`S S`	Y	-
Video Game Controls	`A A`	X	-
Video Game Controls	`Q Q`	L	-
Video Game Controls	`W W`	R	-

Overriding Keyboard Shortcuts

In your configuration file you can override any keyboard command using the `inputMapping` section. For any key you override, the default key will be unassigned.

```
inputMapping:
  volumeUp: q
  volumeDown: a
  channelUp: alt+up
  channelDown: alt+down
```

> **Note**
>
> The browser may have some reserved key combos that can't be overridden. If you're having trouble with a key combo, try another one.

The combos supported should be in lower-case; we use Keymaster's syntax and key names. Use the "Override ID" from the table above to find the command you want to override.

Content Engine Primer

> **Tip**
>
> Read this first to understand how to use the reference documentation effectively!

Examples

Each content engine has its own page, and they are structured similarly. We begin with a brief summary and a screenshot followed by YAML code that shows how to use the engine in a channel.

For instance, you might see something like this:

```
image: /content/images/my-image.jpg
```

To use it, put it in a channel:

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  image: /content/image/my-image.jpg # <-- the content engine goes here
```

Or in a layout:

```
channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  loop:
  - image: /content/images/my-image.jpg # <-- here
    duration: 10
  - image: /content/images/my-image-2.jpg
    duration: 5
```

Or even in a reference:

```yaml
refs:
  ad1:
    image: /content/images/my-image.jpg # <-- here

channels:

- number: 123
  name: My channel
  abbr: EXAMPLE
  ref: ad1
```

Properties

After the examples you'll get an exhaustive list of *unique* properties for each content engine. These properties can be used to customize the behavior and appearance of the engine in your channels and layouts.

Common Properties

Properties that can apply to any engine won't be displayed on every page but instead are available below:

`interruptsBackgroundAudio`

Type:: `true` | `false`

Whether the component takes precedence over the channel's background audio and mutes it while being displayed.

Shorthand Properties

`shorthand` properties are marked in blue and are properties that can be set using shorthand syntax; instead of saying:

```yaml
image:
  src: /content/images/my-image.jpg
```

you can simply write:

```
image: /content/images/my-image.jpg
```

You can’t provide any other options with this shorthand syntax, but it is useful for simple cases such as when you just want to display an image or a video without any additional configuration.

Default Values

If there are multiple options for an optional property the default value will be labeled `like this`. Other available values will be in `gray`.

Images

Television Simulator can display any image your browser can render.

```
image: /content/images/my-image.jpg
```

```
image:
  src: /content/images/my-image.jpg
  aspectRatioBehavior: cover
```

Properties

`src` shorthand required

Type: `string`

The URL of the image to display, relative to the content root or absolute.

`aspectRatioBehavior`

Type: `cover` | `contain` | `stretch` | `repeat` | `blur`

How the image should be displayed in relation to the screen size. `contain` will fit the image inside the screen, `cover` will fill the screen while cropping the image, `stretch` will stretch the image to fill the screen, `repeat` will tile the image, and `blur` will apply a blur effect to edges of the image.